Universidad ORT

Inteligencia Artificial

Entrega de Obligatorio

|  |  |
| --- | --- |
| Una persona con gafas de sol  Descripción generada automáticamente con confianza media | Un hombre con lentes y corbata  Descripción generada automáticamente |
| Federico Alonso | Horacio Ábalos |

12 de diciembre de 2022

[Link al Repositorio](https://github.com/federicoalonso/IA)

Contenido

[Declaración de autoría: 2](#_Toc121620255)

[Introducción 3](#_Toc121620256)

[Cartpole 3](#_Toc121620257)

[Análisis del tipo de solución y metodología de trabajo 3](#_Toc121620258)

[CartPole 4](#_Toc121620259)

[Versión 1 4](#_Toc121620260)

[Versión 2 4](#_Toc121620261)

[Versión 3 5](#_Toc121620262)

[Versión 4 6](#_Toc121620263)

[Versión 5 - AgentPoleDimensions 8](#_Toc121620264)

[Versión 5.1 - Modificación de la discretización 15](#_Toc121620265)

[Versión 6 – Posición del cart y velocidad AgentCartAndPoleDimensions 18](#_Toc121620266)

[Resultados de la versión 5 22](#_Toc121620267)

[Errores Conocidos 23](#_Toc121620268)

[Opciones de mejora 23](#_Toc121620269)

[Lecciones Aprendidas 23](#_Toc121620270)

# Declaración de autoría:

Nosotros, Horacio Ábalos y Federico Alonso, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

1. La obra fue producida en su totalidad mientras realizábamos el obligatorio de Inteligencia Artificial;
2. Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
3. Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
4. En la obra, hemos acusado recibo de las ayudas recibidas;
5. Cuando la obra se basa en trabajo realizado juntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
6. Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

|  |  |
| --- | --- |
|  |  |
| Alonso Fuentes, Federico Nicolás | Abalos Cambre, Horacio Mathías |
| 12 de diciembre de 2022 | 12 de diciembre de 2022 |

# Introducción

El presente documento se brinda como especificación técnica y explicación del trabajo realizado por el equipo para la entrega del obligatorio de la materia Inteligencia Artificial. La propuesta recibida solicita resolver los problemas *2048* y *CartpoleV1*.

## Cartpole

OpenAI nos propone el ejercicio de conservar verticalmente un objeto con forma de palo sobre un carro que solamente puede moverse en dos dimensiones, izquierda (0) y derecha (1). Dado que en todo momento existen dos posibles acciones (izquierda y derecha), la recompensa asociada a cada paso del entorno será +1 y, según la documentación del OpenAI, consideraremos resuelto el problema cuando se obtenga una recompensa acumulada de 500.

Un luz de freno

Descripción generada automáticamente con confianza media

Para este ejercicio el espacio de observación es el siguiente

|  |  |
| --- | --- |
| Action Space | Discrete(2) |
| Observation Shape | (4,) |
| Observation High | [4.8 inf 0.42 inf] |
| Observation Low | [-4.8 -inf -0.42 -inf] |

Para información más detallada se sugiere visitar la [documentación de OpenAI](https://www.gymlibrary.dev/environments/classic_control/cart_pole/)

# Análisis del tipo de solución y metodología de trabajo

Entendemos que en el problema propuesto no existen datos sobre los cuales un agente pueda aprender y, considerando que solamente disponemos de un entorno sobre el que podemos realizar acciones, el enfoque pertinente para esta solución es *Reinforcement Learning*.

En lo referente a la metodología de trabajo, utilizaremos una aproximación iterativa incremental, a través de la cual buscaremos dividir las tareas en subtareas más pequeñas sobre las cuales podamos tomar decisiones de mejora de manera específica. De esta forma, lograremos obtener una solución para posteriormente intentar mejorarla.[[1]](#footnote-1)

# CartPole

## Versión 1

Esta iteración simplemente ofrece las condiciones de trabajo para comenzar con una aproximación más refinada a partir de las siguientes.

Puntualmente se establece una meta muy corta, pero fundamental: el algoritmo funciona y guarda los valores en la tabla que almacena los valores de Q considerando todas las dimensiones de la observación (velocidad angular, posición del carro, velocidad del carro, ángulo del palo).

Se comienza a partir de la versión 2 con el análisis de entrenamiento.

## Versión 2

El algoritmo utiliza ciertos valores (bien podrían ser parametrizables), los cuales comienzan arbitrariamente siendo:

1. *Learning rate* 0.1
2. *Discount* 0.85
3. Discretización de los espacios de observación (las 4 dimensiones) en 50 *buckets* (100 exceden la memoria utilizable por la ejecución)

A partir de los valores anteriores se verifica que no existen problemas de compilación ni de recursos de hardware para comenzar con un entrenamiento extenso. Por este motivo se decide implementar un entrenamiento de 10.000.000.000 (diez mil millones de episodios).

Cada cierta cantidad de iteraciones se guardará una *Qtable* para analizar un desempeño utilizando un agente que simule el juego.

Luego de aguardar la ejecución durante 2 días se decide suspender el entrenamiento (transcurridos 80 millones de episodios) y se procede a ejecutar la simulación.

No se dispone de una impresión de pantalla de esta *performance*, pero utilizando la ultima *Qtable* generada (a los 80 millones de episodios de entrenamiento) se obtiene un *reward* promedio superior a 9 en la simulación de 100 episodios de juego. Esto significa que el desempeño apenas fue superior que a las acciones absolutamente elegidas de manera aleatoria.

Analizando el desempeño obtenido, es claro que una discretización de semejante tamaño no es conveniente ya que, en una tabla de 4 dimensiones de estas características más las 2 propias de cada acción disponible, genera un universo de 12.500.000 posibles valores de Q. Inclusive considerando que se generaron 80 millones de episodios de entrenamiento, puede existir la posibilidad que alguna combinación de los estados discretos no haya sido observada.

Como consecuencia lógica, se intentará reducir el tamaño de la discretización de las observaciones.

## Versión 3

Comienza el ciclo de entrenamientos con la discretización reducida a 20 *buckets* por dimensión de observación (se reduce el espacio de posibles combinaciones a 320.000).

Adicionalmente, se decide implementar sesiones de entrenamiento menos extensas que en la versión 2 para que las iteraciones sean mas cortas y significativas (de 1 a 10 millones).

Se realiza la simulación del juego y se obtiene exactamente el mismo resultado que en la versión 2.

Utilizando la Qtable *5600000-qtable.npy* se obtiene el siguiente desempeño en 10 episodios de simulación:

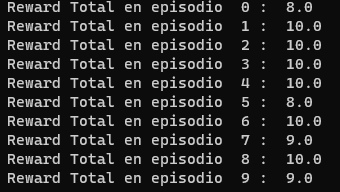


Imagen - Desempeño de 10 episodios simulados con 5600000-qtable.npy

Lógicamente este no es el resultado esperado, por lo que se verifica que la *Qtable* no fue correctamente generada. A continuación, se expone uno de los episodios simulados imprimiento (a) la acción realizada y (b) la *Qtabe* en el estado discreto de cada paso, así como (c) el *reward* total obtenido en el episodio:

Texto

Descripción generada automáticamente

Imagen - Detalle de la simulación con 5600000-qtable.npy

Resulta evidente que se debe modificar el código por lo que se concluye esta iteración dejando los siguientes puntos de mejora para la iteración 4:

1. Verificar que la *Qtable* esté inicializándose y calculándose correctamente
2. Modificar el *Learning Rate* y *Discount* utilizados
3. Incluir algún tipo de métrica para analizar el entrenamiento realizado

## Versión 4

Comienza la revisión del código y se confirma que no estaba calculándose, inicializándose y guardándose correctamente la *Qtable*. Una vez corregido el error se procede a modificar los parámetros de *Learning Rate* y *Discount* (representando el valor asignado a la importancia de los episodios futuros frente a los actuales), ambospara varias combinaciones pertenecientes al intervalo (0;1].

Se incluye código para generar métricas de *rewards* máximas, mínimas y promedio obtenidas en base a los episodios de entrenamiento, se guardan las *Qtables* sincronizadamente de forma que observando el grafico pueda ser posible utilizar la *Qtable* asociada al mejor desempeño.

Una vez introducidos los cambios se ejecuta un entrenamiento de 10 millones de episodios, pero no se observa una mejoría en el desempeño del modelo ya que en promedio los *rewards* continúan siendo bajos.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Gráfica - Rewards máximo, promedio y mínimo para la versión 2 con 2 millones de episodios de entrenamiento

La Gráfica 1 nos muestra que el mejor *reward* máximo obtenido se produce aproximadamente a los 2 millones de episodios pero que posteriormente el modelo empeora su desempeño.

Se corre una iteración de 100 episodios con la *Qtable* guardada a los 100 mil y a los 2 millones de episodios, así como con la última generada a los 10 millones. En todos los casos se observa que el *reward* promedio es 10. Se adjunta solamente una de las impresiones (todas las obtenidas eran exactamente iguales) de la simulación con 100 episodios. La impresión corresponde al último episodio:

Texto

Descripción generada automáticamente

Imagen - Impresion de Qtable[discrete\_state] para la tabla generada a los 2 millones de episodios

Se observa que ambas acciones tienen la misma “consecuencia” para nuestro modelo dado que tienen el mismo valor asociado, esto genera algo de desconcierto dado que es evidente que no puede resultar el mismo valor para una determinada observación, es decir: suponiendo que el carro se este desplazado a la derecha (sin importar su ubicación), con el palo en ángulo mayor a 0 y velocidad angular positiva, no puede ser lógicamente admisible que el modelo entienda indistinta cualquier acción.

De lo anterior se desprende que, entre otras cosas que puedan alterar el funcionamiento, o bien:

1. el código no funciona como se espera en lo referente a la *Qtable* (sin embargo, parece estar calculando y guardando valores);
2. la cantidad de episodios de entrenamiento no fue suficiente (esto básicamente siempre es real);
3. se está introduciendo mucho ruido a través de dimensiones de observaciones que no son realmente necesarias

Claro está que para una primera versión de la solución se podría solamente considerar el ángulo del palo y su velocidad angular, ya que la velocidad del carro y su posición parecen no ser tan relevantes para que el juego acabe. Si bien esto es una simplificación, entendemos que es válida para acercarnos a una solución que funcione (o al menos mejore el resultado hasta ahora obtenido) y posteriormente incluir las restantes dimensiones de observación.

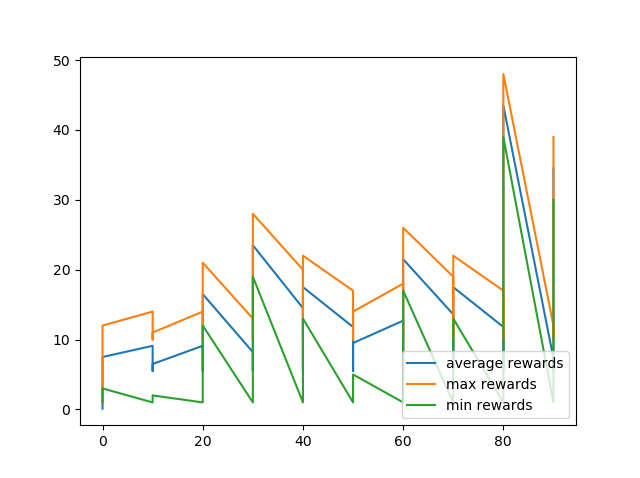
## Versión 5 - AgentPoleDimensions

Considerando que la *Qtable* de la versión anterior estaba cargada con el mismo valor para ambas acciones, decidimos recomenzar el código. A partir de este momento, modificamos la implementación de la discretización para utilizar *KBinsDiscretizer* importándolo desde *sklearn.preprocessing* y el método de decrementar los valores de *Learning Rate* y *Eploration Rate* basados en una solución propuesta en el tutorial *Balancing CartPole with Machine Learning, Python* (Saito, 2018).

Conforme mencionado anteriormente, debido a que se percibe que el modelo no está aprendiendo de las iteraciones realizadas, se decide comenzar nuevamente, cambiando el enfoque para una solución que solamente considere la velocidad angular y ángulo del *pole* para, posteriormente, incluir la velocidad y ubicación del *cart* en las dimensiones de la *Qtable*.

El código restante se reescribe con mayor atención a la reutilización y considerando criterios de mantenibilidad para su posterior modificación y evolución.

Se procede a graficar los *rewards* obtenidos y, tras correr una iteración de entrenamiento de apenas 100 episodios, se genera la Grafica XX que nos muestra claramente como los *rewards* promedio y mínimo modificaron el patrón hasta ahora percibido y pasan a acompañar al *reward* máximo.



Gráfica - Rewards máximo, promedio y mínimo para la version 5 en 100 episodios de entrenamiento

A partir de la gráfica, se selecciona la *Qtable* guardada en la iteración 80. Al correr una simulación de 100 episodios se obtiene:

* Episodio: 0 Reward obtenido: 29.0
* Episodio: 1 Reward obtenido: 14.0
* Episodio: 2 Reward obtenido: 31.0
* Episodio: 3 Reward obtenido: 39.0
* Episodio: 4 Reward obtenido: 14.0
* Episodio: 5 Reward obtenido: 37.0
* Episodio: 6 Reward obtenido: 45.0
* Episodio: 7 Reward obtenido: 37.0
* Episodio: 8 Reward obtenido: 15.0
* Episodio: 9 Reward obtenido: 63.0
* Episodio: 10 Reward obtenido: 42.0
* Episodio: 11 Reward obtenido: 27.0
* Episodio: 12 Reward obtenido: 37.0
* Episodio: 13 Reward obtenido: 93.0
* Episodio: 14 Reward obtenido: 27.0
* Episodio: 15 Reward obtenido: 35.0
* Episodio: 16 Reward obtenido: 51.0
* Episodio: 17 Reward obtenido: 35.0
* Episodio: 18 Reward obtenido: 71.0
* Episodio: 19 Reward obtenido: 39.0
* Episodio: 20 Reward obtenido: 41.0
* Episodio: 21 Reward obtenido: 46.0
* Episodio: 22 Reward obtenido: 27.0
* Episodio: 23 Reward obtenido: 78.0
* Episodio: 24 Reward obtenido: 76.0
* Episodio: 25 Reward obtenido: 39.0
* Episodio: 26 Reward obtenido: 17.0
* Episodio: 27 Reward obtenido: 126.0
* Episodio: 28 Reward obtenido: 63.0
* Episodio: 29 Reward obtenido: 33.0
* Episodio: 30 Reward obtenido: 15.0
* Episodio: 31 Reward obtenido: 41.0
* Episodio: 32 Reward obtenido: 44.0
* Episodio: 33 Reward obtenido: 78.0
* Episodio: 34 Reward obtenido: 33.0
* Episodio: 35 Reward obtenido: 43.0
* Episodio: 36 Reward obtenido: 54.0
* Episodio: 37 Reward obtenido: 43.0
* Episodio: 38 Reward obtenido: 45.0
* Episodio: 39 Reward obtenido: 31.0
* Episodio: 40 Reward obtenido: 56.0
* Episodio: 41 Reward obtenido: 73.0
* Episodio: 42 Reward obtenido: 50.0
* Episodio: 43 Reward obtenido: 59.0
* Episodio: 44 Reward obtenido: 52.0
* Episodio: 45 Reward obtenido: 15.0
* Episodio: 46 Reward obtenido: 82.0
* Episodio: 47 Reward obtenido: 49.0
* Episodio: 48 Reward obtenido: 37.0
* Episodio: 49 Reward obtenido: 33.0
* Episodio: 50 Reward obtenido: 27.0
* Episodio: 51 Reward obtenido: 55.0
* Episodio: 52 Reward obtenido: 29.0
* Episodio: 53 Reward obtenido: 33.0
* Episodio: 54 Reward obtenido: 46.0
* Episodio: 55 Reward obtenido: 33.0
* Episodio: 56 Reward obtenido: 68.0
* Episodio: 57 Reward obtenido: 71.0
* Episodio: 58 Reward obtenido: 37.0
* Episodio: 59 Reward obtenido: 60.0
* Episodio: 60 Reward obtenido: 31.0
* Episodio: 61 Reward obtenido: 37.0
* Episodio: 62 Reward obtenido: 50.0
* Episodio: 63 Reward obtenido: 16.0
* Episodio: 64 Reward obtenido: 37.0
* Episodio: 65 Reward obtenido: 31.0
* Episodio: 66 Reward obtenido: 53.0
* Episodio: 67 Reward obtenido: 59.0
* Episodio: 68 Reward obtenido: 55.0
* Episodio: 69 Reward obtenido: 61.0
* Episodio: 70 Reward obtenido: 33.0
* Episodio: 71 Reward obtenido: 51.0
* Episodio: 72 Reward obtenido: 37.0
* Episodio: 73 Reward obtenido: 47.0
* Episodio: 74 Reward obtenido: 31.0
* Episodio: 75 Reward obtenido: 33.0
* Episodio: 76 Reward obtenido: 35.0
* Episodio: 77 Reward obtenido: 35.0
* Episodio: 78 Reward obtenido: 40.0
* Episodio: 79 Reward obtenido: 41.0
* Episodio: 80 Reward obtenido: 69.0
* Episodio: 81 Reward obtenido: 29.0
* Episodio: 82 Reward obtenido: 31.0
* Episodio: 83 Reward obtenido: 60.0
* Episodio: 84 Reward obtenido: 12.0
* Episodio: 85 Reward obtenido: 41.0
* Episodio: 86 Reward obtenido: 19.0
* Episodio: 87 Reward obtenido: 27.0
* Episodio: 88 Reward obtenido: 29.0
* Episodio: 89 Reward obtenido: 45.0
* Episodio: 90 Reward obtenido: 33.0
* Episodio: 91 Reward obtenido: 17.0
* Episodio: 92 Reward obtenido: 33.0
* Episodio: 93 Reward obtenido: 118.0
* Episodio: 94 Reward obtenido: 26.0
* Episodio: 95 Reward obtenido: 37.0
* Episodio: 96 Reward obtenido: 37.0
* Episodio: 97 Reward obtenido: 39.0
* Episodio: 98 Reward obtenido: 35.0
* Episodio: 99 Reward obtenido: 56.0

Los resultados de este enfoque son mucho más alentadores por lo que se procede a entrenar 2 mil episodios para posteriormente verificar su desempeño. Durante la ejecución, se observa que el entrenamiento arroja un episodio que es truncado al superar la cantidad de *reward* total obtenida superior a lo solicitado por el ambiente (500). Esto sucede a los 154 episodios de entrenamiento, por lo que se decide utilizar la *Qtable* generada (se había incluido en el código una condición para guardar la *Qtable* si el episodio de entrenamiento era truncado por el ambiente, es decir, si el agente había ganado el juego). Al correr 150 episodios de simulación con la *Qtable* anterior se obtuvo el siguiente resultado:

Texto

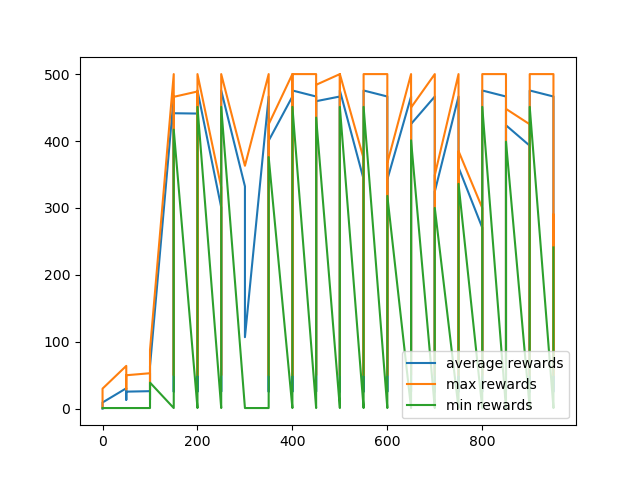
Descripción generada automáticamente

Imagen - Resultado de simulación con Qtable generada a los 154 episodios de entrenamiento de la versión 5

Como podemos apreciar, se perdieron solamente 3 de los 150 episodios generados, esto induce a pensar que el error cometido es del 2% (lógicamente solo considerando los 150 episodios simulados).

Se observa un comportamiento particular: No se toma ninguna acción para que el carro no se exceda de los límites permitidos. Esto se deriva de omitir la posición del carro en las dimensiones de observación utilizada, tarea reservada para la versión 6.

A continuación, se adjunta la gráfica del entrenamiento realizado con 1000 iteraciones:



Gráfica - Rewards máximo, promedio y mínimo en 2000 episodios de entrenamiento de la versión 5

A continuación, se adjunta la gráfica del entrenamiento realizado con 2000 iteraciones:

Gráfico, Histograma

Descripción generada automáticamente

Gráfica - Rewards máximo, promedio y mínimo en 2000 episodios de entrenamiento de la versión 5

Se realiza también un entrenamiento de 10000 iteraciones para verificar si el desempeño mejora con más entrenamiento, grafica adjunta:

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

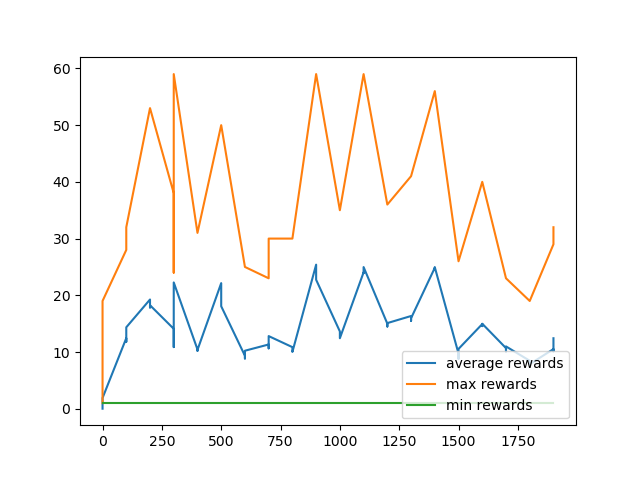
Gráfica - Reward máximo, promedio y mínimo para 10000 iteraciones de entrenamiento en la versión 5

Si bien existe una mejora y conocemos la existencia de máximos y mínimos locales, no se puede observar en las gráficas 3, 4 y 5 un cambio tan significativo entre el resultado de los 1000, 2000 o 10000 episodios entrenados. Se entiende que sería conveniente entrenar el modelo con un numero de iteraciones muy superior (en el orden de los 20 millones) para verificar la existencia de posibles valores locales en los que el algoritmo se pueda haber estancado, y para maximizar la ocurrencia de las observaciones de los episodios en el espacio muestral.

La versión 5 se considera una versión entregable de este desafío, pero se intentará evolucionar la entrega a una versión 6 que considere la velocidad del carro y su posición con el objetivo de no permitir que se trunque el episodio por la posición. Adicionalmente, es un desafío interesante conseguir simular solamente un episodio que sea truncado únicamente por un tiempo máximo a ser determinado.

## Versión 5.1 - Modificación de la discretización

Se intenta ajustar la discretización para un refinamiento más fino con 24 *buckets* en cada dimensión observada (velocidad angular y ángulo del pole).



Gráfica - Rewards máximo, promedio y mínimo para discretización de 24 buckets en dimensiones del pole

Con un refinamiento de 10 *buckets* por dimensión los resultados parecen ser un poco menos volátiles, sin embargo, no es un cambio relevante en el rendimiento.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfica - Rewards máximo, promedio y mínimo para discretización de 10 buckets en dimensiones del pole

Con un refinamiento de 2 *buckets* por dimensión Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Gráfica - Rewards máximo, promedio y mínimo para discretización de 2 buckets en dimensiones del pole

De acuerdo con lo observado en las gráficas 6, 7 y 8 varios intentos fueron realizados con menor y mayor cantidad de *buckets*, pero no se observa una mejora significativa del desempeño en cuanto al refinamiento, se opta por mantener los valores inicialmente utilizados y se procede a integrar las dimensiones de observación relacionadas al *cart*.

## Versión 6 – Posición del cart y velocidad AgentCartAndPoleDimensions

Se integran las dimensiones del *cart* y con apenas 2 mil episodios de entrenamiento no se puede distinguir una mejora en el desempeño comparándolo contra la versión 5.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Gráfica - Reward máximo, promedio y mínimo para 2000 episodios entrenados considerando las dimensiones del cart, discretización de 12 buckets por dimensión

Se intenta con un refinamiento más grueso, con 6 *buckets* por dimensión del *cart*. En esta gráfica se observa un intento de continuar aprendiendo en un desempeño cuya grafica del *reward* parece tender a subir.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Gráfica - Reward máximo, promedio y mínimo para 2000 episodios entrenados considerando las dimensiones del cart, discretización de 6 buckets por dimensión

Para verificar el comportamiento del desempeño se correrán 2 millones de episodios para comparar la gráfica correspondiente a la versión 5.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfica - Reward máximo, promedio y mínimo para 2 millones de episodios entrenados considerando las dimensiones del cart, discretización de 6 buckets por dimensión

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfica - Reward máximo, promedio y mínimo para 2 millones de episodios entrenados considerando las dimensiones del cart, discretización de 10 buckets por dimensión

De la observación de las gráficas 9, 10, 11 y 12, surge que la integración de más dimensiones de observación introduce ruido en el modelo. Entendemos que, dado que no concluye ningún episodio de entrenamiento por ser truncado por el ambiente al alcanzar el máximo reward establecido no es, ni siquiera, necesario poner a prueba simulaciones de la versión 6 para compararla con la versión 5. Por lo tanto, se decide analizar los resultados de la versión 5.

## Resultados de la versión 5

A lo largo de la solución de esta tarea realizamos varios entrenamientos que arrojaron un episodio siempre inferior al 200 que ya conseguía truncar el juego, es decir, ganar. Consideramos interesante ver como escala esa Qtable en particular con 150 episodios de juego. Poniendo a prueba los mejores entrenamientos obtenidos

|  |  |
| --- | --- |
| 101-qtable.npy | Texto  Descripción generada automáticamente con confianza media |
| 128-qtable.npy | Texto  Descripción generada automáticamente con confianza media |
| 137-qtable.npy | Interfaz de usuario gráfica, Texto  Descripción generada automáticamente con confianza media |
| 154-qtable.npy | Texto  Descripción generada automáticamente |
| 157-qtable.npy | Texto  Descripción generada automáticamente con confianza media |
| 113-qtable.py | Texto  Descripción generada automáticamente |

Para verificar el desempeño en un escenario de mayor rigurosidad (a pesar de no representar una muestra significativa del espacio de observaciones posible), simulamos 10000 episodios con las *Qtables* que mejor se desempeñaron:

|  |  |
| --- | --- |
| 101-qtable.npy |  |
| 113-qtable.npy |  |
| 9999-qtable.npy |  |

Simulación absurda de 1000000 episodios jugados:

|  |  |
| --- | --- |
| 101-qtable.npy |  |
| 113-qtable.npy |  |
| 9999-qtable.npy |  |

# Errores Conocidos

# Opciones de mejora

* Sería interesante considerar el paso del tiempo en el cartpole
* Entendemos que los entrenamientos podrían haber sido de una cantidad mayor de episodios
* El seteo de los parámetros podría haber sido investigado en profundidad
* El código en líneas generales puede ser mejorado
* Se presentan una cantidad de Agentes que comparten mucha codificación

# Lecciones Aprendidas

Claramente en el ejercicio de Cartpole se disponía de información que no era relevante para la solución. Alternativamente, no supimos como trabajar con ella y no tuvimos tiempo de investigar una solución que considere la posición y velocidad del cart.

Entendemos que un problema relativamente sencillo para un ser humano promedio como equilibrar un palo es algo desafiante de programar abstraídamente del ambiente sin caer en una solución que se acople demasiado en él.

Un problema de pocas dimensiones de observación puede escalar muchísimo en tiempo y en memoria cuando existen representaciones de continuidad asociadas a alguna de ellas. Esto requiere de un análisis que no siempre es trivial y se basa en la experiencia para saber como discretizar correctamente sin precipitarse en una omisión de información o en una discretización demasiado gruesa que no permita tomar decisiones adecuadas

1. Cada una de las versiones mencionadas en este documento se relaciona con una rama del repositorio dónde se pueden encontrar los recursos de cada etapa (Agente, Agente para simular el juego, Qtables generadas en el entrenamiento, graficas, etc). [↑](#footnote-ref-1)