Aprendizaje por refuerzo en planeación de movimiento para coche autónomo

Mariana R. Hernández Rocha
Maestría en Ciencias en Computación
ITAM
CDMX, México
marianarhr13@gmail.com

***Abstract*—En este documento se aborda el tema de planeación de movimiento para un coche autónomo AWS DeepRacer utilizando como método de solución el aprendizaje por refuerzo.**

Se utilizó la plataforma de Amazon Web Services para las pruebas del modelo.

I. INTRODUCCIÓN

A partir del surgimiento del *DARPA Urban Challenge* [1], el interés por los vehículos autónomos ha incrementado de manera significativa. El dicha competencia, los equipos debían construir un vehículo autónomo capaz de conducir en el tráfico, realizar maniobras complejas como incorporarse a una vía, rebasar otros coches, estacionarse y negociar en intersecciones.

Y la razón por la que tanto la academia como empresas privadas están apostando por los coches autónomos es que esta nueva tecnología significa un cambio significativo en muchos ámbitos. Desde una reducción en los índices de accidentes provocados por falta de atención hasta menores emisiones de gases como dióxido de carbono (consecuencia del exceso de tráfico en zonas urbanas).

Esto aunado al continuo interés y desarrollo de métodos como el Aprendizaje de Máquina ha abierto nuevos caminos para explorar métodos de solución inovadores. Uno de estos métodos es el aprendizaje por refuerzo, el cual ha tenido resultados fructíferos para áreas como la robótica.

En general, se han obtenido buenos resultados de utilizar técnicas de aprendizaje de máquina en aplicaciones de planeación de movimiento para agentes [2] [3].

En este documento, se hablará del uso del aprendizaje por refuerzo como método para la planeación de movimiento de un coche autónomo AWS Deep Racer en un ambiente simulado utilizando las herramientas proporcionadas por la plataforma.

El documento se divide en seis secciones:

1) En la introducción se presenta la motivación para desarrollar herramientas de Aprendizaje de Máquina que resuelvan problemas de robótica.
2) La segunda sección muestra la complejidad del problema de planeación de movimientos.
3) La tercer sección explica en qué consiste el aprendizaje por refuerzo y el algoritmo utilizado en este proyecto.
4) La sección cuatro describe al modelo, explica en qué consisten las variables involucradas.
5) En la quinta sección se detalla el proceso de entrenamiento del modelo.
6) La sección seis entrega resultados de entrenamientos realizados con diferentes parámetros.
7) Finalmente, la sección siete contiene comentarios respecto a los resultados obtenidos.

II. MOTION PLANNING

Para poner en contexto el problema de planeación de movimiento *(motion planning)* nos basaremos en el trabajo de LaValle [4]. Suponga que desea encontrar un camino de q_1 a q_G en C_{free} donde el espacio completo se representa como $C = C_{free} \cup C_{obs}$. Esta representación se muestra en la figura 1.

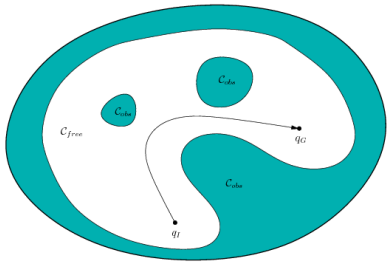

Fig. 1. Motion planning

A pesar de parecer un problema bastante simple, se ha probado [5] que es un problema que pertenece a la categoría PSPACE-hard, dicha demostración acota el problema por debajo mientras que la cota superior de este problema lo coloca en el espacio PSPACE [6]. Lo que convierte a este problema en un problema NP completo.

Cada una de las clases de complejidad mencionadas anteriormente contiene un conjunto de problemas para los cuales existen algoritmos que requieren recursos pertenecientes a los límites propios de la clase.

Que el problema de motion planning sea de tipo NP conlleva a que su solución puede ser comprobada en tiempo polinomial por una máquina de Turing pero esto no implica que el problema puede ser resuelto en tiempo polinomial. Además la clasificación NP hard vuelve a este problema al menos tan difícil como los problemas más difíciles de la clase NP (NP

completo). En la figura 2 se muestra un esquema que ilustra la relación entre las clases de problemas.

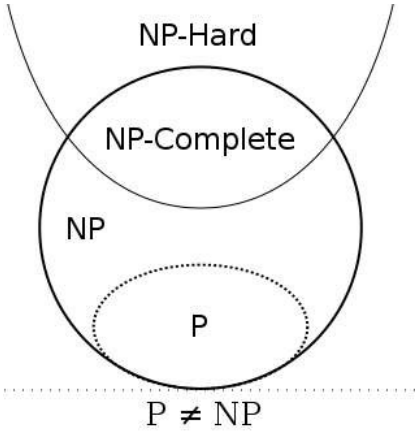

Fig. 2. Complejidad

Estas clasificaiones nos permiten estimar cuánto tiempo o espacio se requiere para resolver distintos problemas. Todo esto bajo el supuesto de que $P \neq NP$, supuesto que no se ha demostrado verídico pero tampoco se ha contradicho.

III. APRENDIZAJE POR REFUERZO

Aprendizaje por refuerzo *(reinforcement learning RL)* es un método de aprendizaje de máquina que resuelve el problema de cómo hacer que un agente autónomo, en este caso un coche autónomo, el cual interactúe en su ambiente mediante sensores y actuadores pueda aprender a escoger las acciones óptimas que lo lleven a lograr sus objetivos y evitar aquellas acciones que interfieran en la obtención de estos.

El agente actúa con base en el estado en el que se encuentra y el entorno retroalimentará con una recompensa y el nuevo estado en el que se encuentra el agente. La interacción entre los distintos componentes del aprendizaje por refuerzo presentan en la figura 3.

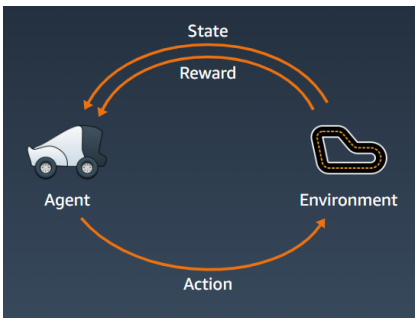

Fig. 3. Esquema de aprendizaje por refuerzo

El agente aprende por prueba y error, inicialmente actuando de manera aleatoria pero eventualmente aprendiendo qué acciones conllevan a recompensas gratificantes.

El algoritmo de aprendizaje por refuerzo utilizado en esta simulación es el de *Optimización por Política Proximal (PPO)* [7] ya que el rendimiento durante el entrenamiento es más rápido. Este algoritmo ha sido probado exitosamente en una gran variedad de problemas, desde control robótico hasta Atari.

Uno de las mayores causas de inestabilidad en *RL* es que los datos de entrenamiento dependen de la política actual ya que el agente genera su propio conjunto de entrenamiento y esto conlleva a que las distribuciones generadas sobre las observaciones están en constante cambio mientras el agente aprende. Además, *RL* presenta una alta sensibilidad a el ajuste de hiperparámetros.

Este algoritmo se clasifica como *policy gradient method* porque no aprende de información almacenada, aprende en línea. Es decir, aprende directamente a partir de lo que sea con lo que el agente se encuentre. Una vez que un lote de experiencia ha sido utilizado para actualizar el gradiente, se descartan los movimientos y la política continúa.

La función de pérdida para los métodos *policy gradient* es la siguiente:

$$L^{PG}(\theta) = \hat{E}_t[\log \pi_\theta(a_t \mid s_t)\hat{A}_t] \tag{1}$$

La primer parte $\log \pi_\theta(a_t \mid s_t)$ es la política es una red neuronal que toma los estados observados del ambiente como entrada y sugiere acciones como salida.

La segunda parte $\hat{A}_t$ se conoce como *advantage function* y trata de estimar el valor relativo de la acción seleccionada en el estado actual. Responde a la pregunta de si la acción que tomó el agente fue mejor de lo que se esperaba. Para calcularla se requieren de dos componentes:

$$\hat{A}_t = \text{Discount rewards - Value function}$$

- *Discount rewards*: suma de todas las recompensas que obtuvo el agente en cada paso del episodio.Más adelante se retoma este factor de descuento.
- *Value function*: esta función trata de estimar cuál será la recompensa final en este episodio empezando desde es estado actual.

La función de pérdida para PPO se define:

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2}$$

Donde el término:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \tag{3}$$

Es una razón de probabilidad entre las salidas de las políticas vieja y nueva. Será mayor a 1 si la acción es más verosímil ahora de lo que era en la versión vieja de la política.

PPO fuerza a las actualizaciones de la política a conservarse si se mueven muy lejos de la política actual.

IV. MODELO

AWS deep racer es un coche autónomo de carreras en escala 1/18 conducido con aprendizaje por refuerzo.

En este modelo:

- Agente: la red neuronal en el vehículo AWS DeepRacer
- Ambiente: la pista que va a correr el coche. Contiene los estados en los que se puede encontrar el robot.

- Estado: una imagen de la cámara frontal del coche. Representa el ambiente en un punto en el tiempo.
- Acción: movimiento del coche a cierta velocidad y ángulo de giro del volante *(steering)*.
- Recompensa: asignado por la *función de recompensa* es un puntaje que retroalimenta al agente después de actuar.

Los parámetros que caracterizan la función de recompensa para incentivar el comportamiento del coche son:

- Posición en la pista: variables *x,y* que describen las coordenadas en metros.
- Heading: orientación en grados respecto al eje x.
- Milestones: lista ordenada de puntos de medición a lo largo de la pista, cada uno corresponde a una coordenada *(x,y)*
- Ancho de la pista
- Distancia respecto a la línea del centro
- Llantas dentro de la pista
- Velocidad en m/s.
- Steering

V. ENTRENAMIENTO

El entrenamiento del modelo consiste en encontrar o aprender la función que maximiza la recompensa para que el modelo optimizado decida qué acciones (velocidad y ángulo de dirección) realizará el vehículo para completar el recorrido.

La función se representa con una red neuronal y el entrenamiento de la red conlleva a encontrr las ponderaciones de red óptimas dados los estados observados y las acciones del vehículo.

La integración de AWS DeepRacer con SageMaker permite que el marco de trabajo para aprendizaje por refuerzo sea TensorFlow.

En las siguientes subsecciones se explica con más detalle qué elementos deben ser configurados para entrenar el modelo.

A. Pista

Cada pista se clasifica por dificultad de acuerdo con la cantidad y tipo de curvas que tenga.

B. Espacio de acción

Contiene las acciones que el agente puede tomar tanto en el simulador como en el espacio físico. En un vehículo de la vida real, es un espacio continuo sin embargo, el AWS DeepRacer lo simplifica a un conjunto discreto de acciones.

El espacio de acción es configurable y las variables que pueden cambiar son el ángulo máximo de steering, la granularidad del steering, la velocidad máxima y la granularidad de la velocidad.

La granularidad se refiere a el número finito de posibles ángulos y velocidades en que se divide el rango de steering y velocidad.

C. Función de recompensa

A continuación se muestra la parte del código que define el comportamiento del coche.

```
# obtener los puntos de la pista mas
# cercanos a la posicion actual y
# generar vectores
theta = math.acos(producto/
        (abs(norma_next)*abs(norma_prev)))
theta = theta*(180/math.pi)

# la recompensa empieza con un valor
# muy bajo
reward = 1e-3

# marcas que delimitan posiciones
# a lo ancho de la pista
marca1 = 0.1 * track_width
marca2 = 0.25 * track_width
marca3 = 0.5 * track_width

# la primer condicion es que el coche
# se mantenga dentro de la pista
if all_wheels_on_track and
(0.5*track_width-distance_from_center)>=0.05:

    # la segunda condicion es que el coche
    # se mantenga cerca de la linea central
    if distance_from_center <= marca1:
        reward = 1
    elif distance_from_center <= marca2:
        reward = 0.5
    elif distance_from_center <= marca3:
        reward = 0.1
    else:
        reward = 1e-3

    # si hay un ángulo entre los puntos
    # previo, actual y futuro, tal vez
    # exista una curva
    if theta<=10 :

            # tolerancia de steering
            steering_tol = 10

        # si gira mucho el volante
        # se penaliza, esto para
        # evitar movimientos zig zag
        if (steering > steering_tol):
            reward *=0.8

    elif (theta > 15) and (speed>0.3):

        # viene curva y va rapido
        reward *= 0.6

# evitar que conduzca en reversa
else:
    reward = params['progress']
```

Con cada entrenamiento distinto del modelo, se realizaron ligeras modificaciones a la función de recompensa. Por ejemplo, la penalización por llevar una alta velocidad en zona de curvas.

D. Hiperparámetros

1) Gradient descent batch size: Es un muestra aleatoria de una cantidad de experiencias recientes tomada del *buffer* de experiencia. El muestreo se hace aleatorio para evitar las correlaciones inherentes a la información de entrada.

Normalmente se usar un tamaño de lote grande para promover actualizaciones más estables y "suaves" a los pesos de la red neuronal, sin embargo elegir un número demasiado grande puede alentar o acelerar el entrenamiento.

El lote se compone de un conjunto de imágenes capturadas por la cámara montada en el coche y las acciones ejecutadas.

2) Number of epochs: El número de veces que el algoritmo pasará a través de todo el conjunto de entrenamiento para actualizar los pesos de la red neuronal durante el descenso del gradiente.

Un mayor número de épocas promueve actualizaciones más estables pero corresponde a un entrenamiento más lento. Si el tamaño de lote es pequeño, se recomienda usar un menor número de épocas.

3) Learning rate: En cada actualización, una parte del nuevo peso puede provenir del descenso del gradiente y el resto de un valor existente de peso. La tasa de aprendizaje controla qué tanto contribuye un descenso del gradiente a los pesos de la red.

Usar una tasa de aprendizaje alta tiene mayores contribuciones del descenso del gradiente y provoca un entrenamiento más veloz, sin embargo, es posible que la recompensa no converja.

4) Entropy: El grado de incertidumbre que determina cuándo agregar aleatoriedad a la política de distribución. Esta aleatoriedad ayuda a que el vehículo explore el espacio de acciones de manera más amplia.

Un valor grande de entropía hace que el vehículo explore más el espacio de acción.

5) Discount factor: El factor de "descuento" determina qué cantidad de recompensas futuras se descuentan al calcular la recompensa en un estado dado como la recompensa promedio de todos los estados futuros.

Un factor de cero significa que el estado actual es independiente de pasos futuros mientras que, un factor de uno significa que las contribuciones de todos los pasos futuros están incluidas.

Por ejemplo, un factor de 0.9 en un tiempo específico tiene una recompensa esperada del orden del de diez pasos en el futuro.

6) Loss type: Es el tipo de función objetivo para actualizar los pesos de la red. El algoritmo de entrenamiento cambiará incrementalmente la estrategia del coche para que gradualmente transite de tomar acciones aleatorias a escoger acciones estratégicas que maximicen su recompensa.

Pueden surgir problemas como que, al realizar grandes cambios durante el entrenamiento, se vuelva inestable y el agente termine sin aprender nada.

Dos funciones de pérdida que se utilizaron fueron *Huber* y mínimos cuadrados *(Mean Squared)*. Ambas funciones actúan de manera similar para actualizaciones pequeñas pero cuando son más grandes, Huber tendrá menores incrementos comparado con Mean Squared. Esto vuelve más adecuada la función de Huber cuando se presentan problemas de convergencia mientras que Mean Squared es mejor para entrenar rápidamente.

7) Número de episodios entre cada iteración de la actualización de época: El tamaño del *buffer* de experiencia usado para almacenar la información de entrenamiento de la política de recompensa. Un episodio es un periodo en el cual el vehículo comienza de un punto dado y termina ya sea completando la pista o saliendo de ella.

Distintos episodios pueden tener diferentes duraciones.

E. Condición para detenerse

La condición que detiene el entrenamiento del modelo es definida por tiempo. Cumplido ese lapso de entrenamiento, el modelo entrega resultados y está listo para ser evaluado.

VI. RESULTADOS

La simulación de AWS DeepRacer se ejecuta en AWS RoboMaker y permite evaluar el desempeño del modelo, publicar su desempeño y compararlo con los modelos de otros usuarios de AWS Deep Racer. En esta sección se muestran los entrenamientos que más destacaron.

Para poder interpretar las gráficas, se consideran dos variables principales: la recompensa obtenida por episodio y el porcentaje de pista que se completó por episodio.

Las gráficas de puntos en color verde indican la recompensa obtenida por episodio y las gráficas con líneas en color verde son la media.

Las gráficas de puntos en color morado indican la el porcentaje de pista completado antes de salirse de ella por episodio y las gráficas con líneas en color morado son la media.

El resultado ideal es que las dos líneas que representan la media, converjan y la línea de porcentaje de pista completada llegue al 100.

A. Modelos que lograron entrar a la tabla de clasificación

Por las fechas en que se desarrolló este proyecto, fué posible participar en dos carreras diferentes. En la primer ocasión, se logró enviar el modelo a competir con otros modelos al rededor del mundo, las gráficas que muestran su desempeño se muestran en las figuras 4 a 9.

El segundo modelo que logró entrar a la tabla de clasificación, fue durante la misma competencia. Las gráficas que muestran el desempeño del modelo se muestran en las figuras 10 a 14.

Evaluation results

Trial	Time	Trial results (% track completed)
1	00:00:30.770	100%
2	00:00:31.387	100%
3	00:00:31.153	100%

Fig. 4. Evaluación en pista Oval Track

Evaluation results

Trial	Time	Trial results (% track completed)
1	00:00:13.660	49%
2	00:00:12.167	47%
3	00:00:25.617	100%

Fig. 5. Evaluación en pista re:Invent 2018

B. *Otros modelos*

A partir del modelo de las figuras 4-9, se siguió entrenando con algunas modificaciones a la función de recompensa, en otras ocasiones entrenando en una pista diferente o cambiando valores a los hiperparámetros.

En las figuras 15 a 20 se encuentran las gráficas que muestran el desempeño de la evolución del modelo.

En la función de recompensa del modelo 5.1, la condición de velocidad era demeritar la recompensa si superaba una velocidad mayor a 1 m/s. y a partir del modelo 5.1 la velocidad bajó a 0.3 m/s.

En el modelo 5.2 la tolerancia de steering cambió a 20 grados, se entrenó en la pista Bowtie y en la función de recompensa, se demeritó la condición de curva con velocidad alta en un 60%. Sin embargo el modelo se evaluó en la pista re:Invent 2018.

Los cambios que se realizaron en el modelo 5.3 fueron a la tasa de aprendizaje, que cambió de 0.001 a 0.003, la entropía cambió de 0.1 a 0.2 y el tipo de *loss function* cambió de *Mean squared error* a *Huber*.

Mientras que el modelo 5.4 regresó a sus valores del modelo 5.3 y anteriores en los hiperparámetros.

Finalmente, en las figuras 21 a 27 se encuentran los resultados de la evaluación correspondientes a las evoluciones del modelo.

VII. CONCLUSIONES

La planeación de movimiento es un problema muy complejo, a pesar de que hay mucha investigación al respecto y se han ofrecido diferentes vías de solución, ninguna es trivial y mucho menos económica en cuestión de recursos.

El aprendizaje por refuerzo es un método iterativo que de igual forma requiere de muchos recursos, pero ha demostrado funcionar de una forma deseable.

Sin duda alguna, de lo más complicado en un modelo de RL es diseñar la función de recompensa y entrenar el modelo.

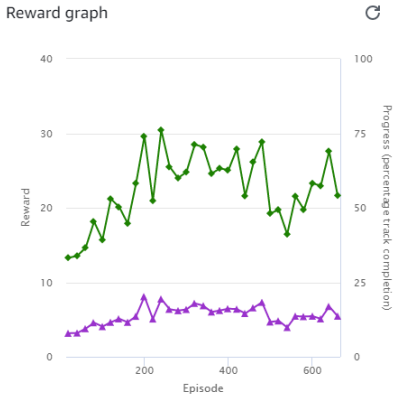

Fig. 6. Métricas

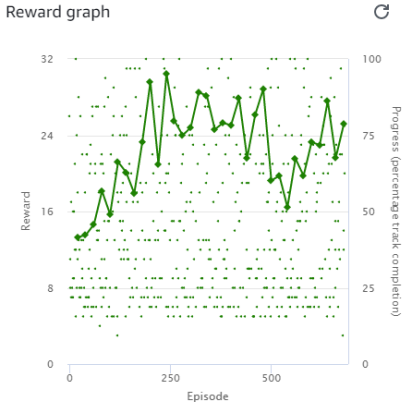

Fig. 7. Gráfica de recompensa

Una clave primordial para un buen entrenamiento es conocer los hiperparámetros involucrados y conocer qué valores deben tomar idóneamente.

Aún quedan muchos detalles que pulir en el campo tanto de robótica como de aprendizaje de máquina para poder llevar los coches autónomos a la vida cotidiana.

REFERENCES

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[2] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[3] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[4] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[5] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE, 1979, pp. 421–427.

[6] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

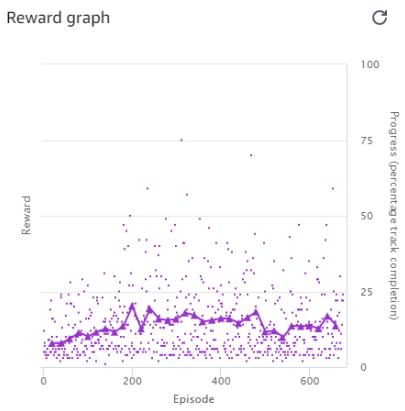

Fig. 8. Gráfica de porcentaje de pista completada

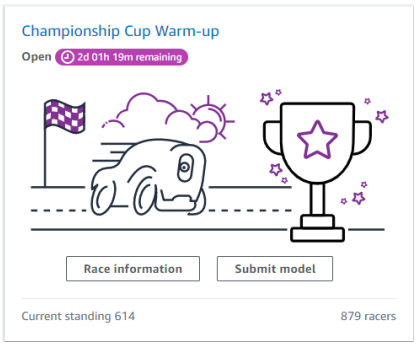

Fig. 9. Posición en tabla 28/11/19

Trial	Time	Trial results (% track completed)
1	00:00:22.940	100%
2	00:00:21.955	100%
3	00:00:18.882	84%

Fig. 10. Evaluación en pista re:Invent 2018

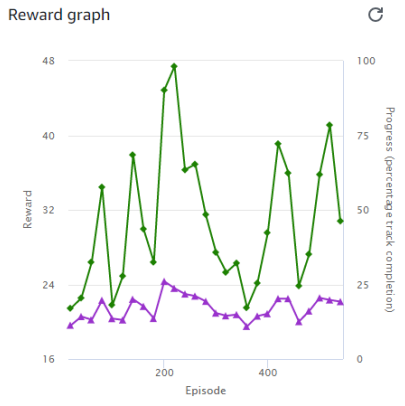

Fig. 11. Métricas

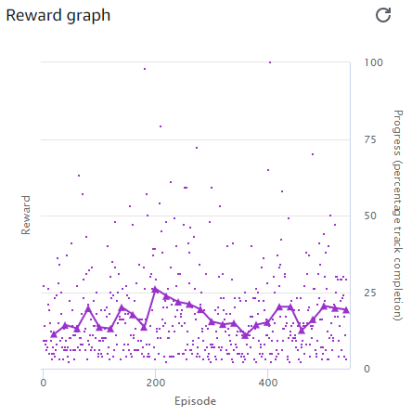

Fig. 12. Porcentaje de pista completada

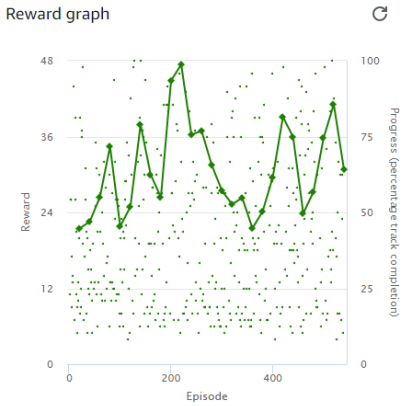

Fig. 13. Gráfica de recompensa

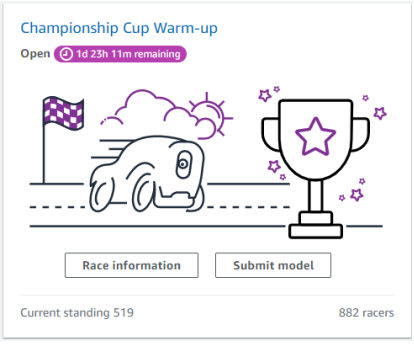

Fig. 14. Tabla de clasificación

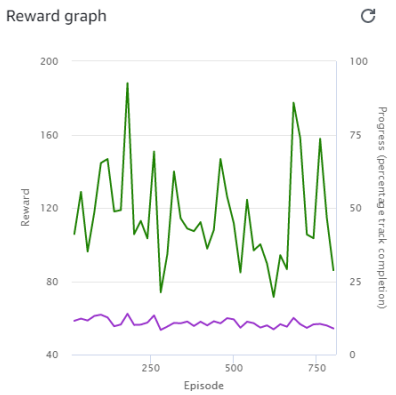

Fig. 15. Métricas modelo 5

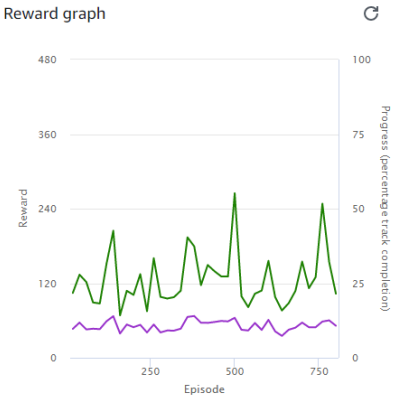

Fig. 16. Métricas modelo 5.1

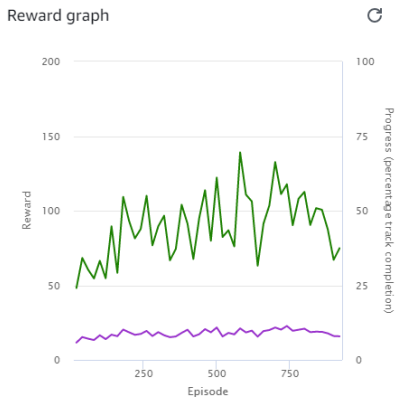

Fig. 17. Métricas modelo 5.2

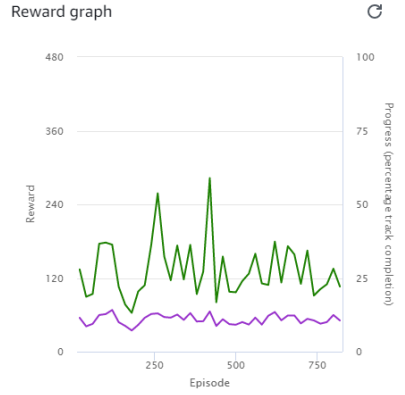

Fig. 18. Métricas modelo 5.3

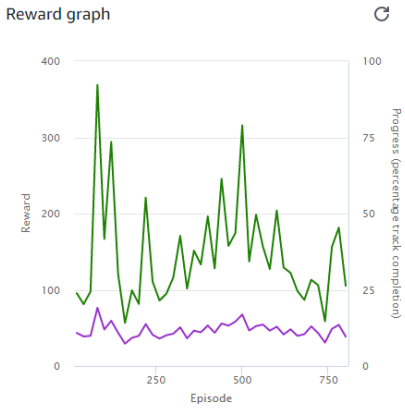

Fig. 19. Métricas modelo 5.4

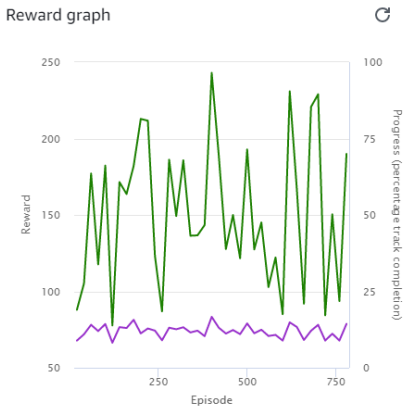

Fig. 20. Métricas modelo 5.5

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:03.926	18%	Off track
2	00:00:06.728	33%	Off track
3	00:00:07.316	37%	Off track

Fig. 21. Evaluación modelo 5

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:04.773	19%	Off track
2	00:00:03.166	14%	Off track
3	00:00:05.554	22%	Off track

Fig. 22. Evaluación modelo 5.1 pista Oval

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:07.339	32%	Off track
2	00:00:01.523	7%	Off track
3	00:00:08.923	47%	Off track

Fig. 23. Evaluación modelo 5.1 pista re:Invent 2018

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:04.732	25%	Off track
2	00:00:09.314	42%	Off track
3	00:00:03.939	21%	Off track

Fig. 24. Evaluación modelo 5.2

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:06.162	33%	Off track
2	00:00:04.745	20%	Off track
3	00:00:06.426	36%	Off track

Fig. 25. Evaluación modelo 5.3

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:18.232	80%	Off track
2	00:00:17.149	97%	Off track
3	00:00:04.078	23%	Off track

Fig. 26. Evaluación modelo 5.4

Evaluation results

Trial	Time	Trial results (% track completed)	Status
1	00:00:13.073	65%	Off track
2	00:00:04.588	19%	Off track
3	00:00:04.935	21%	Off track

Fig. 27. Evaluación modelo 5.5