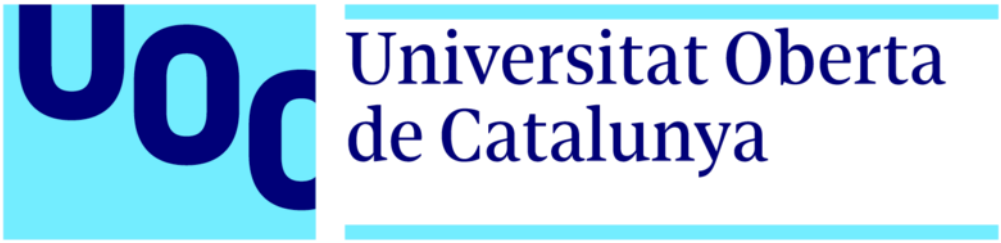

Máster universitario de Ciencia de Datos

Práctica Final – PRA1

Aprendizaje por refuerzo – Implementación de un agente para la guía autónoma.

Autor:

Mario Ubierna San Mamés

Índice de Contenido

Índice de Contenido 2
Índice de ilustraciones 3
1. Entorno 4
1.1. Ejercicio 1.1 (0.5 puntos) 4
1.2. Ejercicio 1.2 (0.5 puntos) 5
1.2.1. Espacios de observaciones 5
1.2.2. Espacios de acciones 8
2. Agente de referencia 10
2.1. Ejercicio 2.1 (1.5 puntos) 10
2.2. Ejercicio 2.2 (1 punto) 11
2.2.1. Configuración perdedora 11
2.2.2. Configuración ganadora 12
2.3. Ejercicio 2.3 (0.5 puntos) 13
3. Propuesta de mejora 14
3.1. Ejercicio 3.1 (2 puntos) 14
3.2. Ejercicio 3.2 (2 puntos) 15
3.2.1. Configuración perdedora 16
3.2.2. Configuración ganadora 16
3.3. Ejercicio 3.3 (2 puntos) 17
4. Bibliografía 18

Índice de ilustraciones

Ilustración 1 - Representación de una ejecución aleatoria. 5

Ilustración 2 - Espacio de observaciones KINEMATICS. 6

Ilustración 3 – Espacio de observaciones GRAYSCALE IMAGE. 7

Ilustración 4 - Espacio de observaciones OCCUPANCY GRID. 7

Ilustración 5 – Espacio de observaciones TIME TO COLLISION. 8

Ilustración 6 - Inputs de la red neuronal en el agente de referencia. 10

Ilustración 7 - Transformación de matriz a vector en el agente de referencia. 10

Ilustración 8 - Evolución de las recompensas y de las pérdidas configuración perdedora, agente de referencia. 12

Ilustración 9 - Evolución de las recompensas y de las pérdidas configuración ganadora, agente de referencia. 12

Ilustración 10 - Comportamiento del agente de referencia. 13

Ilustración 11 - Inputs de la red neuronal en el agente mejorado. 15

Ilustración 12 - Evolución de las recompensas y de las pérdidas configuración perdedora, agente mejorado. 16

Ilustración 13 - Evolución de las recompensas y de las pérdidas configuración ganadora, agente mejorado. 17

Ilustración 14 - Comportamiento del agente mejorado. 17

1. Entorno

Estamos trabajando sobre el problema de guía autónoma y en particular queremos solucionar el caso de conducción por carretera.

Para ello, se elige highway-env como entorno simplificado. El entorno se puede encontrar en el siguiente enlace: https://github.com/eleurent/highway-env.

En particular, nos centramos en el entorno highway, donde se representa una carretera con múltiples carriles y con un cierto número de coches transitando.

El objetivo de nuestro agente es de controlar un vehículo de modo que pueda:

- *evitar colisiones con los otros vehículos en carretera;*
- *tener la máxima velocidad posible;*
- *mantenerse en el carril derecho lo más posible.*

El entorno se considera superado cuando en el tiempo de observación máximo establecido, se hayan verificado las tres condiciones anteriormente introducidas.

El entorno también proporciona diferentes tipos de observaciones que se pueden ver en la documentación adjunta: https://highway-env.readthedocs.io/en/latest/observations/index.html#id1.

1.1. Ejercicio 1.1 (0.5 puntos)

Se pide explorar el entorno y representar una ejecución aleatoria.

Respecto a la exploración del entorno se ha encontrado la siguiente información:

- Por defecto el entorno hace uso del tipo de observación *Kinematics*, y el espacio de acciones es del tipo *DiscreteMetaAction* (todos los tipos de observación y de acciones serán explicados en el siguiente ejercicio).
- El número de carriles que tiene nuestra carretera son 4, pero se pueden añadir más o menos.
- El número de vehículos son 50, este valor también se puede modificar.

- La recompensa por colisión es -1.
- La recompensa por estar en el carril de la derecha es 0.1.
- La recompensa por ir a máxima velocidad es 0.4.
- La recompensa por cambiar de carril es 0.
- Y la recompensa por aumentar o disminuir la velocidad varía de forma lineal dependiendo de la velocidad que lleva el vehículo.

Respecto a la ejecución aleatoria:

Ilustración 1 - Representación de una ejecución aleatoria.

Si se requiere más información sobre la exploración del entorno o de la ejecución aleatoria se puede acceder al *notebook*.

1.2. Ejercicio 1.2 (0.5 puntos)

Explicar los posibles espacios de observaciones y de acciones.

Highway-env nos permite hacer uso de diferentes tipos de observaciones y de acciones para todos los entornos, es decir, que podemos configurar cómo son los estados y las acciones para cada entorno.

Cabe destacar que por defecto cada entorno tiene un tipo de observación y de acciones, pero éstos se pueden modificar.

1.2.1. Espacios de observaciones

Respecto a los diferentes tipos de observaciones nos encontramos con cuatro espacios [1]:

- *Kinematics.*

- *Grayscale Image.*
- *Occupancy grid.*
- *Time to collision.*

KINEMATICS

Este espacio de observación se caracteriza porque es una matriz del tipo V x F, dónde V es el número de vehículos cercanos que hay y F el conjunto de características que definen a cada vehículo.

Cabe señalar que el número de vehículos es una constante, y por defecto toma el valor 50.

Un ejemplo de este espacio de observaciones sería el siguiente:

Vehicle	x	y	v_x	v_y
ego-vehicle	5.0	4.0	15.0	0
vehicle 1	-10.0	4.0	12.0	0
vehicle 2	13.0	8.0	13.5	0
...	...	...	...	...
vehicle V	22.2	10.5	18.0	0.5

Ilustración 2 - Espacio de observaciones KINEMATICS.

Este es el espacio de observaciones más usado, y es el que vamos a utilizar para crear el agente de referencia.

GRAYSCALE IMAGE

La característica principal de este espacio de observaciones es que usa una imagen como tal para definir el estado, es decir, hay una matriz W x H donde W implica el ancho de la imagen y H la altura de la misma.

Un aspecto a tener en cuenta, es que la imagen está en una escala de grises y para transformar e interpretar dicha imagen a color se tiene que hacer una conversión de la misma, a partir de los pesos definidos en la configuración del espacio de observaciones.

Otra de las ventajas es que nos permite almacenar diferentes imágenes, por si se requieren a la hora del aprendizaje hacer uso de estados anteriores.

Un ejemplo del espacio de observaciones sería:

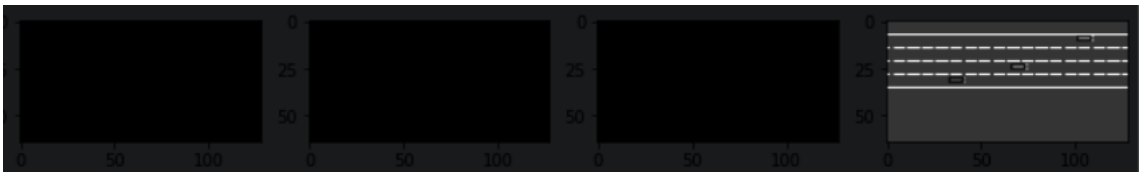

Ilustración 3 – Espacio de observaciones GRAYSCALE IMAGE.

OCCUPANCY GRID

Este espacio de observaciones "fusiona" los dos anteriores, es decir, tenemos una matriz de tamaño W x H, pero ahora estos valores no se corresponden con el tamaño de una imagen, sino que representan el espacio que hay alrededor de nuestro vehículo a partir de celdas rectangulares uniformes.

Otro componente de este espacio de observaciones es F, dando lugar a un array W x H x F, es decir, cada celda se describe a partir de sus características. Por ejemplo, para la característica *"presence*" podríamos tener la siguiente matriz:

0	0	0	0	0
0	0	1	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0

Ilustración 4 - Espacio de observaciones OCCUPANCY GRID.

Lo que se interpreta de la anterior imagen es que hay dos vehículos cercanos al nuestro, es decir, hay dos "1"; uno de ellos está un poco más al norte que nosotros y el otro está más al este.

TIME TO COLLISION

Es el último espacio de observaciones en el que la idea es completamente diferente a las demás, en este caso se busca calcular y representar el tiempo de colisión que hay entre nuestro vehículo y los vehículos que están en la misma carretera.

Para ello se hace uso de una matriz de matrices del tipo V x L x H, V son las diferentes velocidades a las que puede ir nuestro vehículo, por lo tanto, si hay tres velocidades vamos a tener tres matrices. L representa los carriles contiguos a nuestro carril actual y H representa el tiempo necesarios (pasos) para que se produzca una colisión.

Un ejemplo de este espacio de observaciones sería:

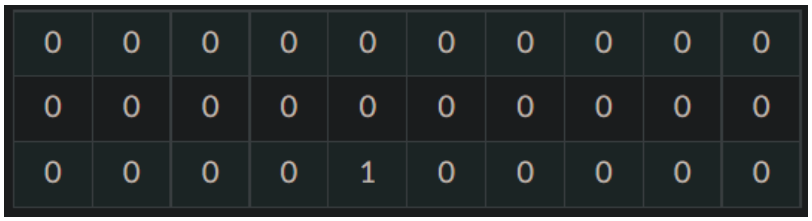

Ilustración 5 – Espacio de observaciones TIME TO COLLISION.

En la anterior imagen vemos que solo tenemos una matriz, por lo que V = 1. Dentro de la matriz hay tres filas o carriles (L = 3) y finalmente H es igual a 10, es decir, se predice la colisión en los próximos 10 pasos. Tal y como podemos ver en la imagen, en el carril de la derecha circula un vehículo con el que nos podemos chocar en 5 segundos (pasos).

1.2.2. Espacios de acciones

Respecto a los diferentes espacios de acciones tenemos [2]:

- *Continuous Actions.*
- *Discrete Actions.*
- *Discrete Meta-Actions.*
- *Manual control.*

CONTINUOUS ACTIONS

Se caracteriza porque el agente es capaz de controlar el vehículo pero a un bajo nivel, es decir, se puede controlar la aceleración y el ángulo de dirección necesarios para alcanzar el objetivo del problema pero accediendo directamente a los parámetros del vehículo.

DISCRETE ACTIONS

Este espacio de acciones es igual que el anterior pero hace una cuantificación uniforme, es decir, que la cuantificación es la misma para todos los niveles.

DISCRETE META-ACTIONS

En este caso, se añade una capa para que el control del vehículo por parte del agente sea más fácil. De esta forma se consigue que el agente sea capaz de llevar el vehículo a la velocidad que se desea.

Estas *meta-actions* se basan en definir los cambios de carril y la velocidad, y estos valores son usados por los controladores de bajo nivel.

Estas acciones son básicamente cinco:

- 0: se corresponde con cambiar al carril de la izquierda.
- 1: mantenernos en el mismo carril y a la misma velocidad.

- 2: cambiarnos al carril de la derecha.
- 3: aumentar la velocidad.
- 4: disminuir la velocidad.

Cabe destacar que este tipo de espacio de acciones va a ser el usado en la práctica.

En resumen, todos los espacios de acciones comentados se basan en lo mismo, pero éste nos permite controlar el vehículo a un alto nivel, es decir, no tenemos que tener conocimiento sobre los diferentes parámetros usados en el comportamiento del vehículo, esto es lo que diferencia este espacio de acciones respecto a los dos anteriores.

MANUAL CONTROL

En este caso el vehículo no aprende de forma automática a moverse, sino que somos nosotros los que con las flechas del teclado movemos el vehículo, es decir, nosotros nos convertimos en el agente del problema.

2. Agente de referencia

En la parte III de la asignatura hemos introducido el agente DQN con replay buffer y target network, que resulta ser un buen candidato para la solución del problema de conducción en carretera, visto que permite controlar entornos con un número elevado de estados y acciones de forma eficiente.

2.1. Ejercicio 2.1 (1.5 puntos)

Seleccionar la observación kinematics e implementar un agente DQN para el entorno highway. Se valorará implementación propia.

Para desarrollar este ejercicio nos hemos basado en la práctica anterior, para ello se han implementado tanto el *replay buffer*, la red neuronal y el agente que se desarrollaron en la anterior entrega.

En este punto hay que destacar dos cosas:

- Las entradas a la red neuronal van a ser 25, ya que tenemos una matriz de 5x5 y cada uno de estos elementos de la matriz definen el estado.

```
self.n_inputs = env.observation_space.shape[0] * env.observation_space.shape[1]
```

Ilustración 6 - Inputs de la red neuronal en el agente de referencia.

- Para hacer uso de la red neuronal de la práctica anterior se ha transformado el estado de una matriz de 5x5 a un vector de 25 para calcular los qvalores.

```
qvals = self.get_qvals(np.reshape(state, (self.n_inputs,)))
```

Ilustración 7 - Transformación de matriz a vector en el agente de referencia.

2.2. Ejercicio 2.2 (1 punto)

Entrenar el agente DQN y buscar los valores de los hiperparámetros que obtengan un alto rendimiento del agente. Para ello, es necesario listar los hiperparámetros bajo estudio y presentar las gráficas de las métricas que describen el aprendizaje.

Para realizar este ejercicio se ha ejecutado el agente sobre diferentes configuraciones de los hiperparámetros (para cada uno de los hiperparámetros), pero por motivo de limitar la extensión del documento se van a mostrar solamente dos configuraciones, siendo la última la usada para el entrenamiento del agente "óptimo".

	Configuración perdedora	**Configuración ganadora**
Learning rate	0.001	0.001
Batch size	**32**	**64**
Number of episodes	1000	1000
Burn in	1000	1000
DNN update	3	3
DDN Sync	100	100
Memory size	10000	10000
Gamma	0.8	0.8
Epsilon	1	1
Epsilon decay	0.99	0.99
Reward threshold	26	26

2.2.1. Configuración perdedora

Se muestran las gráficas tanto de la evolución de las recompensas como de la evolución de la pérdida:

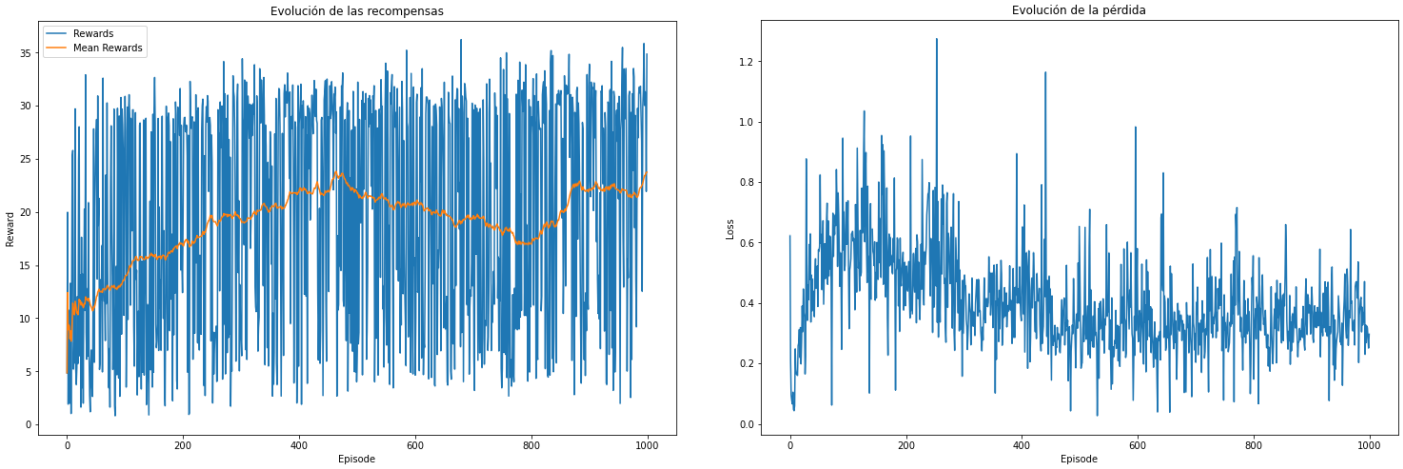

Ilustración 8 - Evolución de las recompensas y de las pérdidas configuración perdedora, agente de referencia.

2.2.2. Configuración ganadora

A continuación se muestran las gráficas respecto a la configuración ganadora:

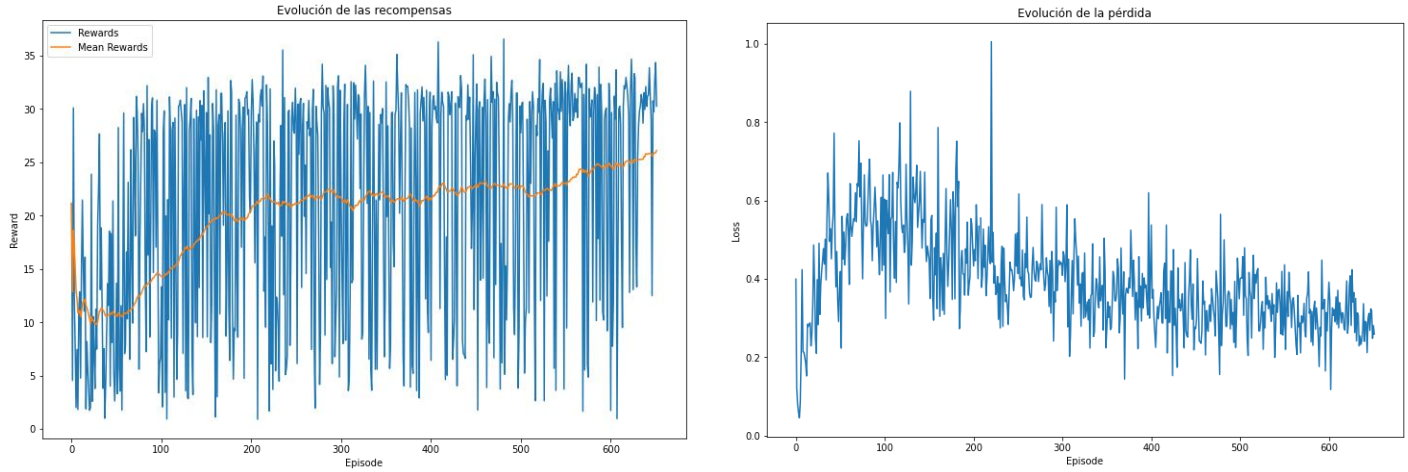

Ilustración 9 - Evolución de las recompensas y de las pérdidas configuración ganadora, agente de referencia.

Como podemos apreciar de las dos anteriores configuraciones, al aumentar al doble el tamaño del *batch* conseguimos que el agente aprenda de una forma más óptima, siendo "lineal ascendente" el aprendizaje. También podemos ver que la pérdida de las redes es menor que en el caso anterior y que el número de episodios para llegar al umbral ha variado, es decir, en la configuración perdedora se ejecutaron los mil episodios mientras que en la ganadora se necesitaron 652 episodios para llegar al umbral.

2.3. Ejercicio 2.3 (0.5 puntos)

Probar el agente entrenado en el entorno de prueba. Visualizar su comportamiento (a través de gráficas de las métricas más oportunas).

Una vez entrenado el agente se han ejecutado 100 episodios para ver el comportamiento del mismo, es decir, si en un entorno real funcionaría de forma correcta todas las veces o no.

El resultado obtenido es el siguiente:

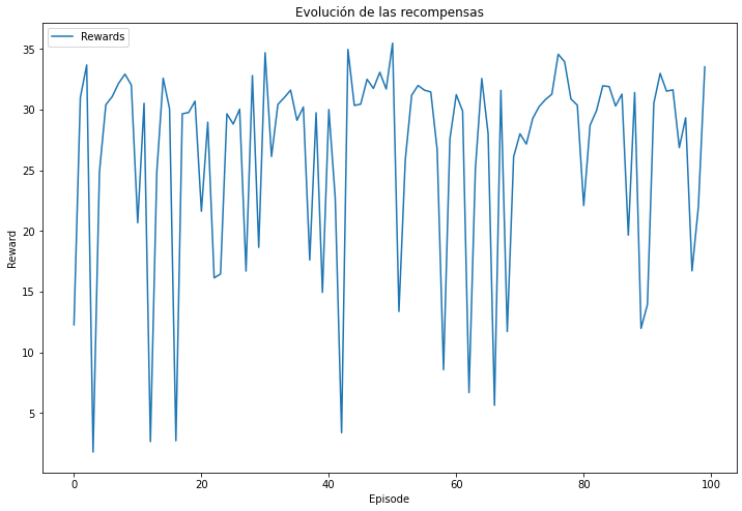

Ilustración 10 - Comportamiento del agente de referencia.

Para analizar el comportamiento del agente se ha usado el valor de las recompensas obtenidas durante 100 episodios. Un episodio satisfactorio sería a partir de 23 puntos en la recompensa, si analizamos la gráfica vemos que la gran mayoría de veces el agente es capaz de obtener una buena recompensa según los hiperparámetros que se han definido.

Observando los episodios en los que se obtiene una recompensa inferior a 23 puntos, vemos que el 15% de las veces el agente no es capaz de llegar a una solución correcta, sin embargo el 85% de las veces sí.

En resumen, el agente de referencia no es el mejor de todos ya que no conseguimos que sea capaz de llegar a la mejor solución todas las veces, pero aun así en la gran mayoría sí que consigue llegar a una buena solución.

3. Propuesta de mejora

En esta parte se pide proponer una solución alternativa al problema de conducción por carretera que pueda ser más eficiente con respecto a lo implementado anteriormente. Para alcanzar este objetivo, se debe usar un tipo de observación diferente de kinematics. Se deja la posibilidad de adaptar el agente DQN, implementado anteriormente, al nuevo espacio de observaciones o implementar un nuevo agente, basado en los algoritmos que hemos visto a lo largo de la asignatura.

3.1. Ejercicio 3.1 (2 puntos)

Implementar el agente identificado para el tipo de observación seleccionada en el entorno highway.

Justificar las razones que han llevado a probar este tipo de observación entre las disponibles y porque se ha elegido este tipo de agente. Detallar qué tipos de problemas se espera se puedan solucionar con respecto a la implementación anterior. Se valorará implementación propia.

Para realizar un agente mejorado se ha decidido hacer uso del tipo de observación *occupancy grid*, éste tal y como se mencionó en el primer ejercicio, el estado se caracteriza por ser una matriz de matrices. Las dos primeras dimensiones se corresponden con el espacio alrededor de nuestro vehículo y la tercera dimensión son las características, esta tercera dimensión es igual que la del espacio *kinematics*.

Se ha decido hacer uso de este tipo de observación básicamente por dos motivos:

- Para hacer uso del espacio *grayscale image* hay que tener conocimientos de cómo tratar las imágenes y ese no es mi caso todavía, espero cursar en el segundo semestre *Deep learning* y así aprender a hacer uso de imágenes y redes neuronales.
- Por lo tanto, la otra opción que quedaba era hacer uso del espacio *time to collision*, pero éste representa menos información que *occupancy grid*, por lo que al agente le va a costar más aprender en la dirección adecuada para resolver el problema de forma óptima.

Respecto al gente utilizado se ha decido mantener el mismo agente, es decir, un agente DQN. Esto es así básicamente para poder hacer una comparativa real en cómo afecta hacer uso de un tipo de espacio de observaciones frente a otro con los hiperparámetros ajustados para el mismo problema, si se usa un agente diferente dicha comparativa distorsiona los resultados a la hora de comparar el cómo afecta un tipo de estado y los hiperparámetros.

Como último punto de este apartado, destacar que los *inputs* de la red neuronal ahora cambian de 25 a 484 (ahora tenemos una matriz de 4x11x11), por lo tanto la red neuronal quedaría así:

```
self.n_inputs = env.observation_space.shape[0] *
                env.observation_space.shape[1] *
                env.observation_space.shape[2]
```

Ilustración 11 - Inputs de la red neuronal en el agente mejorado.

3.2. Ejercicio 3.2 (2 puntos)

Entrenar el agente identificado y buscar los valores de los hiperparámetros que obtengan el rendimiento "óptimo" del agente.

Al igual que sucedía con el agente de referencia, se han usado diferentes configuraciones, pero para limitar la extensión del documento se van a mostrar solo dos, siendo la última los hiperparámetros seleccionados.

	Configuración perdedora	**Configuración ganadora**
Learning rate	0.001	0.001
Batch size	**32**	**64**
Number of episodes	**1500**	**1000**
Burn in	1000	1000
DNN update	3	3
DDN Sync	500	500
Memory size	10000	10000

Gamma	0.8	0.8
Epsilon	1	1
Epsilon decay	0.99	0.99
Reward threshold	32	32

3.2.1. Configuración perdedora

Se muestran las gráficas tanto de la evolución de las recompensas como de la pérdida:

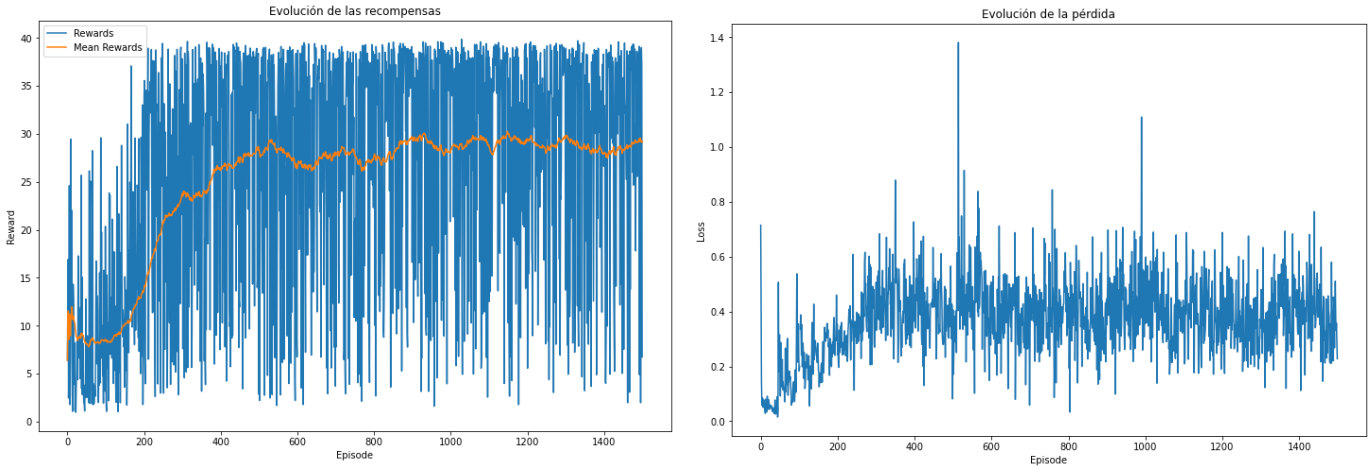

Ilustración 12 - Evolución de las recompensas y de las pérdidas configuración perdedora, agente mejorado.

3.2.2. Configuración ganadora

Las gráficas de la evolución de la recompensa y la pérdida son:

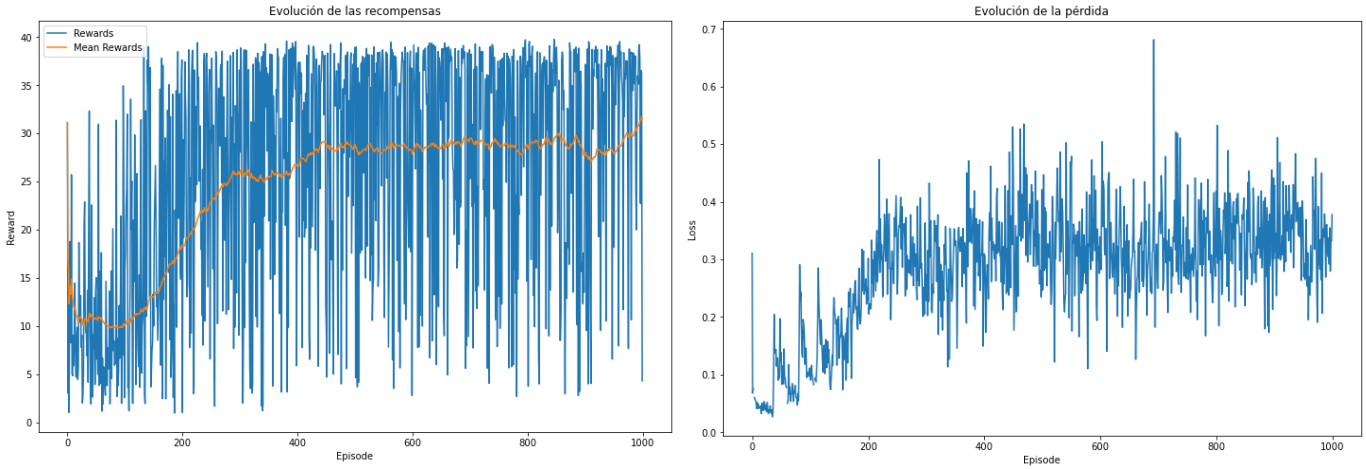

Ilustración 13 - Evolución de las recompensas y de las pérdidas configuración ganadora, agente mejorado.

Como podemos apreciar en las dos anteriores configuraciones, cuando aumentamos el tamaño del *batch* el agente es capaz de aprender mejor, llegando a unas mayores recompensas y a una menor pérdida en la red neuronal.

3.3. Ejercicio 3.3 (2 puntos)

Analizar el comportamiento del agente identificado entrenado en el entorno de prueba y compararlo con el agente implementado en el punto 2 (a través de gráficas de las métricas más oportunas).

Una vez entrenado el agente se han ejecutado 100 episodios para ver el comportamiento del mismo, el resultado obtenido es el siguiente:

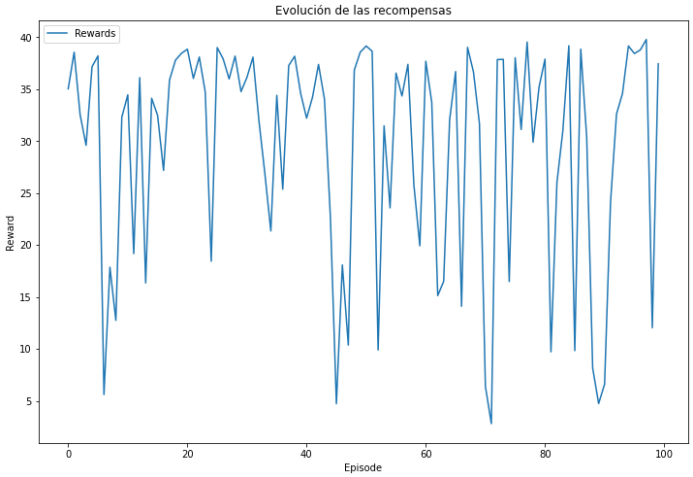

Ilustración 14 - Comportamiento del agente mejorado.

Observando los episodios en los que se obtiene una recompensa inferior a 23 puntos (la misma medida usada para el agente de referencia), vemos que en este caso los picos cuyos valores de recompensa son bajos son bastante similares entre el agente de referencia y el agente mejorado, es decir, más o menos el agente es capaz de llegar a una buena solución en el 85% de los casos.

La gran mejora entre este agente y el anterior, es que al hacer uso de el tipo de observación *occupancy grid* frente a *kinematics* conseguimos que el agente obtenga una mayor recompensa, es decir, si nos fijamos en los picos cuyos valores de recompensa son elevados vemos que éstos son mayores en el agente mejorado, de media se consigue llegar a una mejor solución 5 puntos superior que en el agente anterior, en otras palabras, cuando antes el agente llegaba a una recompensa de 35 ahora se consigue llegar a una de 40.

4. Bibliografía

[1] «Observations — highway-env documentation». https://highway-env.readthedocs.io/en/latest/observations/index.html (accedido dic. 25, 2021).

[2] «Actions — highway-env documentation». https://highway-env.readthedocs.io/en/latest/actions/index.html (accedido dic. 25, 2021).