Informe Técnico - Simulación NS3

Integrantes

Fabio Steven Tovar Ramos

Fecha de entrega

17 de junio del 2020

Contenido

1. Descripción del problema

Generar una red ad hoc de mínimo 20 nodos, con diferentes tipos de tráfico y servicios en NS3. También se debe utilizar la herramienta Open AI-Gym definiendo un ambiente multiagente con mínimo dos agentes que hagan uso de al menos dos métricas de aprendizaje para definir su comportamiento.
Comparar los resultados, y generar el informe técnico y la descripción del código.

2. Solución propuesta

Se propone una red ad hoc de 25 nodos conectados mediante wifi con un protocolo de enrutamiento Ad-hoc On-demand Distance Vector (AODV), en el cual hay comunicación de tipo TCP, UDP e ICMP entre los nodos. La evaluación del desempeño de los agentes en el ambiente se realiza con las métricas obtenidas mediante la operación de ping entre un par de nodos, se recolectan el RTT y la cantidad de paquetes enviados y recibidos.

La calidad de la red se ve degradada por la distancia en la cual se ven sometidos los nodos a medida que se van moviendo al pasar el tiempo, por lo tanto es necesario variar el poder de la transmisión (entre 0 y 50 decibeles, que es un rango de operación realista) para responder a estas necesidades. Por lo tanto, el objetivo de los agentes es variar esta variable de manera tal que se pueda asegurar una calidad muy alta optimizando el poder usado.

3. Diseño de la solución

Se diseñaron dos agentes para encargarse de esta tarea. El primero de ellos es un agente que operará intentando encontrar el valor óptimo del poder de la red para una configuración determinada, para ésto, se asumirá que los nodos no cambiarán su posición y por lo tanto tiene sentido realizar una búsqueda binaria de tal manera que sea posible encontrar el valor óptimo en múltiples intentos.
Sin embargo el comportamiento de este agente es subóptimo, puesto que es necesario que pierda algunos paquetes en sus intentos por encontrar un valor óptimo.

El otro es un agente inteligente, el cual aprenderá de los errores y aciertos del anterior agente para ajustar su comportamiento. Este agente funciona con un motor de aprendizaje de máquina el cual será entrenado antes de pasar al ambiente real, para lo cual es necesario recopilar los datos recolectados por el agente mencionado en el anterior párrafo.

Los datos recolectados fueron el tiempo de la simulación, la distancia entre el nodo más lejano y el centroide de la configuración de los nodos en un momento dado, la distancia entre los nodos emisor y receptor, la potencia óptima y la recompensa obtenida en ese tiempo.

Para utilizar un modelo de aprendizaje de máquina hay que determinar qué es lo que se desea aprender, por lo tanto se realizó en primer lugar un análisis de los datos obtenidos en varias simulaciones con distintos tiempos de ejecución y distintas semillas, para ver si tenían alguna posible relación. En la figura 1 se ve cómo se relacionan el radio, el poder obtenido y la distancia entre los nodos, donde son evidentes algunas cosas, como por ejemplo que hay valores de poder muy altos concentrados cuando el radio es grande.

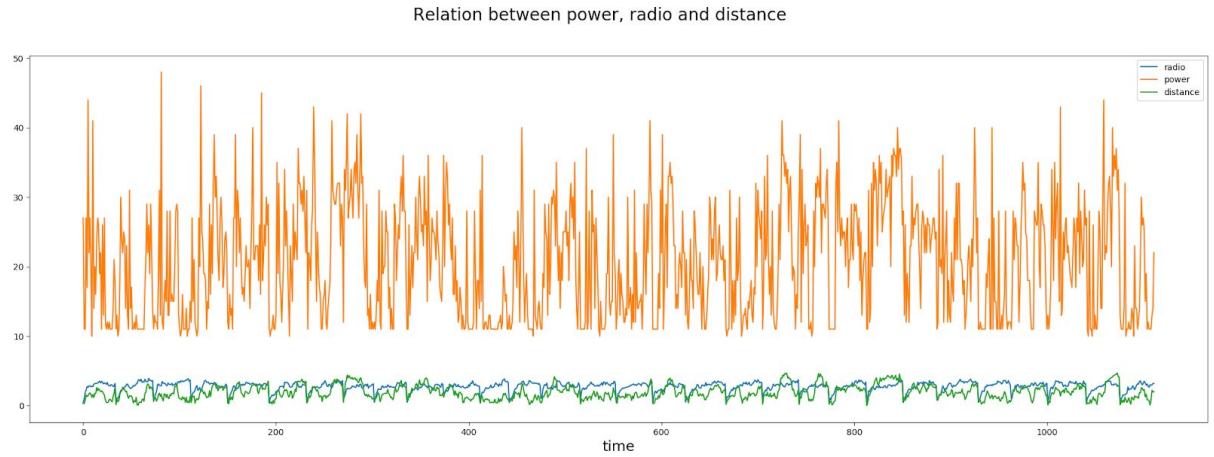

Fig. 1. Relación entre poder, radio y distancia

Sin embargo no es tan fácil de apreciar si realmente existe una relación entre las variables, por lo tanto se decidió realizar un análisis más detallado de la correlación entre los datos obtenidos. Para lo cual se realizó una matriz en la cual se graficó cada posible par de características, aquí es mucho más fácil de identificar la existencia de una correlación fuerte entre la potencia obtenida y la distancia del nodo emisor y el receptor, así como también su relación con el radio de la configuración. Esta matriz es mostrada en la figura 2.

Además muestra cosas interesantes, y a la vez problemáticas, como por ejemplo que el poder de transmisión se concentra en su mayoría entre los 10 y los 15, lo que podría causar un mal comportamiento en el momento en el cual se entrene el modelo debido a que este desbalanceo haría que posiblemente que se realicen más predicciones que caigan en este rango solamente porque hay más ejemplos.

Dados estos resultados, se determinó que la mejor opción sería seleccionar los datos en los cuales la recompensa hubiese sido positiva, y tomando como características el radio, distancia y tiempo entrenar la red neuronal de manera tal que intente ajustarse de la mejor manera a estos datos.

En un principio se trató el problema como un problema de clasificación, en el cual es necesario predecir la mejor clase dada una entrada, donde las clases serían los distintos niveles que es posible tomar en la red, es decir desde 0 dB a 50 dB. Sin embargo, este tratamiento es erróneo, puesto que los datos que se intentan predecir no son datos categóricos, por lo cual la red tenderá a tener un comportamiento bastante negativo especialmente para ciertas clases, y los errores que cometa serán bastante graves y afectarían de gran manera el rendimiento del agente.

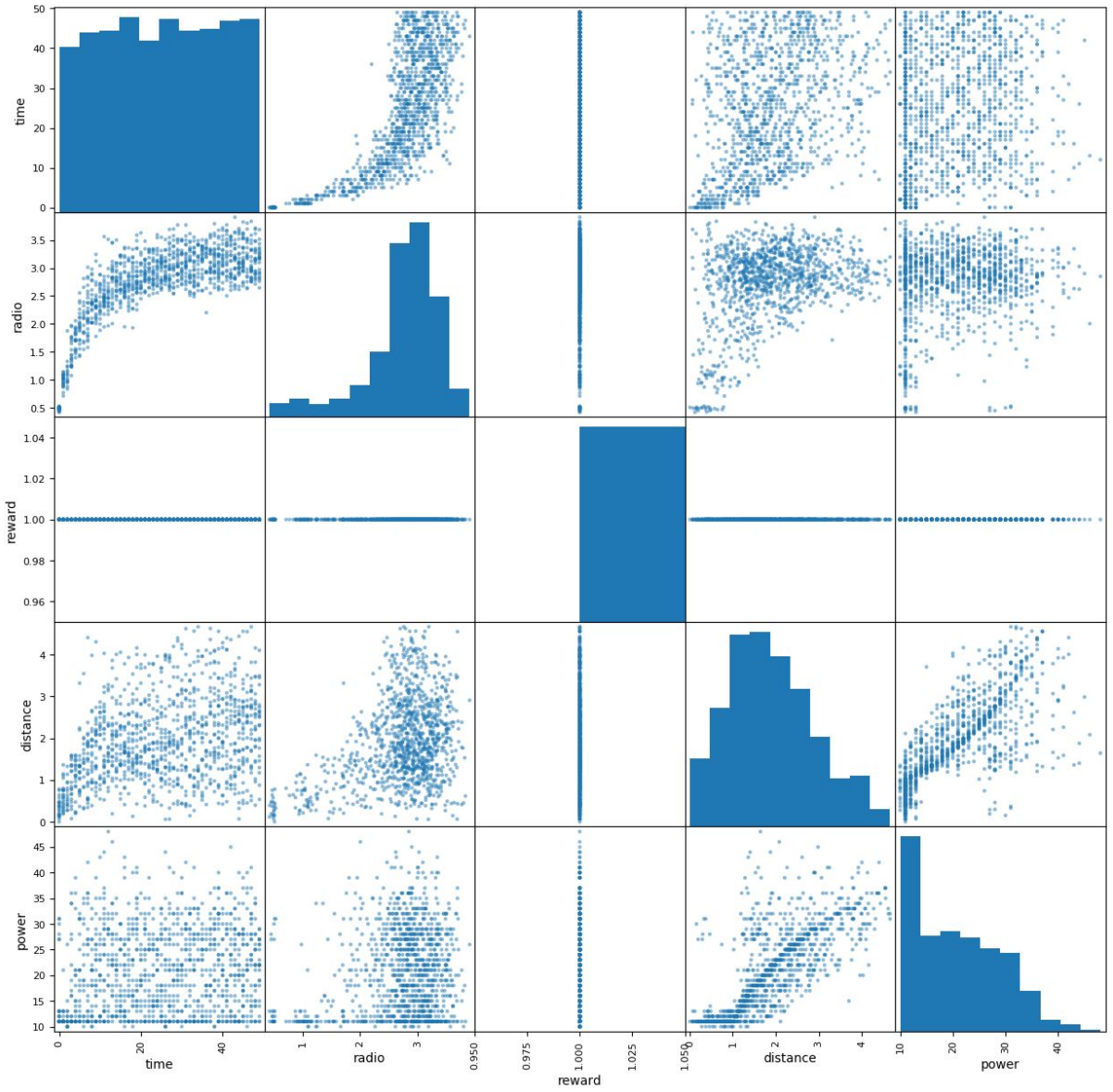

Fig. 2. Relación existente entre las características observadas en la simulación.

Por otro lado, tratar este problema como un problema de regresión discreta tiene mucho más sentido, puesto que los valores erróneos serán muy cercanos a los que deberían ser los óptimos, lo cual significa una disminución en la tasa de paquetes perdidos del agente.
Con la implementación de esta aproximación se obtiene un aprendizaje bastante bueno del modelo, puesto que la red se ajusta muy bien a los datos, los errores son muy pequeños y además no afectan en mayor medida la calidad de la red.

En la figura 3 se puede observar, luego de realizar el entrenamiento respectivo, cómo se comporta el modelo que gobierna el agente cuando se evalúa en datos desconocidos, lo cual sustenta la información anterior. Los puntos azules son los datos originales y los naranjas son los que fueron obtenidos con la red neuronal.

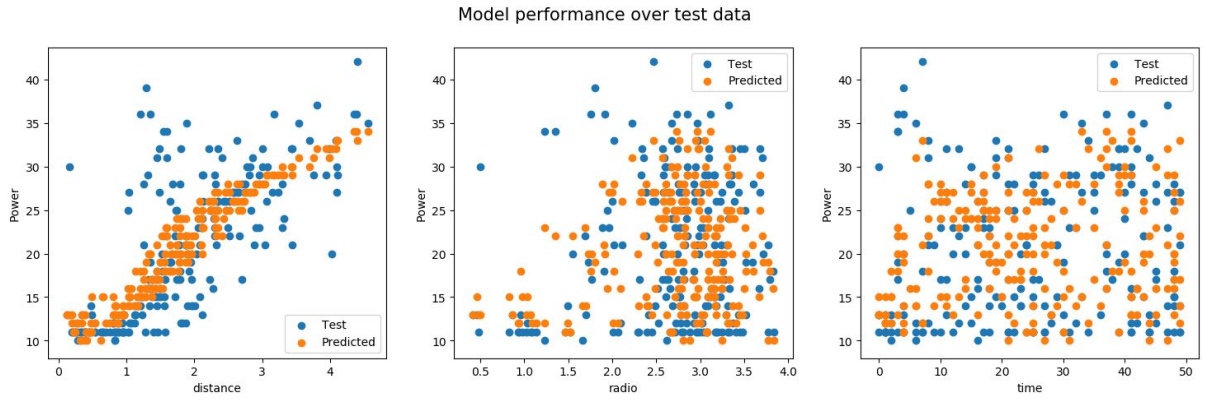

Fig. 3. Comportamiento de la red neuronal sobre datos desconocidos

Con esto es suficiente para resolver el problema en cuestión. El simulador finalmente cuenta con un modo de entrenamiento, en el cual correrá en principio el agente que realiza la búsqueda binaria la cantidad de episodios y pasos por cada episodio que se indique, posteriormente guardará los datos en un csv, con éstos datos entrenará la red neuronal y luego se prueba el rendimiento de ambos agentes sobre el ambiente real y se reúnen sus resultados para por último ser guardados en un csv.

También puede ejecutarse en modo normal, donde el agente inteligente cargará un modelo que debería ser entrenado previamente, y se compara su rendimiento con el otro agente de la misma manera que se expuso anteriormente.

4. Descripción técnica del simulador en NS3

Argumento	Descripción	Valor por defecto
nNodes	Número de nodos en la red	25
stepTime	Intervalo de tiempo entre cada paso (s)	1
txStart	Nivel de transmisión mínimo disponible (dBm)	0
txEnd	Nivel de transmisión máximo disponible (dBm)	50
txLevels	Niveles disponibles	51

Para los valores por defecto del simulador, en principio se crean 25 nodos los cuales son colocados en el ambiente de manera aleatoria siguiendo una distribución uniforme con centro en las coordenadas (2.5, 2.5), y con una distancia máxima a este punto de 0.5 en todas las direcciones.

Posteriormente se les da la habilidad de desplazarse a medida que el tiempo avance. Sin embargo, inician detenidos, puesto que los nodos se moverán durante un tiempo determinado cada cierto tiempo

para facilitar la tarea del agente binario de encontrar el valor óptimo para cada configuración. También se indica que los nodos no podrán moverse más allá de una región de 5x5 unidades.

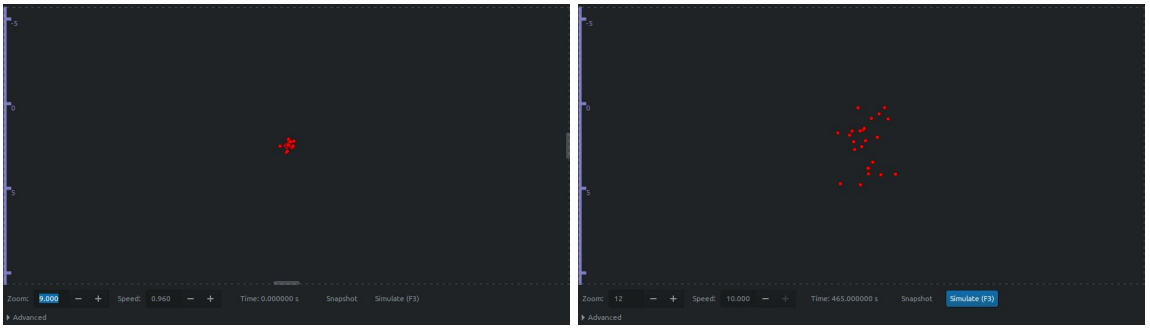

Fig. 4 Visualización de la simulación en el tiempo inicial y luego de haber transcurrido 460 segundos

En la figura 4 se puede apreciar el comportamiento del simulador dadas las restricciones anteriormente descritas. Este gráfico fue tomado como resultado de la ejecución del simulador apoyado con el gestor gráfico incorporado de NS3 (Visualizer).

Luego se instalan los módulos correspondientes al wifi en donde se especifican los niveles de transmisión obtenidos de los argumentos y un modelo de pérdida de la calidad de la señal variante respecto a la distancia de los nodos, el cual está determinado por la siguiente ecuación.

$$L = L_0 + 10nlog_{10}(\frac{d}{d_0})$$

Donde n es una constante de pérdida, d_0 la distancia de referencia, L_0 la pérdida de ruta inicial para una distancia de referencia, d es la distancia actual y L la pérdida de ruta total.

También se determina que el protocolo de enrutamiento para el simulador va a ser AODV, lo que permitirá que todos los nodos estén interconectados y se puedan generar rutas para guiar tráfico haciendo uso de múltiples nodos. En la figura 5 se puede evidenciar las complejas conexiones que se generan por parte de los nodos a medida que avanza el tiempo y los mismos cambian de posición.

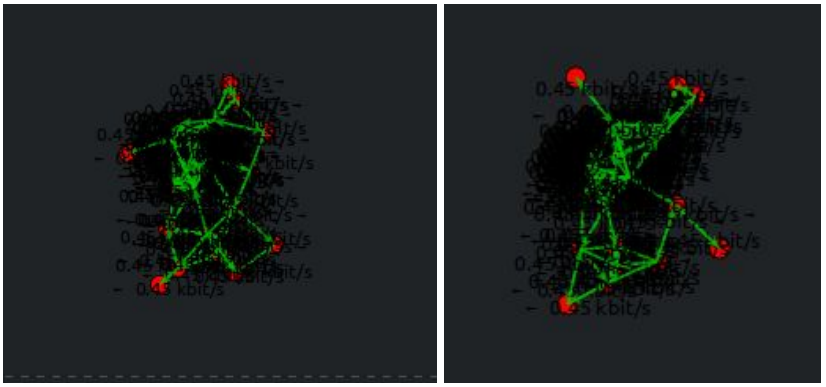

Fig 5. Ejemplo de dos configuraciones obtenidas de la simulación en dos tiempos distintos

Por último se instalan aplicaciones que generan tráfico TCP y UDP intermitentemente con intervalos de 10 segundos entre distintos nodos, y se instala una aplicación que permite realizar ping entre dos nodos, en este caso se seleccionó el primero y el último. El ping se realiza cada 5 segundos, con el

objetivo de asegurar que no queden paquetes pendientes que puedan entorpecer las lecturas de los agentes.

También hay que resaltar que los nodos se mueven durante 2.5 segundos cada 35 segundos aproximadamente. La razón de ésto es que el agente binario necesita realizar su búsqueda sin que los nodos se muevan, para lo que necesita en el peor de los casos 6 iteraciones, debido a la complejidad de la búsqueda binaria sobre los posibles 50 niveles ($\log_2 50 \approx 5.6438562$). Sin embargo, se decide arbitrariamente permitir una iteración extra, cada iteración toma 5 segundos, por tanto 35 segundos son adecuados para el correcto funcionamiento del agente.

Luego de que las iteraciones terminen, los nodos se moverán durante 2.5 segundos a una velocidad constante aleatoria distribuida uniformemente entre 0 y 0.15 metros por segundo y volverán a estar detenidas, así el agente en su próxima observación notará que el ambiente cambió y por tanto reiniciará su búsqueda.

Con respecto a la conexión NS3, en cada observación se envía una tupla formada por tres elementos, el RTT del ping acumulado en toda la simulación, la cantidad de paquetes recibidos correctamente y la posición de todos los nodos. El espacio de acción son los posibles niveles que se pueden tomar, es decir los números enteros entre 0 y 50. La recompensa es 1 si para el paso en el cual se realizan observaciones llegaron paquetes nuevos, o 0 si no. Y por último, la función que ejecuta la acción lo que hace es tomar el manejador del wifi de todos los nodos y colocarle el valor de la acción calculada por el agente.

5. Descripción técnica del ambiente de NS3Gym

Argumento	Descripción	Valor por defecto
start	Iniciar automáticamente la simulación de NS3 (1/0)	1
stepTime	Intervalo de tiempo entre cada paso (s)	5
simTime	Tiempo de simulación (s)	4000
training	Modo de entrenamiento	False
verbose	Modo verboso	False
episodes	Número de escenarios para los agentes	10
stepsByEpisode	Número de cambios de posición de los agentes por escenario	30

Como se había descrito anteriormente, el ambiente de NS3Gym cuenta con un modo de entrenamiento el cual permite recolectar los datos para entrenar el agente cognitivo. Sin embargo puede ejecutarse

omitiendo este modo puesto a que con la simulación están incluidos datos obtenidos previamente y un agente pre entrenado que será cargado si se inicia el ambiente en modo real.

El ambiente también cuenta con una serie de utilidades, como un helper que permite calcular información acerca de las posiciones de los nodos, y clases que manejan el comportamiento de los agentes para el ambiente generado por este simulador.

6. Descripción técnica de los agentes

Agente binario

Se llamará agente binario al agente construido de manera tal que intenta resolver el problema mediante una búsqueda binaria.

Internamente este agente maneja dos valores, un tope y un mínimo. Cada vez que se le solicita obtener una acción, él ajusta su tope y su mínimo como se hace tradicionalmente en una búsqueda binaria.

Para determinar cómo ajustarlos se usa la recompensa obtenida por el simulador y la acción que se ejecutó para obtener esa recompensa, de manera tal que si se obtiene una recompensa 1, el valor deseado será menor o igual a la acción que generó esa recompensa. Por otra parte, si no se obtiene recompensa, el valor deseado será mayor.

El agente también cuenta con una función que le permite reestablecer su mínimo y su tope, de tal manera que cada vez que las posiciones del simulador cambien, se deberá reestablecer su valor para iniciar una nueva búsqueda.

Agente cognitivo

Se llamará agente cognitivo al agente construido de manera tal que intenta resolver el problema mediante predicciones realizadas con un motor de deep learning.

El modelo que determina el comportamiento del agente consiste en una red neuronal de 5 capas (3 ocultas), que acepta entradas de tamaño 3 y su salida es de tamaño 1. Cada una de las capas tiene una función de activación *relu (rectified linear unit)* y tienen desde 1000 unidades en la primera capa hasta 100, disminuyéndolas a la mitad en cada capa. Puede verse esta arquitectura más claramente en la **figura.**

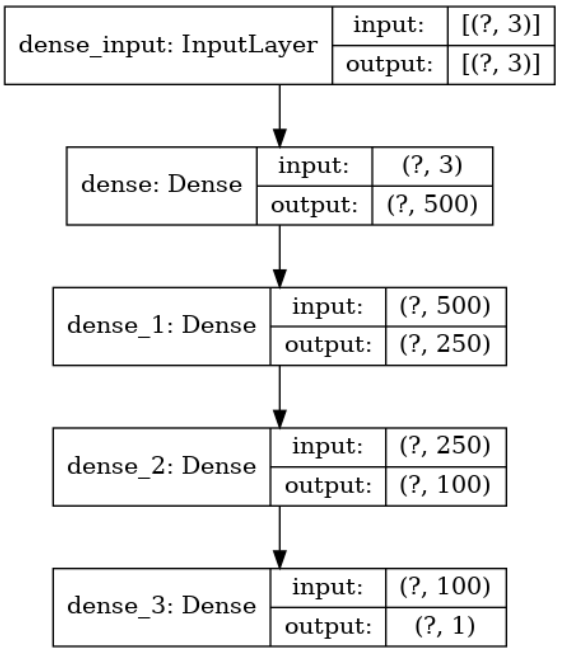

Fig. 6. Arquitectura de la red neuronal del agente

Para hacer funcional al agente es necesario cargarle unos pesos al anterior modelo, esto puede hacerse de dos maneras, ya sea entrenando al agente, o cargando pesos de un agente previamente entrenado. Existe una función que permite realizar cada una de las anteriores acciones. La función que permite entrenar al agente recibe un vector de observaciones y uno de objetivos, adicionalmente puede recibir datos para realizar validación del entrenamiento.

En el caso en el que no se den datos de validación, el agente pre-procesará los datos. Es decir, tomará las entradas y las normalizará, las reordenará aleatoriamente y las dividirá en dos subconjuntos, uno de entrenamiento y uno de pruebas, para posteriormente ser entrenado. En el caso de que los datos de validación sean proveídos, el agente asumirá que los datos ya han sido pre-procesados y solo entrenará al modelo.

Para obtener la predicción de la siguiente acción, el modelo tiene una función que dada una observación, la normaliza y predice el poder en el cual deberían estar los manejadores de wifi de cada uno de los nodos en la simulación.

7. Adicional

Adicionalmente se encontrarán dos *jupyter notebook* en los cuales se realizaron todos los análisis necesarios para determinar la mejor manera para construir el agente cognitivo, desde el análisis de los datos, hasta el entrenamiento y diferentes pruebas para mostrar su eficacia, así como también el análisis de los datos de los agentes sobre el ambiente real.