Robot Path planning using reinforcement learning and nonlinear approximation function (Planeación de trayetoria utilizando aprendizaje por reforzamiento y función de aproximación)

Miguel Angel Rodriguez Ruiz
Ciudad Universitaria
Universidad Autónoma de Ciudad Juárez
Ciudad Juárez Chihuahua México
Email: al115156@alumnos.uacj.mx

David Luviano Cruz
Universidad Autónoma de ciudad Juárez
Ciudad Juárez Chihuahua México
Email: david.luviano@uacj.mx

Luis Carlos Mendez Gonzalez
Universidad Autónoma de ciudad Juárez
Ciudad Juárez Chihuahua México
Email: luis.mendez@uacj.mx

Luis Pérez Domínguez
Universidad Autónoma de ciudad Juárez
Ciudad Juárez Chihuahua México
Email: luis.dominguez@uacj.mx

***Abstract*—In the present work we address the problem of dimensionality in reinforcement learning, this problem is called the curse of dimensionality, which is reflected in the increase of the state-action pairs to be learned, this happen when the number of actions and states in the environment increase, in this manner we propose to use a nonlinear approximation function (neural network) to offer an appriximate action to the agent when this is in a unknokn state in the task, this proposal will be validated by the planning of trajectories for a mobile robot in a unknown environtment using reinforcement learning.**

En el presente trabajo se aborda el problema de la dimensionalidad en el aprendizaje por reforzamiento, este problema se denomina la maldición de la dimensionalidad, lo que se refleja en el aumento de los pares estado-acción a aprender, lo que ocurre cuando el número de acciones y estados en el entorno aumenta , se propone utilizar una función de aproximación no lineal (red neuronal) para ofrecer una acción aproximada al agente cuando éste se encuentre en un estado desconocido en la tarea, esta propuesta será validada por la planificación de trayectorias para un móvil robot en un entorno desconocido utilizando el aprendizaje por reforzamiento.

I. Introducción

Diversos tipos de funciones de aproximación han sido propuestos en el campo de la teoría de control y del aprendizaje máquina, entre ellos, las redes neuronales ofrecen la particularidad de realizar aproximaciones de funciones no lineales, siempre y cuando se tenga acceso a una serie de datos de entrenamiento adecuados [1]. En lo que refiere tareas en que involucran sistemas no lineales o sistemas en donde una programación previa para resolver un problema no es posible, el aprendizaje por reforzamiento (RL) ha sido utilizado con éxito[2], ya que permite a los agentes aprender nuevos comportamientos de tal manera que puedan predecir y adaptarse a los cambios en el medio en donde se desarrollan [3].

El aprendizaje por reforzamiento es un método semi supervisado de aprendizaje, en donde el objetivo es maximizar una función de recompensa escalar la cual es definida por el entorno y las entidades que interactúan con el entorno llamados agentes [11]. En cada paso de aprendizaje, el agente toma una medición del entorno y toma una acción, lo que motiva que el entorno transite a un nuevo estado, esta transición es evaluada por medio de la función de recompensa escalar, es importante mencionar, que a los agentes no se les dice que acción tomar, por lo que deben de explorar el entorno para encontrar las acciones que le proporcionen una mayor recompensa[12].

La realimentación utilizado por el aprendizaje por reforzamiento en forma de función de recompensa es menos informativo que un método de aprendizaje supervisado tal como una red neuronal artificial, pero más informativo que un método de aprendizaje supervisado tal como clustering [15].

Distintos tipos de aplicaciones del aprendizaje por reforzamiento han tenido cabida en la teoría control, entre ellos, para manejar y controlar una red inteligente eléctrica[4],por medio de visión y RL enseñar a un robot a realizar disparos de una pelota a una meta [5],utilizar RL profundo para que un brazo manipulador aprenda movimientos en 3D [6], para coordinar actividades cooperativas entre agentes [7].

Un área en donde el aprendizaje por reforzamiento ha sido provechoso es la planeación de trayectoria para robots móviles,

en donde se genera una trayectoria desde un punto de inicio hasta un punto final respetando ciertas restricciones impuestas al desplazamiento, tales como obstáculos, delimitación del área de trayectoria. La generación de trayectoria puede ser realizada con distintos tipos de algoritmos de búsqueda, en [8] el método de grafos es utilizado para diseñar un espacio de tareas mediante búsqueda heurística, en [9] gráficos un direccionados son usados para la planeación de trayectoria, en [10] utilizan algoritmos genéticos para encontrar una trayectoria óptima.

A pesar del aparente éxito del aprendizaje por reforzamiento en la generación de trayectoria, los problemas abordados están limitados a estados discretos con un número finito de acciones disponibles para los agentes, debido a que se sufre de la llamada maldición de la dimensionalidad, la cual es el crecimiento exponencial de los pares estados-acciones a aprender conforme el número de estados y acciones se incrementan en el problema [13], lo que conlleva a un incremento en el tiempo de cómputo y de la cantidad de memoria necesaria para almacenar los datos asociados al algoritmo.

Por lo anterior, es necesario incorporar una estrategia adicional al aprendizaje por reforzamiento, la cual nos ofrezca la oportunidad de generalizar los resultados obtenidos con el objetivo de minimizar el tiempo de cómputo y memoria necesaria. En este artículo se tomaran las ventajas del aprendizaje de reforzamiento junto con una red neuronal multicapa, los cuales serán usados para la planeación de trayectorias en entornos dinámicos, el algoritmo estará integrado por dos etapas de aprendizaje, la primera se usará el algoritmo Q-learning [14], en donde el modelo de la tarea será conocido por medio de una función de transición de estados y de la función de recompensa escalar, en esta etapa el agente explorara el entorno de tarea con la finalidad de colectar información estados-acciones, es decir cuál es la acción optima a tomar cada uno de los estados explorados, en la segunda etapa, la información obtenida por el algoritmo de RL será usada para entrenar una red neuronal multicapa la cual nos brindara las acciones optimas a tomar por el agente (robot) en estados que no fueron explorados en la primera etapa de aprendizaje.

La propuesta sometida nos dará la oportunidad de obtener las acciones optimas a tomar cuando el agente se encuentre en estados que son desconocidos por este, debido a que no fueron explorados en la primera etapa o por que el entorno de trabajo cambio, lo que nos ofrecerá un grado de robustez.

II. Aprendizaje por reforzamiento

En esta sección se presenta el proceso de decisión de Markov y la caracterización de su solución óptima, los cuales son la base del aprendizaje por reforzamiento.

Definition 1: Un proceso finito de Markov para un agente es una tupla (S, A, f, ρ):

$$\begin{array}{l} f : S \times A \times S \to [0,1] \\ \rho : S \times A \times S \to \mathbb{R} \end{array} \tag{1}$$

donde S es el conjunto finito de estados en el entorno, A es el conjunto finito de acciones disponibles para el agente, f es la función de probabilidad de transición de estados y ρ es la función de recompensa la cual se asume acotada [3].

El estado actual del entorno de trabajo el cual el agente observa es definido como $s_t \in S$, el cual describe al estado en cada paso de tiempo discreto t ,el agente observa el estado y toma una acción definida como a_t. Como resultado el entorno cambia su estado a algún $s_{t+1} \in S$ de acuerdo a la función de probabilidad de transición de estados f , la probabilidad de acabar en el estado s_{t+1} después que la acción a_t es ejecutada en el estado s_t es $f(s_t, a_t, s_{t+1})$.

El agente recibe una recompensa escalar $r_{t+1} \in \mathbb{R}$, de acuerdo a la función de recompensa $\rho : r_{t+1} = \rho(s_t, a_t, s_{t+1})$. Esta recompensa evalúa el efecto inmediato de la acción a_t , es decir, la transición desde el estado s_t al estado s_{t+1} .Esta recompensa refleja que tan productiva fue la acción tomada a_t. Sin embargo esta señal no dice nada acerca de los efectos a largo plazo de esta acción tomada.

Para sistemas deterministas, la función de probabilidad de transición de estados f y la función de recompensa ρ toman la forma:

$$\begin{array}{rcl} \bar{f} & = & S \times A \to S \\ \bar{\rho} & = & S \times A \to \mathbb{R} \end{array}$$

Donde la recompensa es completamente determinada por el estado actual y la acción actual $r_{t+1} = \rho(s_t, a_t)$. Algunos procesos de decisión de Markov tienen estados terminales los cuales son estados donde una vez alcanzados no pueden ser abandonados en el futuro, se toma que todas las recompensas recibidas en un estado terminal se toman como cero. En tal caso, el proceso de aprendizaje es usualmente separado en distintos episodios, los cuales son trayectorias comenzando desde algún estado inicial y finalizando en un estado terminal [16].

El comportamiento del agente es descrito por su política π, la cual especifica como el agente escoge sus acciones en un estado determinado del medio. La política π puede ser determinista :

$$\bar{\pi} = S \to A$$

o estocástica:

$$\pi = S \times A \to [0,1]$$

Una política es llamada estacionaria si esta no cambia en el tiempo. El objetivo final del aprendizaje por reforzamiento es encontrar una política π, para todo estado s que maximice el retorno R :

$$R^{\pi} = E\left\{ \sum_{t=0}^{\infty} \gamma^k r_{t+1} \middle| s_0 = s, \pi \right\} \tag{2}$$

donde $\gamma \in [0,1)$ es el factor de descuento, la expresión anterior es tomada sobre la probabilidad de transición de

estados sobre la política π, debemos notar que R representa la recompensa acumulada por el agente en el largo plazo.

Existen otras formas de definir el retorno R en función de la actividad realizada [17]. El factor de descuento γ puede ser considerado como codificación de la incertidumbre creciente acerca de la recompensa que sera obtenida en el futuro o como un medio para acotar la suma en (2) de otro modo crecería de manera no acotada.

Por lo tanto el objetivo del agente es maximizar su rendimiento a largo plazo caracterizado por el retorno R, solo recibiendo la realimentación acerca de su inmediata actuación en forma de la señal de recompensa r. Una forma de obtener el anterior resultado es por medio del cálculo de una función óptima estado-acción de valor llamada función Q (Q-function)

$$Q^h : S \times A \rightarrow \mathbb{R} \tag{3}$$

la cual da un retorno R esperado dada una política π comenzando desde cualquier par estado-acción:

$$Q^{\pi}(s,a) = E\left\{ \sum_{t=0}^{\infty} \gamma^k r_{\,t+1} \middle| s_0 = s, a_0 = a, \pi \right\} \tag{4}$$

La función Q óptima es definida como Q^*

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{5}$$

La cual satisface la ecuación de optimalidad de Bellman:

$$Q^*(s,a) = \sum_{\acute{s} \in S} f(s,a) \left[\rho\left(s,a,s^{'}\right) + \gamma \max_{a^{'}} Q^*\left(s^{'},a^{'}\right) \right] \tag{6}$$

La secuencia Q_t converge a Q^* bajo las siguientes condiciones [3]:

- Distintos valores de la función Q son guardados y actualizado para cada par estado-acción.
- La sumatoria $\sum_{t=0}^{\infty} \alpha$ es finita.
- Asintóticamente todas los pares stado-acción son visitadas de manera infinita.

III. Aprendizaje por reforzamiento con red neuronal

En esta sección se describe la estrategia propuesta en este artículo, la cual consta de dos etapas de aprendizaje, la primera se usara un algoritmo de aprendizaje de reforzamiento y en la segunda etapa se entrenara una red neuronal con los datos obtenidos en la primera.

A. Aprendizaje por reforzamiento

En esta etapa se usara el algoritmo de aprendizaje Q-learning , el cual asegura la convergencia hacia los valores Q óptimos, la cual se reflejara en una política de acciones π optimas, lo anterior por medio de garantizar que el algoritmo Q-learning es una contracción [14].

En el algoritmo se Q-learning la experiencia del agente consiste en una secuencia de episodios en donde el agente:

- observa su estado actual s_t
- Selecciona y realiza una acción a_t
- Observa el estado siguiente s_{t+1}
- Recibe la recompensa r_t
- Ajusta los valores Q_t usando un factor de aprendizaje α de acuerdo a :

$$\begin{aligned} Q_t(s_t,a_t) &= (1-\alpha) Q_{t-1}(s_t,a_t) + ... \\ &\quad +\alpha \left[r_t + \gamma \max_{a'} Q_{t-1}(s_{t+1},a') \right] \end{aligned} \tag{7}$$

donde $\alpha \in (0,1)$ es el factor de aprendizaje. La secuencia Q_t converge a Q^* converge bajo ciertas condiciones, incluyendo que el agente se mantenga intentado realizar todas las acciones en todos los estados, lo que implica que el agente en ocasiones debe de explorar y en otras realizar lo que dicta la política de acciones encontrada [18]. La exploración que realizaremos será escogiendo una acción aleatoria con probabilidad $\epsilon \in (0,1)$ y escogiendo una acción codiciosa (que ofrezca una mayor recompensa) con una probabilidad $(1-\epsilon)$.

Bajo las condiciones anteriores mencionadas, el algoritmo converge en un numero finito de iteraciones, lo que indica que la fase de aprendizaje ha sido terminada y se ha obtenido las acciones optimas a ejecutar por el agente en cada uno de los estados visitados.

B. Fase de aprendizaje mediante red neuronal

Las redes neuronales tienen una habilidad poderosa de fusión de datos y tolerancia a fallas. Ya que las redes neuronales de una sola capa solo resuelven los problemas de clasificación lineales, una red neuronal con múltiples capas es utilizada para estimar las acciones mejor valuadas que serán usadas por los agentes en los estados que no fueron visitados durante la fase de aprendizaje por reforzamiento. El flujo de aprendizaje es mostrado en la Figura 1.

Las acciones disponibles para los agentes en los estados no visitados durante la fase de aprendizaje por reforzamiento serán aproximados por la siguiente red neuronal compuesta por 1 capa oculta:

$$\hat{a}_{t+1} = W\sigma(V[s_t]) \tag{8}$$

donde $W \in R^{n \times m}$, $V \in R^{m \times n}$, m es el número del nodo oculto el cual es diseñado por el usuario, n es el número de agentes que se encuentran en el entorno en este caso $n = 1$,

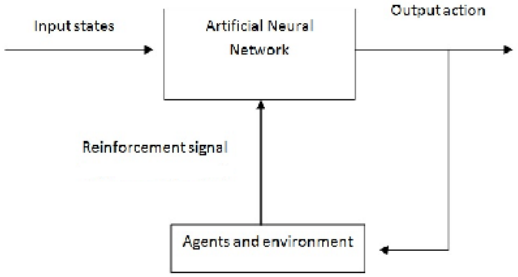

Fig. 1. Flujo de aprendizaje para la red neuronal

V representa los pesos en la capa oculta, σ es una función de activación neuronal, se usa una función sigmoidea en esta propuesta.

Las entradas para la red neuronal son los estado s_t y la salida de la red serán las acciones aproximadas $\hat{a}_t$ que el agente deberá de tomar para llevar a cabo la meta encomendada. Es de notar, que el aprendizaje utilizado en las redes neuronales es un método de aprendizaje supervisado. Por lo que las muestras para el entrenamiento de la red neuronal provendrán de la fase anterior de aprendizaje, el cual nos brindará como entradas de entrenamiento los estados del sistema y como salida las acciones óptimas encontradas para cada uno de los agentes.

El error entrenamiento $e(k)$ a la salida de la neurona j está definido como

$$e_j(k) = y_j(k) - d_j(k)$$

donde $y_j(k)$ es el estado medido por el agente, y $d_j(k)$ es el patrón de acciones de los agentes. Las sumas instantáneas de los errores al cuadrado de la salida es dado por

$$\varepsilon(k) = \frac{1}{2}\sum_{j=1}^{l} e_j^2(k)$$

donde l es el número de neuronas de la capa de salida. Usando el algoritmo backpropagation para entrenar la red neuronal, el peso que conecta la neurona i con la neurona j es actualizado de la siguiente manera:

$$w_{ij}(k+1) = w_{ij}(k) - \eta \frac{\partial \varepsilon(k)}{\partial w_{ij}(k)}$$

donde η es el parámetro de razón de aprendizaje del algoritmo backpropagation. El termino $\frac{\partial \varepsilon(k)}{\partial w_{ij}(k)}$ puede ser calculado:

$$\frac{\partial \varepsilon(k)}{\partial w_{ij}(k)} = \frac{\partial \varepsilon(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{ij}(k)}$$

Las derivadas parciales están dadas por

$$\begin{aligned} \frac{\partial \varepsilon(k)}{\partial e_j(k)} &= e_j(k), \frac{\partial e_j(k)}{\partial y_j(k)} = 1, \\ \frac{\partial y_j(k)}{\partial v_j(k)} &= \varphi_j'\left[v_j(k)\right], \frac{\partial v_j(k)}{\partial w_{ji}(k)} = y_i(k) \end{aligned}$$

Una función lineal es utilizada en la capa de salida de la red neuronal

$$w_{ij}(k+1) = w_{ij}(k) - \eta y_i(k) e_j(k) \tag{9}$$

De tal manera que w_{ki} puede ser actualizado como:

$$w_{ki}(k+1) = w_{ki}(k) - \eta \frac{\partial \varepsilon(k)}{\partial w_{ki}(k)}$$

además

$$\begin{aligned} &\frac{\partial \varepsilon(k)}{\partial w_{ki}(k)} = \frac{\partial \varepsilon(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial y_i(k)} \frac{\partial y_i(k)}{\partial v_i(k)} \frac{\partial v_i(k)}{\partial w_{ki}(k)} \\ &= \left\{ e_j(k) \varphi_j'\left[v_j(k)\right] W_{ij} \right\} \varphi_i'\left[v_i(k)\right] y_k(k) \\ &= e_i(k) \varphi_i'\left[v_i(k)\right] y_k(k) \end{aligned}$$

Por lo tanto

$$w_{ki}(k+1) = w_{ki}(k) - \eta y_k(k) e_i(k) \varphi_i'\left[v_i(k)\right] \tag{10}$$

donde $e_i(k) \varphi_i'\left[v_i(k)\right]$ es el error en backpropagation [15].

IV. Planeación de trayectoria

Con la finalidad de validar la propuesta realizada, se presenta una tarea de planeación de trayectoria para un robot móvil, en la cual el agente o robot deberá de encontrar el camino optimo (el que le ofrezca una mayor cantidad de recompensa numérica), evitando obstáculos presentes en el entorno y evitando salir del espacio físico del medio de trabajo. La única información disponible que tendrá el agente será la función de recompensa y la función de transición de estados determinista.

En cada paso de aprendizaje, los estados actuales (la posición en forma de coordenadas del agente en el plano XY) será observada y referida a una tabla estado-acción. En el caso que el agente se encuentre en un estado no disponible en la tabla, la acción que deberá ejecutar el agente será proveída por la estimación realizada por la red neuronal.

A. Resultados de simulación

La simulación será realizado en un entorno de trabajo discretizado, el cual tendrá un tamaño en el plano de 5×5 en el plano XY, $x \in [1,5]$ $y \in [1,5]$ por lo que el estado será representado por coordenadas cartesianas que represente la posición del robot, el vector de estado $s \in [x,y]$. Cada robot tendrá 4 acciones disponibles que podrá realizar: moverse hacia arriba, abajo, izquierda y derecha, $A \in$ [izquierda,derecha,arriba,abajo] .

La posición inicial del robot es mostrada en la figura 2, la función de recompensa ρ está dada:

$$\begin{aligned} r &= -5 \text{ si sale del entorno o choca} \\ r &= 10 \text{ si llega a la meta} \\ r &= 0 \text{ cualquier otra situación} \end{aligned}$$

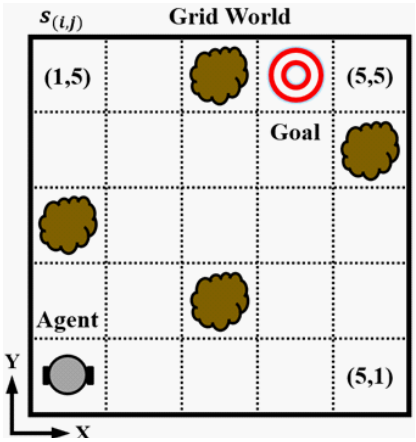

Fig. 2. Tarea de navegacion propuesta

TABLE I
PARÁMETROS UTILIZADOS EN EL ALGORITMO Q-LEARNING

Parametro de exploracion ε	0.7
Razon de aprendizaje α	0.1
Factor de descuento γ	0.98

La tabla con los valores estados-acción tendrá un tamaño de 100 entradas, $|S| = 5 \times 5$, $|A| = 4$.Los valores obtenidos al finalizar la primera etapa de aprendizaje utilizando el algoritmo Q-learning son mostrados en la figura 3, despues de 29 iteraciones se logra la convergencia, los parámetros usados se muestran en la tabla (I). En la figura (4), se muestran las trayectorias optimas obtenidas mediante Q-learning, todas estas trayectorias ofrecen la misma recompensa total al seguirlas de 98.38.

Para la segunda etapa de aprendizaje, los valores Q estado-acción óptimos encontrados en la primera fase, son usados para entrenar una red neuronal de una capa oculta, en donde las entradas a la red serán los estados y la salida será las acciones. La codificación de las acciones para su uso en la red neuronal son 1-izquierda, 2-derecha, 3-arriba, 4-abajo. El algoritmo utilizado para entrenar la red neuronal es Backpropagation, las especificaciones de la red neuronal se muestran en la tabla (II).

La figura (5) muestra la acción propuesta por la red neuronal en color verde, cuando el robot se encuentra en un estado que no se encuentra en la tabla de valores estados acciones (3), la red neuronal ofrece una acción aproximada a realizar, esta acción tiene como objetivo acercarnos a un estado conocido por el agente en color azul, lo que producirá que se integre a una de las trayectorias optimas encontradas en la primera fase de aprendizaje

V. CONCLUSIONES

La propuesta presentada ofrece un método hibrido entre una técnica de aprendizaje semi-supervisado (aprendizaje por

	Recompensas dependiendo la acción			
Posición	Left	Right	Up	Down
Q11	0.49027057	7.130905021	7.130905021	0.49027057
Q12	3.300203452	9.968480681	2.130905488	5.490270104
Q13	0.49027057	0.49027057	0.49027057	0.49027057
Q14	7.292940763	13.77215438	10.68214364	7.292940763
Q15	5.947573542	12.31807983	5.682144102	12.2929403
Q21	5.490270104	9.968480681	9.968480681	3.300203452
Q22	8.300202985	5.559819669	12.08929957	8.300202985
Q23	7.292940763	13.77215438	13.77215438	10.5598192
Q24	12.2929403	15.53603974	12.31807983	12.2929403
Q25	10.94757308	7.318080298	7.318080298	13.90550713
Q31	8.300202985	12.08929957	4.968481147	5.559819669
Q32	0.49027057	0.49027057	0.49027057	0.49027057
Q33	12.2929403	15.53603974	15.53603974	8.905507598
Q34	13.90550713	18.05877027	10.5360402	13.90550713
Q35	0.49027057	0.49027057	0.49027057	0.49027057
Q41	10.5598192	10.73914234	13.77215438	7.292940763
Q42	8.905507598	12.34110216	15.53603974	12.2929403
Q43	13.90550713	13.91672277	18.05877027	13.90550713
Q44	15.97521583	14.0360657	21.82664072	15.97521583
Q45	15.49026917	15.49026917	15.49026917	15.49026917
Q51	12.2929403	5.981495688	12.34110216	5.981495688
Q52	13.90550713	7.459822216	13.91672277	10.98149522
Q53	15.97521583	8.916723241	8.916723241	12.45982175
Q54	0.49027057	0.49027057	0.49027057	0.49027057
Q55	23.14384188	14.10045021	14.10045021	14.10045021

Fig. 3. Valores estado-accion

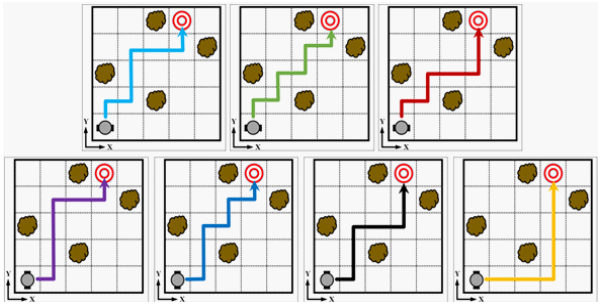

Fig. 4. Trayectorias optimas encontradas con aprendizaje por reforzamiento

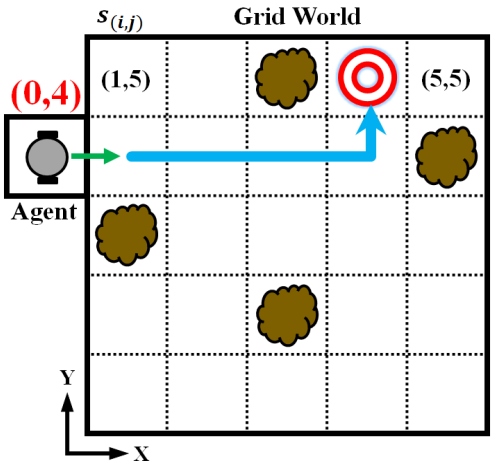

Fig. 5. Acción aproximada ofrecida por la red neuronal

TABLE II
PARAMETROS UTILIZADOS EN LA RED NEURONAL ARTIFICIAL

Numero de capas ocultas	1
Numero de neuronas de la 1^{ra} capa oculta	10
Funcion de activacion en la capa de salida	Funcion Lineal
Funcion de activacion en capa oculta	Funcion Sigmoide

reforzamiento) y una técnica de aprendizaje supervisado (red neuronal), lo que permite tener un grado de robustez cuando las condiciones del entorno de trabajo cambian, es de hacer mención, que las acciones propuestas por la red neuronal son sub-optimas, en el sentido que en general no ofrecen la trayectoria más corta hacia una trayectoria optima conocida, esto debido a errores de aproximación, como trabajo a futuro queda generalizar la propuesta en espacios no discretos, así como implementar la técnica de manera experimental en robot móviles diferenciales.

Una de las principales ventajas de esta técnica es que se evita el volver a ejecutar el algoritmo Q-learning cuando las condiciones del entorno cambian, cuando el problema tiene muchos estados y acciones disponibles, la convergencia es lenta.

REFERENCES

[1] Sofge D, White D (1990) Neural network based process optimizationand control. In Proceedings of the 29th IEEE Conference on Decision and Control. Hawai, USA

[2] Kaelbling L. P., Reinforcement learning: A survey, Journal of Artificial Intelligence Research, 4(3), 237-285,1996.

[3] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

[4] Kara, E. C., Berges, M., Krogh, B., & Kar, S. (2012, November). Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework. In Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on (pp. 85-90). IEEE.

[5] Asada, M., Noda, S., Tawaratsumida, S., & Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine learning, 23(2), 279-303.

[6] Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017, May). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 3389-3396). IEEE.

[7] Foerster, J., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Advances in Neural Information Processing Systems (pp. 2137-2145).

[8] Bhattacharya, S., Likhachev, M., & Kumar, V. (2010, May). Multi-agent path planning with multiple tasks and distance constraints. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 953-959). IEEE.

[9] Wang, K. H. C., & Botea, A. (2011). MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. Journal of Artificial Intelligence Research, 42, 55-90.

[10] Cai, Z., & Peng, Z. (2002). Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. Journal of Intelligent & Robotic Systems, 33(1), 61-71.

[11] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the eleventh international conference on machine learning (Vol. 157, pp. 157-163).

[12] Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008.

[13] Foerster, J., Nardelli, N., Farquhar, G., Torr, P., Kohli, P., & Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. arXiv preprint arXiv:1702.08887.

[14] Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[15] Haykin, S. (1994). Neural networks: a comprehensive foundation. Prentice Hall PTR.

[16] Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1), 99-134.

[17] Wei, A., Hu, X., & Wang, Y. (2013). Consensus of linear multi-agent systems subject to actuator saturation. International Journal of Control, Automation and Systems, 11(4), 649-656.

[18] Busoniu, L., Babuska, R., & De Schutter, B. (2006, December). Multi-agent reinforcement learning: A survey. In Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on (pp. 1-6). IEEE.