Implementación de Técnicas de Aprendizaje por Refuerzo

Cristina García[1], Gabriel Zimmermann[1], Jaquelina Alegre[1]

{crissa777@, grabrielemanuel, jacquelinaalegre@}gmail.com

[1] Alumno de UTN - Facultad Regional Resistencia, French 414,
3500 Resistencia, Chaco Argentina

Resumen
Este informe presenta un análisis de la técnica de aprendizaje por refuerzo implentando el algoritmo de Q-learning realizando una comparación entre las distintas políticas de selección de acción (ε-greedy, softmax y aleatorio).El dominio de aplicación es un grid de 6x6 a 10x10. El objetivo es llegar a la meta aproximandose a la política óptima en la menor cantidad de movimientos posibles y maximizando la recompensa.

Palabras claves: Aprendizaje por refuerzo, Q-learning, agente, entorno, política/estrategias, función de refuerzo, función de evaluación, episodio.

1 Introducción

La técnica de aprendizaje por refuerzo a diferencia de otras técnicas de aprendizaje parte de un enfoque totalmente distinto que se basa en aprender de la experiencia es decir a través de "Prueba y error"[1].
Es un aprendizaje orientado al objetivo, dado el caso de estudio se define como objetivo llegar a la meta con la máxima recompensa.
En este modelo los agentes aprenden comportamientos por medio de interacciones de ensayo y error dentro de un medio dinámico, cuentan con una descripción del estado actual y deben seleccionar la siguiente acción de un conjunto de posibles acciones, es decir el agente implementa un mapeo de estados a probabilidades de selección de las diferentes acciones disponibles para cada estado en un determinado tiempo, cuyo resultado debe maximizar el refuerzo escalar proporcionado después de realizar la acción elegida.
Utilizaremos el algoritmo de Q-learning que es un modelo de diferencia temporal sin política, lo interesante de aplicar el algoritmo con los diferentes métodos de aprendizaje por refuerzo es que especifican como el agente cambia su política como resultado de su experiencia, es decir cómo aprende, la meta del agente es maximizar la cantidad total de refuerzo a lo largo del tiempo.

En este trabajo se describe la implementación de la solución de un grid en el que un agente desde un estado inicial debe llegar a la meta (estado final), pasando por diferentes estados que maximicen la recompensa obtenida.
En la primera parte, se describe el marco teórico del presente informe. En la segunda, se presenta la descripción de la solución implementada explicando los problemas encontrados, para finalizar con las simulaciones realizadas y resultados obtenidos y por último las conclusiones obtenidas a partir de ellos.

2 Marco teórico

El desarrollo de la investigación toma como punto de partida los fundamentos teóricos referentes a la técnica aprendizaje por refuerzo y el algoritmo de Q-learning, a continuación se realiza la descripción de los conceptos teóricos que permiten aplicar dicho método y representar una solución factible para el dominio del problema.

2.1 Aprendizaje por Refuerzo

El aprendizaje por refuerzo consiste en aprender qué acciones realizar, dado el estado actual del ambiente, con el objetivo de maximizar una señal de recompensa numérica, lo cual requiere de un mapeo de situaciones a acciones [2] .

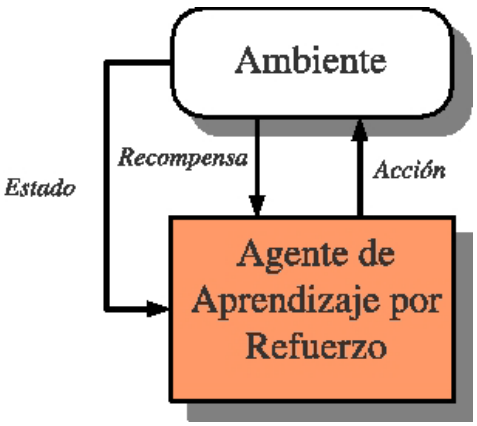

Fig. 1. Arquitectura básica de un sistema de aprendizaje por refuerzo.

El aprendizaje por refuerzo es aprender que hacer, es decir, aprender cómo mapear situaciones a acciones, de manera de maximizar un valor numérico denominado recompensa. Se debe aprender qué acciones llevan a la mayor recompensa mediante prueba y error. En la mayoría de los casos, no sólo es importante la recompensa inmediata sino también las recompensas subsecuentes. El aprendizaje por refuerzo no se define por caracterizar métodos deter-

minados de aprendizaje, sino más bien por caracterizar el problema de aprendizaje en sí mismo. Un aspecto a considerar en aprendizaje por refuerzo es cuanto explorar y cuanto explotar. Para obtener recompensas, un agente debe preferir acciones que probó en el pasado y le dieron buenos resultados. Pero para descubrir dichas acciones tiene que probar acciones que no ha seleccionado previamente. El agente debe explotar lo que ya conoce para obtener recompensas, pero también debe explorar para elegir mejores acciones en el futuro.
En el aprendizaje por refuerzo la única información de retroalimentación es un escalar conocido como recompensa. Dicha recompensa se recibe posteriormente a la ejecución de una acción por parte del agente. Una característica importante de este tipo de aprendizaje es que al final lo que se busca es una función que nos indique cual es la mejor secuencia de acciones necesarias para alcanzar una meta predefinida.

2.2 Características del aprendizaje por refuerzo[2]:

- **Está dirigido por objetivos.** Este objetivo se expresa por el logro de la máxima recompensa recibida a largo plazo que devuelve el entorno al realizar una acción sobre él, si bien esta no se conoce, es la que resulta más adecuada para el sistema.
- **El comportamiento del entorno es, en general, desconocido y puede ser estocástico**, es decir, que la evolución del entorno y la recompensa generada pueden obedecer a una cierta función de probabilidad.
- **La recompensa puede tener un cierto retardo,** significa que, la de recompensa haber realizado una acción puede reflejarse hasta un cierto número de evaluaciones posteriores.
- **Dado que el comportamiento del entorno es desconocido**, el aprendizaje por refuerzo conlleva una fuerte carga de prueba y error.
- **Un problema a resolver es lograr el balance óptimo entre la exploración-explotación.**
  Después de realizar la evaluación debería permitir encontrar la política óptima para el dominio del problema, eligiendo si es mejor explorar el entorno para mejorar el conocimiento del problema (a costa de empeorar a corto plazo la recompensa obtenida) o explotar el conocimiento acumulado (intentando maximizar la recompensa).

2.3 Elementos del aprendizaje por refuerzo[1]

Más allá del agente y del ambiente, se pueden identificar 4 subelementos de un sistema de aprendizaje por refuerzo: una política, una función de recompensa, una función de valor, y un modelo del ambiente.

- **Agente:** Es el sujeto cuyo funcionamiento consiste en interactuar con el entorno realizando acciones que le permiten conocer los estados por los que puede transitar y obteniendo recompensas después de ejecutar dichas acciones.
- **Entorno:** Es el objeto sobre el que opera el agente, presenta generalmente un comportamiento desconocido y estocástico es responsable de generar las recompensas asociadas a las acciones y cambios de estado con lo cual puede evolucionar.

- **Política:** una política define el comportamiento de una agente aprendiz en un momento dado. En otras palabras, una política es un mapeo de un conjunto de estados S percibidos del ambiente a un conjunto de acciones A que deben ejecutarse en dichos estados. En algunos casos la política puede ser una función simple o una tabla de entradas, y en otros casos puede ser un proceso complejo de búsqueda. La política es lo más importante de un agente que aprende por refuerzo, en el sentido de que es suficiente para determinar su comportamiento.
  En general, las políticas pueden ser estocásticas.

$$\pi : S \rightarrow A \qquad (1)$$

- **Función de refuerzo:** genera la recompensa en función del estado del entorno y la acción realizada sobre él, teniendo como objetivo maximizar la recompensa total obtenida a largo plazo.

$$R : S \rightarrow R, o\ bien\ R : S \times A \rightarrow R \qquad (2)$$

- **Función de evaluación (función de valor):** Esta función se utiliza para escoger la acción a realizar la que conduzca al estado con mayor valor, realizando una estimación de la recompensa que se va a recibir, partiendo de un cierto estado y siguiendo una cierta política.

$$F : S \rightarrow R\ o\ bien\ F : S \times A \rightarrow R \qquad (3)$$

- **Modelo del entorno:** permite predecir el comportamiento del entorno y aprovechar esta información para resolver el problema.[3]

Algoritmo Q-Learning para entornos deterministas

1) Inicializar (con 0's o valores aleatorios) $Q(s,a), \forall\, s \in S, a \in A$
2) Repetir (por cada episodio)
3) Inicializar s
4) Repetir (por cada paso del episodio):
   a) Elegir a desde s usando una política derivada desde Q
   b) Ejecutar acción a , observar estado resultante s' y reward r.
   c) $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma\, max_{a'} Qs \leftarrow s'; Q(s',a') - Q(s,a)]$
   d) $s \leftarrow s';$
5) hasta que s es terminal

[Algoritmo Q-Learning, obtenido de la bibliografía [1].]

Políticas

El agente necesita una política para poder para seleccionar las acciones a ejecutar cuando está en un determinado estado s, esta define el comportamiento del agente :

Política aleatoria

Consiste en escoger siempre una acción aleatoriamente.

Política ε-greedy

La técnica de ε -greedy es una variación de la exploración greedy [4]. El agente identifica la mejor acción de acuerdo a la función ε –greedy (s,a). La constante ε es un valor entre 0 y 1 que determina la probabilidad con la que se elegirá una acción que no corresponda a la que provee el mayor refuerzo.
La función de exploración para un estado s y una acción a se define de la siguiente manera:

- Consiste en asumir una probabilidad (1- ε) de escoger la acción con mayor estimación, y una probabilidad ε de escoger aleatoriamente entre todas las acciones.

- Esta estrategia aumenta la exploración en el comportamiento de las acciones, lo que permite que las estimaciones de la recompensa promedio sean mejores y a largo plazo se consiga mejor recompensa.

Donde elegir Acción(s) es una distribución uniforme sobre todas las posibles acciones a tomar a partir de un estado dado.
Explora más que una política greedy.

Política soft-max
Se basa en escoger con mayor probabilidad a las acciones con una estimación más alta.
El parámetro τ es una variable de control llamado temperatura que determina el grado de exploración.
• Si $\tau \rightarrow \infty$, todas las acciones son equiprobables.
• Si $\tau \rightarrow 0$, la política soft-max se convierte en la política greedy.
Generalmente se reduce τ con el tiempo, para garantizar la convergencia.
No siempre es fácil definir τ , porque depende del orden de magnitud de Q(s,a).
Utiliza la función de distribución de Boltzmann

$$\Pi(a) = \frac{e^{\frac{Q(a)}{\tau}}}{\Sigma e^{\frac{Q(a)}{\tau}}} \qquad (4)$$

Procesos de decisión Markov
Los procesos de decisión Markov (MDPs, por sus siglas en inglés), también conocidos como programas dinámicos estocásticos o problemas de control estocástico, son modelos usados para la toma de decisiones en secuencia cuando las salidas del sistema controlado son inciertas [5]. Formalmente, un MDP es una tupla (S; A; P; ;R), donde:

- S es un conjunto de estados,
- A es un conjunto de acciones,
- $P(s_{t+1}|s_t; a_t)$ son las probabilidades de transición de estados para todos
- los estados s_t; $s_{t+1} \in S$ y acciones $a \in A$,
- $\gamma \in [0; 1)$ es el factor de descuento, y
- $R : S \rightarrow R, o\ bien\ R : S \times A \rightarrow R$ es la función de recompensa.

3 Descripción de la Implementación de la Solución

Se implementó la solución mediante un software desarrollado en HTML5, CSS3, Javascript (ECMAScript 5) con soporte en Chrome 25.0+ y Firefox 17+.
Se utilizó el framework AngularJS v1.1.5 y diversas bibliotecas para la generación de gráficos, animaciones y contenido, listadas a continuación:

- Angular Drag & Drop v1.0.0 by Amit Gharat: Soporte de draggables y droppables proporcionados por la biblioteca jQuery UI para poder mover la meta y al agente en el grid mediante eventos de arrastrar y soltar.
- Bootstrap (CSS + JS) v2.3.2 by Twitter: colección de herramientas de software libre para la creación de sitios y aplicaciones web.
- HighCharts JS v3.0.1: biblioteca escrita en Javascript que permite la creación de gráficos dinámicos complejos.
- Jquery JoyRide v2.0.3: Biblioteca para la creación de recorridos animados en una aplicación web.
- jQuery Pines Notify (pnotify) Plugin v1.2.0: Biblioteca empleada para la generación de notificaciones animadas mediante javascript.
- jQuery v1.9.1: Biblioteca que permite manipular el DOM, manejar eventos, crear animaciones e interactuar con AJAX de una forma simplificada.
- jQuery UI v1.9.2: Biblioteca de animaciones, plug-ins y widgets para el framework jQuery.
- Rotate.js by Zachary Johnson: plug-in para añadir efectos de rotación a los efectos provistos por jQuery. Empleado en la rotación del agente al desplazarse por el mapa.
- jQuery spritely v0.6.4: Biblioteca para animar Sprites en CSS mediante javascript.
- Animate.css by Dan Eden: Colección de efectos en CSS para animar entradas y salidas de widgets o marcado HTML.

Debido a inconsistencias en las APIs de Web Workers, la implementación evita la concurrencia y aplicación de técnicas Multihilo, favoreciendo la simulación de concurrencia mediante temporizadores y funciones asíncronas.
El software desarrollado permite una alta parametrización del entorno y la generación de gráficos comparativos durante su ejecución.
Es posible explorar la matriz Q y editar todos los aspectos del conocimiento del agente y el entorno de forma intuitiva y sencilla.
Además cuenta con una visita guiada explicando las funcionalidades más importantes.
El estilo visual y gran parte del marcado HTML fueron tomados de la plantilla Photon UI.

3.1 Problema episodico

Problema en el que existe un estado final. Por ejemplo un juego en el que se gana o se pierde. **Un episodio** consiste en una secuencia de pasos desde un estado inicial a un estado final.

3.2 Entorno o Medio ambiente:

El medio ambiente será un grid cuadrado cuyas dimensiones mínima y máxima son 6x6 y 10x10 respectivamente. Este posee un agente, un estado inicial (I) y un estado final (F). Además de I y F, cada estado (cuadrícula del grid) puede tener 4 calidades diferentes. Césped (rojo) calidad mala, césped (verde) buena, camino de piedra (marrón) excelente y color arena es neutral.
Existen además los estados pared (montículo de piedras) por el que no está permitido el tránsito del agente y otro estado pozo por el cual el agente no pude pasar porque muere.
El agente puede moverse en 8 direcciones diferentes, como indican las flechas, siempre que no salga del grid o el próximo estado sea una pared.
El objetivo del agente será evitar pasar por estados pozo o encontrar paredes en su recorrido. El objetivo del agente es alcanzar el estado final con la máxima recompensa a largo plazo.
Cuando el agente pasa por un estado pozo o llega al estado final, se inicia un nuevo episodio desde un estado inicial aleatorio distinto de pozo, pared o el estado final.
El objetivo de la solución implementada es aproximarse a la política óptima.

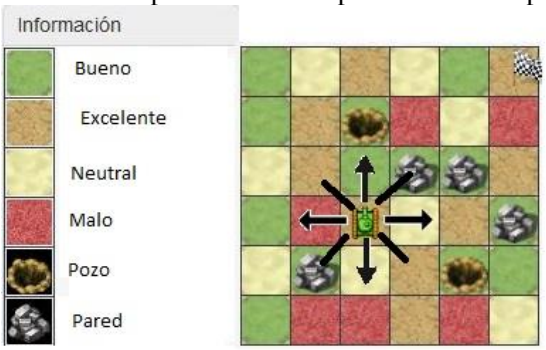

Fig.2. Representación del entorno.

3.3 Aprendizaje del agente

El aprendizaje del agente se realiza a través de la implementación del algoritmo de Q-Learning. Basándose en una estrategia de selección de acción (Greddy, e-Greddy, Softmax y Aleatorio) con el objetivo de que al final del entre-

namiento el agente sea capaz de evaluar la calidad de tomar ciertas acciones a tomar a partir de un estado determinado.
El parámetro gamma puede tomar valores comprendidos entre cero y uno , para obtener recompensas inmediatas debe tomar valores cercanos a cero, sin embargo poder lograr las mejores recompensas a futuro debe aproximarse a uno, para el caso de estudio quedo definido en 0,9.

3.4 Recompensas del agente:

Las recompensas elegidas fueron las siguientes:
Si el movimiento elegido por el agente lo coloca en el estado final, su recompensa es 200.
Si el movimiento elegido por el agente lo lleva a un estado pared, este estado no es válido.

3.5 Estados:

Los estados están definidos por la posición actual del agente, (i, j). Por lo que cada coordenada posible es un estado diferente. Cada par de coordenadas (i, j) están asignadas a un identificador, para facilitar la búsqueda de este estado.

3.6 Estrategia avariciosa con valores iniciales optimistas:

Se basa en partir de unos valores estimados mucho mayores que las recompensas medias reales. De esta forma, al seleccionar una acción se obtiene una recompensa menor que la estimada y se reduce la estimación. Esto hace que la siguiente iteración no escoja esta acción (que dejaría de tener la estimación más alta) lo que favorece la exploración.
Esta estrategia consigue una alta exploración inicial seguida de una fuerte explotación final.

3.7 Posibles acciones del agente:

El mapa del entorno del problema está representado por una matriz cuadrada de $n * n$, donde la posición del agente esta dado por las coordenadas del estado actual del agente (i, j), las acciones posibles que el agente puede tomar en relación a la posición en la que se encuentra, solo puede elegir estados que cumplan con la siguiente condición: $\forall\, i,j \;\geq\; 0 \;\wedge\; i,j \;\leq\; n.$

4 Simulaciones realizadas y resultados obtenidos

Tabla 1. Parametría de las simulaciones

	Política	**Cantidad de ciclos**	**Tamaño del grid**	**Cantidad de pozo**	**Cantidad de paredes**	**Temperatura τ**	**Probabilidad de exploración ε**
Simulación 1	ε-greedy	10000	8 x 8	15	13	--	0,2
Simulación 2	ε-greedy	10000	8 x 8	15	13	--	0,35
Simulación 3	soft-max	10000	8 x 8	15	13	50	--
Simulación 4	soft-max	10000	8 x 8	15	13	90	--
Simulación 5	aleatoria	10000	8 x 8	15	13	--	--

Tabla 2. Resultados de las simulaciones

	Tiempo promedio de entrenamiento del agente (ms)	**Cantidad de episodios en aproximarse a la política optima**	**Cantidad de veces que llego a la meta en una simulación**	**Cantidad de veces que muere el agente en una simulación**
Simulación 1	155	2850	7296	2704

Simulación 2	181	1800	5467	4533
Simulación 3	819	320	4926	5074
Simulación 4	478	250	1761	8239
Simulación 5	159	1250	312	9688

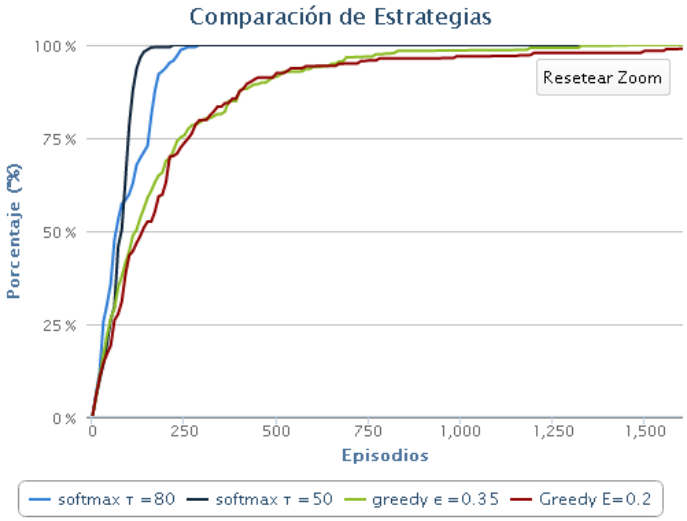

Fig. 3. Simulaciones realizadas

5 Conclusiones

El modelo definido para el problema ha funcionado correctamente. A primera vista, las pruebas de la implementación han dado buenos resultados en las distintas simulaciones efectuadas.

Los resultados obtenidos permiten observar el comportamiento del agente los algoritmos de aprendizaje por refuerzo presenta las siguientes características:

La política aleatoria suele dar mejores resultados que e-greedy. Si el mapa contiene una gran cantidad de pozos, el rendimiento de la política de aprendizaje aleatorio empeora porque el agente es propenso a caer en pozos durante todo el transcurso del entrenamiento. Sin embargo, políticas de selección de acción como softmax o e-greedy buscan explotar, en mayor o

menor medida, los conocimientos obtenidos.
En muchas ocasiones, al crecer el tamaño del mapa y el número de pozos, si bien ε -greedy aprende con mayor velocidad en un principio, a medida que épsilon tiende a 0, la velocidad de aprendizaje se decrece. Incluso con un ε de 0.2, la politica demoraba notablemente en estabilizar la matriz Q. Al principio se aproximaba más rápido que la estrategia aleatoria, pero al transcurrir los episodios se consolida el conocimiento y comienza a tomar el mismo camino en muchos episodios por lo que aprende más lentamente, mientras que la estrategia aleatoria no sufre de este problema.
En la mayoría de las simulaciones realizadas, la política que dio mejores resultados fue softmax. Al combinar explotación y exploración haciendo una selección ponderada con un factor de aleatoriedad entre las acciones posibles, no sufre de los problemas asociados a la estrategia aleatoria en mapas de muchos pozos, ni el ralentizamiento de las estrategias codiciosas con un ε bajo al transcurrir los episodios.
La estrategia política y softmax tienen, en la gran mayoría de las pruebas, un aprendizaje más rápido que ε-greedy.
La matriz Q alcanza, en promedio, un grado aceptable de estabilidad (la variación de sus valores de una iteración a otra es despreciable después del episodio 5000 aproximadamente).

6 Bibliografía

6.1 Referencias

[1] RICH E., KNIGHT K. - *Inteligencia Artificial - Ed. Mc Graw Hill. 1996*. Apuntes de la Cátedra.

[2] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[3] A. Tamar, D. Di Castro, and R. Meir. Integrating a partial model into model free reinforcement learning. Journal of Machine Learning Research,1966.

[4] (Ratitch) Bohdana Ratitch – Current Exploration Approaches in RL – Lecture Notes for COMP-526B –McGill University (2003)

[5] M. Puterman. Markov Decision Processes. Wiley, 1994.