SARSA_BB: Un algoritmo *on policy* para Sistemas Clasificadores

Marcelo Errecalde, Ana Garis, Guillermo Leguizamón
LIDIC - Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
5700 San Luis, Argentina
{merreca,agaris,legui}@unsl.edu.ar

Resumen

El modelo básico de Aprendizaje por Refuerzo (AR) está integrado por un agente y un ambiente que interactúan entre sí. El agente debe, mediante un proceso de prueba y error, aprender a mapear situaciones en acciones intentando, a lo largo del tiempo, maximizar la recompensa que el ambiente le provee. El AR caracteriza una clase de problemas de aprendizaje, cuya resolución se ha basado en dos grandes clases de métodos: los Sistemas Clasificadores (SC) y los métodos de diferencia temporal (o métodos TD). El objetivo del presente trabajo es realizar una contribución en la transferencia de experiencias entre SC y métodos TD. Para ello, se presenta un nuevo esquema para la actualización de la fortaleza de las reglas de un Sistema Clasificador, tomando como base el método de TD denominado SARSA. El algoritmo resultante, al que denominamos SARSA_BB, tiene varios atributos interesantes: a) su fórmula de actualización se ha demostrado que garantiza la convergencia a una política óptima bajo condiciones particulares; b) no require mayores modificaciones a la forma de actualización estándard utilizada en SC; c) es un algoritmo *on-policy* y por lo tanto puede tener un mejor desempeño que algoritmos *off-policy* como Q-Learning, en problemas donde la exploración que efectivamente realiza el agente impacta significativamente en las recompensas recibidas desde el ambiente. Para mostrar este último aspecto, SARSA_BB, es comparado con Q-Learning en un problema con estas características.

Palabras claves: Sistemas Clasificadores, Aprendizaje por Refuerzo, Algoritmos on-policy.

1. Introducción

Los Sistemas Clasificadores (SCs) [2] y los métodos de diferencias temporales (TD) [7, 11] son dos grandes clases de métodos ampliamente utilizados dentro del área de Inteligencia Artificial conocida como Aprendizaje por Refuerzo (AR) (en inglés “Reinforcement Learning”).

El modelo básico de Aprendizaje por Refuerzo (AR) está integrado por un agente y un ambiente que interactúan entre sí. El agente debe, mediante un proceso de prueba y error, aprender a mapear situaciones en acciones, intentando a lo largo del tiempo maximizar la recompensa que el ambiente le provee. Si bien los SCs y los métodos TD comparten características que los encuadran dentro del modelo de AR, desde sus inicios siempre fueron estudiados en forma paralela. Sin embargo, Sutton [7], Twardowski [10], Roberts [4], Wilson [12], Dorigo y Bersini [1] observaron ciertas similitudes entre

ellos, en particular entre el algoritmo base en SCs (Bucket Brigade) y el método de TD más popular, conocido como Q-Learning (QL).

Los SCs en la formulación de Holland en 1986, son más generales que los métodos de TD, pero esta generalidad dificulta el entendendimiento de la gran cantidad de interacciones involucradas [12]. Esto ha llevado a plantear distintas simplificaciones de los SCs que permitan una mejor comprensión de su funcionamiento. Estas simplificaciones han permitido además, reconocer similitudes entre el algoritmo de Bucket Brigade de SCs y el algoritmo QL como las planteadas en [1] y [12].

En este artículo, se presenta un nuevo esquema para la actualización de la fortaleza de las reglas de un Sistema Clasificador, pero en lugar de basarlo en Q-Learning se toma como base el método de TD denominado SARSA. El algoritmo resultante, denominado SARSA_BB, tiene varios atributos interesantes: a) su fórmula de actualización se ha demostrado que garantiza la convergencia a una política óptima bajo condiciones particulares [6]; b) no requiere mayores modificaciones en la forma de actualización estándard utilizada en SC como la requerida cuando se utiliza QL; c) sus características de algoritmo *on-policy* puede redundar en una mejor performance en relación a algoritmos *off-policy* como QL, en problemas donde la exploración que efectivamente realiza el agente impacta significativamente en las recompensas recibidas desde el ambiente. Para mostrar este último aspecto, SARSA_BB, es comparado con QL considerando un problema que es la combinación de dos problemas previamente estudiados en el ámbito de AR.

El presente artículo está organizado de la siguiente manera. La Sección 2 describe las simplificaciones de un SC realizadas en [1] como paso inicial para derivar un SC basado en QL (D_{MAX}-VSCS). Estas mismas simplificaciones son asumidas en este trabajo como primer paso en la definición de nuestra propuesta (SARSA_BB). La Sección 3 contiene una descripción del algoritmo de aprendizaje denominado SARSA, que forma la base para SARSA_BB. En la Sección 4 se resaltan las similitudes entre la primera aproximación a SARSA_BB (D_{ACT}-VSCS) y el algoritmo SARSA. Asimismo se muestra mediante un ejemplo que estos dos algoritmos no son totalmentes equivalentes, presentándose posteriormente en la Sección 5 las modificaciones necesarias para obtener una equivalencia directa con SARSA y que resulta en el algoritmo SARSA_BB. Esta Sección también incluye la descripción de un problema y el análisis de los resultados obtenidos a partir de la aplicación de los algoritmos SARSA_BB y QL al problema propuesto. Finalmente, las conclusiones se presentan en la Sección 6.

2. VSCS: Un sistema clasificador muy simplificado

VSCS (a Very Simple Classifier System), es un SC muy restringido, propuesto en [1] como paso inicial para resaltar las similitudes entre un SC y Q-Learning. Las restricciones impuestas en un VSCS son las siguientes: (i) los clasificadores tienen una condición y una acción, (ii) la lista de mensajes tiene capacidad igual a 1 y el único slot es reservado para mensajes ambientales, (iii) el alfabeto de codificación de los clasificadores no incluye el símbolo ”don’t care” (#), (iv) el conjunto de clasificadores contiene todos los posibles clasificadores (pares estado-acción) haciendo innecesario el uso de algoritmos genéticos. Esta última restricción permite una correspondencia uno a uno entre los clasificadores y los pares de estado acción de Q-Learning.

El algoritmo VSCS que resulta de estas restricciones es mostrado en la Figura 1.

En VSCS, la ecuación que establece el cambio en la fortaleza de un clasificador c es:

$$S_{t+1}(c_c, a_c) = S_t(c_c, a_c) + \alpha \cdot \left[\frac{R_a}{\alpha} + S_{t+1}(c_d, a_d) - S_t(c_c, a_c) \right] \quad (1)$$

donde $S_{t+1}(c_c, a_c)$ es la fortaleza del clasificador c en el tiempo $t+1$, las condiciones y las acciones de los clasificadores son llamadas c y a respectivamente, y los subscriptos c y d identifican los cla-

Inicialización
    Crear un clasificador por cada par estado-acción;
    t := 0;
    Inicializar $S_t(c_c, a_c)$, la fortaleza en el tiempo t del clasificador c;
Repetir por siempre
    Leer(m) {m es el mensaje del sensor};
    Sea M el conjunto de clasificadores que hacen matching con m;
    Elegir el clasificador a activar (disparar) $c \in M$, con una probabilidad dada por $S_t(c_c, a_c) / \sum_{d \in M} S_t(c_d, a_d)$;
    Cambiar la fortaleza de los clasificadores de acuerdo al algoritmo de Bucket Brigade;
    t := t+1;
    Ejecutar(a_c);

Figura 1: El sistema VSCS

sificadores a los que las condiciones y acciones pertenecen (así, por ejemplo, c_d es la condición del clasificador d). R_a es la recompensa ambiental recibida por el agente.

En [1], se observa que esta regla de actualización guarda similitudes con la regla usada en Q-Learning. En el caso de Q-Learning la actualización del valor de Q para el par estado-acción (s, a) es:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \cdot [R_a + \gamma \cdot \max_{a'} Q(s', a') - Q_t(s, a)] \tag{2}$$

donde s' es el estado obtenido cuando se ejecuta la acción a en el estado s. Si en el VSCS, la condición del clasificador c, c_c, representa el estado del sistema, y la parte de acción, a_c, representa la acción, podemos establecer una analogía con Q-Learning y tomar $c_c \Leftrightarrow s$, $a_c \Leftrightarrow a$ y $c_d \Leftrightarrow s'$, pudiendo reescribir la ecuación 1 como:

$$S_{t+1}(s, a) = S_t(s, a) + \alpha \cdot \left[\frac{R_a}{\alpha} + S_{t+1}(s', a') - S_t(s, a)\right] \tag{3}$$

En [1] se plantea que cambiando la fórmula 3 y ajustándola para coincidir con la fórmula 2 se obtiene un sistema clasificador que denominan D$_{MAX}$-VSCS y que difiere *solamente* en la forma en que se calcula el error, en particular:

1. Q-Learning evalúa el siguiente estado eligiendo el valor de la *mejor* acción mientras que en VSCS es evaluado por la fortaleza del par estado-acción *realmente usado.*

2. en VSCS, la evaluación del siguiente estado no es descontado (no es multiplicado por γ).

Indudablemente la inclusión del factor de descuento γ en la fórmula del VSCS puede ser importante dependiendo de la naturaleza del problema. Además es una componente esencial para garantizar que el valor esperado final de una acción esté acotado cuando no existe un estado absorvente (ver [9] para una descripción de estos aspectos).

Sin embargo, la incorporación del operador *max* en D$_{MAX}$-VSCS no es un aspecto secundario. Más allá de que la intensión es hacer un algoritmo totalmente equivalente a Q-Learning, esto tiene dos consecuencias importantes. Por un lado, el tipo de actualización utilizada se escapa totalmente del esquema de actualización de fortalezas utilizada en Sistemas Clasificadores. Por otra parte, el uso del operador *max* de Q-Learning, no siempre puede resultar en la mejor política de control del agente. Para corroborar esta afirmación, en las secciones siguientes mostraremos que con una pequeña

modificación en los tiempos en que se realiza la actualización de las fortalezas de los clasificadores, un VSCS puede ser totalmente equivalente a otro algoritmo de AR (SARSA), que además de exhibir garantías de convergencia similares a las de Q-Learning, en algunos tipos de ambientes puede obtener un mejor desempeño.

3. SARSA: Un algoritmo TD para control *on-policy*

Dentro del ámbito de AR se desarrolló otro algoritmo muy similar al de Bucket Brigade, que tiene incluso más semejanza que con Q-Learning. En el año 1994 Rummery y Niranjan[5] exploraron el método denominado *QL modificado* que posteriormente en 1996 Sutton rebautizó como SARSA [8]. SARSA, al igual que Q-Learning, es un método de diferencia temporal (TD). Si bien ambos métodos intentan aprender los valores óptimos de Q para todos los pares estado-acción (s, a) ($Q^*(s, a)$), la forma en que ambos métodos aproximan Q^* es sustancialmente diferente. Q-Learning es un método *off-policy* y su fórmula de actualización presentada en la ecuación 2 aproxima directamente Q^*, *independientemente de la política que está siendo utilizada por el agente para elegir sus acciones*. SARSA, en cambio, es un método *on-policy* que aproxima Q^* mediante un proceso general denominado *iteración de política generalizada*. Como todo método *on-policy*, SARSA continuamente estima Q^π para la política de comportamiento π que está siguiendo el agente, y al mismo tiempo cambia π para hacerla más *greedy* con respecto a Q^π (ver [9] para una descripción exhaustiva del proceso de iteración de política generalizada).

La regla de actualización utilizada en SARSA es:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \cdot [R_a + \gamma \cdot Q(s', a') - Q_t(s, a)] \tag{4}$$

donde s es el estado del ambiente en el tiempo t, a es la acción seleccionada en ese estado, R_a es la recompensa obtenida al ejecutar a, s' es el estado al que se arribó luego de ejecutar a en s y a' es la acción que posteriormente se seleccionó en s'. Como vemos, esta regla utiliza cada elemento de la quíntupla (s, a, R_a, s', a'), lo que da origen al nombre SARSA.

La forma general de SARSA está dada por el algoritmo mostrado en la Figura 2.

1- Inicializar cada entrada $Q(s, a)$ arbitrariamente
2- DO (por cada episodio)
3- Determinar estado inicial s
4- WHILE (no finalice el episodio)
5- Seleccionar una acción a desde el estado s usando una política derivada de Q (por ejemplo $\epsilon-$greedy)
6- WHILE (no se llegue a un estado terminal)
7- Ejecutar la acción a. Observar recompensa r y nuevo estado s'
8- Seleccionar acción a' desde s' usando una política derivada de Q (por ejemplo $\epsilon-$greedy)
9- $Q(s, a) = Q(s, a) + \alpha.[r + \gamma.Q(s', a') - Q(s, a)]$
10- $s = s'$
11- $a = a'$

Figura 2: Algoritmo de Aprendizaje SARSA.

Las propiedades de convergencia de SARSA dependen de la naturaleza de la dependencia de la política sobre Q. Por ejemplo, una política podría usar ϵ-greedy. SARSA converge con probabilidad

uno a una política óptima en la medida que todos los pares estado-acción son visitados un número infinito de veces y la política converge en el límite a la política greedy [9].

4. D_{ACT}-VSCS: Primera aproximación a un SC basado en SARSA

Siguiendo el mismo razonamiento que el utilizado en [1], se podría definir un algoritmo D_{ACT}-VSCS, incorporando en la fórmula 3 el factor de descuento γ de manera tal de obtener una regla de actualización de fortalezas para sistemas clasificadores que sea equivalente con la utilizada por SARSA (fórmula 4). La regla de actualización para D_{ACT}-VSCS, quedaría:

$$S_{t+1}(s,a) = S_t(s,a) + \alpha \cdot [R_a + \gamma \cdot S_{t+1}(s',a') - S_t(s,a)] \tag{5}$$

Un error común al analizar las fórmulas de actualización, es considerar sólo la *forma* de la regla y no el *tiempo* en que las actualizaciones se llevan a cabo. En este sentido, si consideramos las fórmulas 4 y 5 podríamos decir que la regla de actualización utilizada en D_{ACT}-VSCS es totalmente equivalente a SARSA. Sin embargo, esto no es necesariamente cierto. Para entender este aspecto, es conveniente desarrollar la fórmula 5 de forma tal que todos los términos que participan en la actualización queden explícitamente separados:

$$S_{t+1}(s,a) = S_t(s,a) + \underbrace{\alpha \cdot R_a}_{A} + \underbrace{\alpha \cdot \gamma \cdot S_{t+1}(s',a')}_{B} - \underbrace{\alpha \cdot S_t(s,a)}_{C} \tag{6}$$

En SARSA, los términos de la actualización referenciados como A, B y C, son utilizados para actualizar la función de valor del par (s,a) *una vez que la acción* a' *ya ha sido seleccionada en el estado* s'.

Sin embargo los SC en general y el VSCS en particular, realizan las actualizaciones de estos términos en dos etapas distintas.

Cuando un clasificador gana la subasta y su acción asociada es elegida para ser enviada al ambiente, se le descuenta una proporción de su fortaleza (término C) y se le acredita una proporción de la recompensa ambiental (término A). Sin embargo, el pago que recibe del siguiente clasificador que se activa (término B) en retribución por haber favorecido su activación, se realiza una vez que el próximo clasificador ha ganado la siguiente subasta. Es importante notar que hasta entonces, la fortaleza del clasificador asociado con el par (s,a) exhibe una disminución considerable, ya que usualmente el término C descontado es mayor que el término A acreditado (el primero es una proporción de las recompensas acumuladas mientras que el segundo es una proporción de sólo una recompensa). Este aspecto, puede no ser importante cuando el estado s' es distinto del estado anterior s. No ocurre los mismo si el agente, cuando toma la acción a en s vuelve a caer en s ($s' = s$) como podría ser el caso cuando el agente choca contra una pared de un laberinto y permanece en el mismo estado. En este caso, el clasificador asociado con la acción (s,a) estará en desventaja para ser seleccionado nuevamente con respecto a las otras acciones en el mismo estado.

Para apreciar mejor esta diferencia, observemos el siguiente ejemplo donde se tienen sólo dos estados $\{0,1\}$ y las acciones posibles son $\{0,1,2,3\}$. Las diferencias entre las actualizaciones de las fortalezas realizadas por D_{ACT}-VSCS y las modificaciones de la tabla Q realizadas por SARSA se muestran en la tabla 1. En ambos algoritmos las acciones se seleccionan en forma greedy. Sólo se muestra el valor de las fortalezas y de la tabla Q que son relevantes al ejemplo. Los tiempos en que las actualizaciones A, B y C de la fórmula 6 utilizada por D_{ACT}-VSCS se resaltan en las correspondientes

entradas de la tabla. Como se puede observar, en el estado 0 cuando se elige una acción por segunda vez (a'), D$_{ACT}$-VSCS ya actualizó los términos A y C de la fórmula 6 correspondientes al clasificador 1 (par estado acción $(0, 0)$). Por lo tanto la acción 0 en ese estado tendrá desventaja en la subasta con respecto a las otras acciones. Para este ejemplo particular, vemos que SARSA quien todavía no actualizó la entrada correspondiente al par $(0, 0)$ vuelve a elegir la acción 0 mientras que D$_{ACT}$-VSCS selecciona la acción 1, correspondiente al clasificador 2.

5. SARSA_BucketBrigade (SARSA_BB)

Expuestas las semejanzas y diferencias entre D$_{ACT}$-VSCS y SARSA, vemos que para obtener un SC equivalente a SARSA sólo es necesario utilizar la misma fórmula de actualización que D$_{ACT}$-VSCS y garantizar que la actualización de la fortaleza de un clasificador dada por la fórmula 6 se realice una única vez al final del ciclo de ejecución como lo hace SARSA. Utilizando terminología de SC esto equivale a utilizar el algoritmo de Bucket Brigade (con el factor γ incorporado) y que la cámara de compensación (clearinghouse) lleva a cabo sus actividades luego de haber determinado cuál es el siguiente ganador.

El algoritmo resultante lo denominamos SARSA_BucketBrigade (o SARSA_BB) y se muestra en la figura 3

```
1- Inicializar la fortaleza S de cada clasificador c arbitrariamente
2- DO (por cada episodio)
3-      Determinar estado inicial s
4-      Detectar señal del ambiente
5-      WHILE (no finalice el episodio)
6-              oldwin = clasificador ganador de subasta
7-              WHILE (no se llegue a un estado terminal)
8-                      Ejecutar acción a correspondiente a oldwin
9-                      Observar recompensa r y nuevo estado s'
10-                     Detectar señal del ambiente
11-                     win = clasificador ganador de subasta
12-                     S(oldwin) = S(oldwin) + a [r + g S(win) - S(oldwin) ]
13-                     oldwin = win
14-                     s = s'
15-                     a = a'
```

Figura 3: Algoritmo de Aprendizaje SARSA_BB

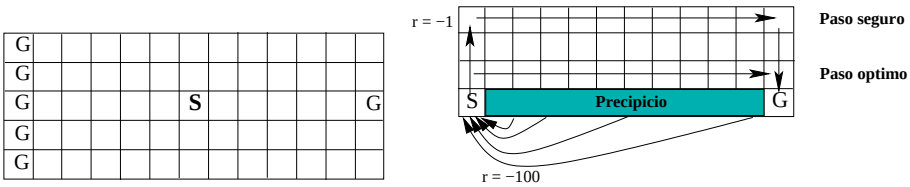
5.1. Presentación del Problema

En esta sección se presenta el problema propuesto para nuestro estudio. El problema puede visualizarse como una combinación de dos problemas muy interesantes para ser estudiados en el ámbito del aprendizaje por refuerzo. El primero es el problema del *muro del placer* [3] el cual puede ser observado en la Figura 4(a). En el muro del placer, el agente se encuentra situado inicialmente (estado inicial S) en el medio de un entorno artificial (mundo-grilla) y el objetivo es alcanzar uno de los estados objetivos (G) que se encuentran a su derecha o izquierda en el menor número de pasos. La política aprendida por un algoritmo que aproxima directamente la política óptima (como Q-Learning) sería indiferente en el estado S entre ir hacia la izquierda o la derecha. En S lo mejor que se puede

<table>
<tr><th>Paso en la ejecución</th><th>Algoritmo D$_{ACT}$-VSCS</th><th>Algoritmo SARSA</th><th>Comentarios</th></tr>
<tr><td>Estado Inicial s</td><td>s: 00
Ganador anterior: 5)
Clasificador Fortaleza (F)
1) 00:00 10
2) 00:01 9.5
3) 00:10 8
4) 00:11 7
5) 01:00 10</td><td>$s = 0$
<table><tr><td>Tabla Q</td><td colspan="4">Acción</td></tr><tr><td>Estado</td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>10</td><td>9.5</td><td>8</td><td>7</td></tr><tr><td>1</td><td>10</td><td></td><td></td><td></td></tr></table></td><td>En D$_{ACT}$-VSCS, suponemos que el ganador del ciclo anterior de ejecución corresponde al clasificador 5)</td></tr>
<tr><td>Elegir acción a</td><td>Matching: 1) 2) 3) 4)
$\alpha = 0.1$
$\gamma = 0.1$
Clasificador Fortaleza Oferta
1) 00:00 10 1.0
2) 00:01 9.5 0.95
3) 00:10 8 0.8
4) 00:11 7 0.7
Ganador: 1)
Acción correspondiente al ganador: 00</td><td>Dado que se encuentra en el estado 0, se elegirá entre los valores de la tabla Q correspondientes a $Q(0,0)$, $Q(0,1)$, $Q(0,2)$ y $Q(0,3)$
$Q(0,0)$ es el que tiene el mayor valor
Acción elegida: 0</td><td>- D$_{ACT}$-VSCS elige la acción del clasificador que haya realizado la mayor oferta en la subasta. Si un clasificador tiene fortaleza f su oferta es $= \alpha f$.
-SARSA selecciona la acción con mayor valor Q, de acuerdo al estado en que se encuentre.</td></tr>
<tr><td>Modificar valor de utilidad</td><td>Ganador anterior: 5) Ganador actual: 1)
Decrementar la oferta al ganador:
$S(1) = S(1) - \alpha S(1)$ (C)
Pagar al ganador anterior:
$S(5) = S(5) + \alpha\gamma S(1)$
Clasificador Fortaleza
1) 00:00 9
2) 00:01 9.5
3) 00:10 8
4) 00:11 7
5) 01:00 10.1</td><td></td><td>$S(n)$ es utilizado para denotar la fortaleza del clasificador n. Así por ejemplo $S(1)$ denota la fortaleza del clasificador 1 y es una abreviatura para $S(0,0)$. El paso identificado como (C) corresponde al término C de la fórmula 6.</td></tr>
<tr><td>Ejecutar acción a, obtener nuevo estado s’y recompensa r</td><td>acción a: 00
nuevo estado s’: 00
recompensa r: 1</td><td>acción a: 0
nuevo estado s’: 0
recompensa r: 1</td><td></td></tr>
<tr><td>Recompensar</td><td>Recompensar al ganador:
$S(1) = S(1) + \alpha r$ (A)
Clasificador Fortaleza
1) 00:00 9.1
2) 00:01 9.5
3) 00:10 8
4) 00:11 7</td><td></td><td>El paso identificado como (A) corresponde al término A de la fórmula 6.</td></tr>
<tr><td>Seleccionar acción a’</td><td>Matching: 1) 2) 3) 4)
$\alpha = 0.1$
$\gamma = 0.1$
Clasificador Fortaleza Oferta
1) 00:00 9.1 0.91
2) 00:01 9.5 0.95
3) 00:10 8 0.8
4) 00:11 7 0.7
Ganador: 2)
Acción correspondiente al ganador: 01</td><td>Dado que se encuentra en el estado 0, se elegirá entre los valores de la tabla Q correspondientes a $Q(0,0)$, $Q(0,1)$, $Q(0,2)$ y $Q(0,3)$
$Q(0,0)$ es el que tiene el mayor valor
Acción elegida: 0</td><td></td></tr>
<tr><td>Modificar valor de utilidad</td><td>Ganador anterior: 1)
Ganador actual: 2)
Decrementar la oferta al ganador:
$S(2) = S(2) - \alpha S(2)$
Pagar al ganador anterior:
$S(1) = S(1) + \alpha\gamma S(2)$ (B)
Clasificador Fortaleza
1) 00:00 9.195
2) 00:01 8.55
3) 00:10 8
4) 00:11 7
5) 01:00 10.1</td><td>Actualizar Q(0,0):
$Q(0,0) = Q(0,0) + \alpha[r + \gamma Q(0,0) - Q(0,0)]$
<table><tr><td>Tabla Q</td><td colspan="4">Acción</td></tr><tr><td>Estado</td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>9.2</td><td>9.5</td><td>8</td><td>7</td></tr><tr><td>1</td><td>10</td><td></td><td></td><td></td></tr></table></td><td>SARSA realiza las modificaciones de sus utilidades una única vez y al final del ciclo. D$_{ACT}$-VSCS modifica por tercera vez los suyos. El paso identificado como (B) corresponde al término B de la fórmula 6.</td></tr>
</table>

Cuadro 1: Diferencias en las formas de actualización de D$_{ACT}$-VSCS y SARSA.

hacer es ir en línea recta al objetivo más cercano, el cual se encuentra a 6 pasos en cualquiera de las dos direcciones. Sin embargo, si se toma en cuenta que la exploración ”va a ocurrir”, en el estado S es mejor ir hacia la izquierda, ya que si una acción exploratoria como ”arriba” o ”abajo” gasta un paso de tiempo, el agente permanece aún a la misma distancia del objetivo más cercano.



(a) Problema del “Muro del Placer”

(b) Problema del “Precipicio”.

Figura 4: Dos problemas para analizar el rol de la la exploración

El segundo problema (Figura 4(b)) al que denominamos *problema del precipicio* [9], consiste en un agente situado inicialmente en un extremo de un mundo tipo grilla (estado S), que debe alcanzar un estado objetivo (estado G) en el otro extremo de dicho entorno en el cual existe un precipicio. La recompensa es -1 para cualquiera de las transiciones, excepto aquellas que conducen al “precipicio”, las cuales tienen un recompensa de -100. Si el agente cae en el precipicio, éste es devuelto automáticamente al estado inicial S. La particularidad de este problema está relacionada con los pasos posibles que el agente puede seguir para alcanzar dicho objetivo. Como vemos en la Figura 4(b), se han destacado dos tipos de pasos; uno, el óptimo dado que implica una mayor acumulación de recompensas. Sin embargo, por la naturaleza del problema, este paso podría ser considerado peligroso dado que su traza pasa por el borde del precipicio. Esto implica que durante las posibles fases de exploración el agente puede caer en el precipicio con la consecuente disminución de las recompensas acumuladas que éste trata de maximizar. El otro paso, si bien no es el óptimo en términos de las recompensas acumuladas, es el más seguro considerando las posibles fases de exploración del agente. Estos dos problemas si bien en apariencia son significativamente diferentes, tienen en común un aspecto importante: la exploración tiene una incidencia directa y fundamental en las recompensas que recibe el agente. Este aspecto se torna crucial si el agente necesita tener una política de exploración persistente (como la requerida en ambientes dinámicos) y no es posible reducir gradualmente la exploración hasta llegar a una política totalmente greedy, como es asumido en muchos trabajos de AR. En este sentido, trabajos previos [3, 9] han mostrado que los algoritmos *on-policy* se desempeñan mejor en este tipo de ambientes que los algoritmos *off-policy* como Q-Learning

El nuevo problema combina características de los dos problemas presentados según se muestra en la Figura 5. El problema es denominado “*placer-precipitado*”. En este mundo-grilla, el agente está situado inicialmente en la parte inferior al centro (estado S). Cuando el agente alcanza el estado G recibe una recompensa igual a 100. Si el agente se topa con un precipicio es castigado con una recompensa de -100. Para cualquier otro caso no recibe recompensa.

La parte inferior de este mundo-grilla es similar al problema del precipicio (Figura 4(b)) ubicándose un precipicio sobre el límite inferior derecho del mundo-grilla. Sin embargo, a diferencia del problema original (Figura 4(b)) existen aquí dos caminos óptimos constituidos por los pasos que bordean el gran obstáculo ubicado enfrente del estado S. Sin embargo sólo el de la izquierda es *seguro* ya que la exploración no puede hacerlo caer en el precipicio.

Es importante destacar que los estados objetivos se encuentran en la parte superior del mundo-grilla separados de la parte inferior por barreras dispuestas a modo de muro y que existe un único

paso para llegar a la parte superior. Dado que el agente necesariamente tiene que pasar por el estado Y (Figura 5, al centro), este podría ser visualizado como el estado objetivo del problema del precipicio original. Luego, la parte superior del mundo-grilla desde el estado Y es alcanzada sólo a través del corredor formado por las barreras cuya salida a la parte superior donde se encuentran los estados objetivos, estado X, representaría el estado inicial del problema del *muro del placer*. A partir de ese estado, el agente se encuentra en la etapa de decidir el camino hacia la izquierda o la derecha según se explicó para ese problema. Por esta razón es la disposición de la barrera que forma una especie de embudo que desemboca en la parte central de la región superior del mundo-grilla propuesto.

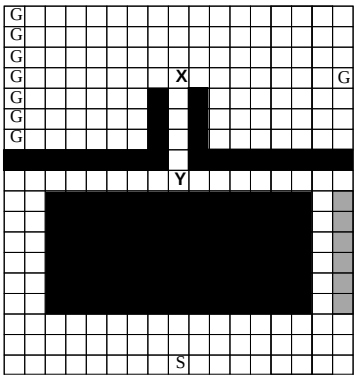


Figura 5: Problema combinado de los problemas presentados en Figuras 4(b) y 4(a).

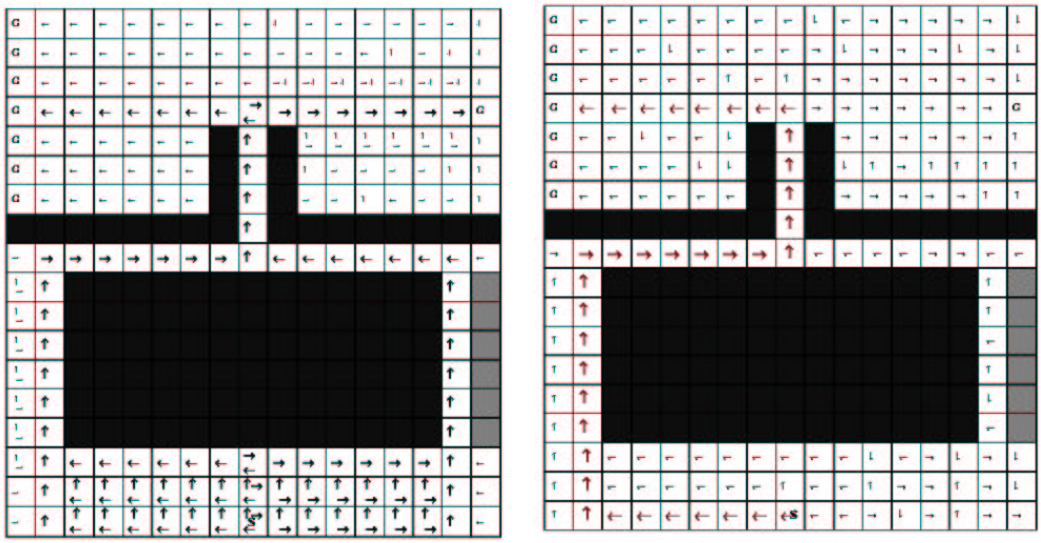
En general, para el problema *placer-precipitado*, el agente parte siempre del mismo estado inicial S y sus movimientos están limitados a 4 posibilidades: derecha, izquierda, arriba y abajo. Cada una de estas acciones lo conducirá a una celda lindante, excepto en el caso que dicha celda sea una barrera o no exista (fuera de los límites del mundo-grilla). Si "cae" por el precipicio, vuelve al estado inicial. Las acciones son determinísticas y el ambiente es estacionario. El aprendizaje del agente está dividido en episodios. Cada vez que ingresa en uno de los estados G o ejecuta un número máximo predeterminado de iteraciones o cae al precipicio, el episodio finaliza y el agente vuelve al estado inicial. Las recompensas recibidas k pasos del tiempo en el futuro son decrementadas por un factor de γ^{k-1}, con $\gamma < 1$, por lo que problema de maximizar las recompensas acumuladas en este caso es equivalente, a llegar a alguno de los estados G en el menor número de pasos posible.

5.2. Experimentos

El trabajo experimental desarrollado, no intenta realizar una comparación exhaustiva y concluyente entre los algoritmos SARSA_BB y Q-Leaning. Esto involucraría considerar distintos tipos de ambientes y conjuntos de parámetros lo que escapa al alcance del trabajo actual. El objetivo de la experimentación es mostrar las características *on-policy* de SARSA_BB y las diferencias en cuanto a la política de control aprendida con respecto a métodos *off-policy* como Q-Learning. También se muestra de qué manera las políticas de control aprendidas pueden tener un impacto considerable sobre las recompensas acumuladas de los agentes.

Los resultados mostrados son el promedio de un total de 10 experimentos. Cada experimento consistió en la ejecución de 2000 episodios. Los eventos de fin de episodio se corresponden con los explicados previamente, siendo 20000 el número máximo de iteraciones permitido para cada episodio. El tamaño del paso α fue de $0{,}5$. La estrategia de exploración utilizada fué la ϵ-greedy con $\epsilon = 0{,}2$. Esta estrategia se comporta de forma greedy por defecto, pero con probabilidad ϵ elige una acción en

forma aleatoria. El valor de γ fue de $0{,}95$, lo cual se corresponde con valores usualmente utilizados para este tipo de problemas.



(a) Q-Learning (b) SARSA-BB

Figura 6: Políticas aprendidas por Q-Leaning y SARSA_BB

La figura 6 muestra las políticas aprendidas por Q-Learning y SARSA_BB para el problema seleccionado. Las flechas dibujadas en cada casilla indican la acción sugerida de acuerdo al máximo valor de la tabla Q o el clasificador de máxima fortaleza respectivamente.

Las gráficas muestran que las políticas aprendidas por SARSA-BB y Q-Learning difieren considerablemente. Q-Learning aprende las funciones de valor óptimas pero es indiferente entre seguir un camino seguro o altamente riesgoso. Esto se puede observar en el estado de comienzo S donde el agente tiene prácticamente una política de control óptima simétrica hacia la izquierda y la derecha. SARSA_BB por su parte aprende una política de control óptima pero a la vez segura. Si se observa el estado de comienzo S se verá que la política sugiere ir por el camino que no bordea el precipicio. Lo mismo se registra hasta 2 columnas a la derecha del estado inicial por lo que si consideramos que el factor de exploración es de 0.2 las oportunidades de que el agente pueda ir por el camino que bordea el precipicio son prácticamente nulas.

Asimismo en el estado que previamente marcamos como X, Q-Learning es indiferente entre ir hacia la izquierda o la derecha. Sin embargo, la política aprendida por SARSA_BB sugiere ir hacia el "muro del placer" donde acciones exploratorias como "arriba" o "abajo" no lo alejarán de un estado objetivo. Para este tipo de problemas, la posibilidad de contar con un método de aprendizaje que efectivamente refleje los efectos de la exploración en la política aprendida puede tener un impacto significativo en la performance del agente. Para corrobar este hecho en la figura 7 se muestra la diferencia en las recompensas acumuladas por cada uno de los métodos con un desempeño significativamente mejor de SARSA_BB sobre Q-Learning.

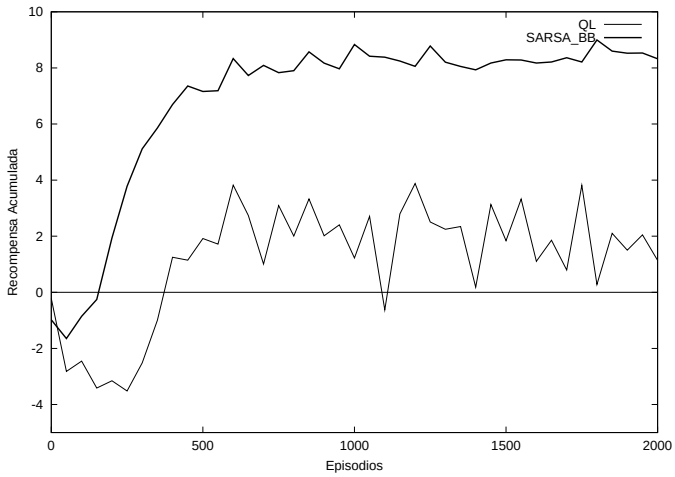


Figura 7: Recompensas acumuladas.

6. Conclusiones

En este trabajo se presentó un nuevo esquema para la actualización de la fortaleza de las reglas de un Sistema Clasificador, tomando como base el algoritmo SARSA, un algoritmo de Aprendizaje por Refuerzo basado en diferencias temporales. El algoritmo propuesto, denominado SARSA_BB (SARSA Bucket Brigade), no requiere cambiar la fórmula de actualización estándard utilizada por el algoritmo Bucket Brigade, como sí lo hacen otros SC basados en Q-Learning. En lugar de ello, para lograr una equivalencia directa entre SARSA y SARSA_BB se debe garantizar que el mecanismo de clearinghouse utilizado por el SC, se realice luego de un ciclo completo de selección de dos clasificadores.

SARSA_BB tiene dos atributos interesantes. Por un lado utiliza una forma de actualización que, bajo condiciones similares a las requeridas por Q-learning, se ha demostrado que converge a una política óptima. Por otra parte, a diferencia de Q-Learning y otros SC basados en Q-Learning, SARSA_BB implementa un algoritmo de control "*on-policy*". Esta característica, convierte a SARSA_BB en buen candidato para ser utilizado en problemas donde la exploración juega un rol significativo en el desempeño del agente. Esto incluye ambientes dinámicos, donde son requeridas capacidades de exploración persistente, o problemas donde las acciones exploratorias pueden traer consecuencias muy negativas.

Las características *on-policy* de SARSA_BB fueron mostradas en un problema que es la combinación de dos problemas anteriores. Estos problemas ya habían sido utilizados para mostrar las falencias de los métodos *off-policy* para ese tipo de situaciones. Los resultados obtenidos con SARSA_BB confirman que en estos casos un algoritmo *on-policy* permitirá en general obtener una política de control del agente más aceptable que la derivada por Q-Learning o un SC basado en Q-Learning.

Ese trabajo, es un paso inicial en la transferencia de experiencias adquiridas en Aprendizaje por Refuerzo al contexto de Sistemas Clasificadores. El trabajo futuro está dirigido a utilizar SARSA-BB en el contexto de SC más complejos de manera tal de proveer al agente con capacidades de generalización y memoria. El objetivo es comparar las técnicas usualmente utilizadas en SC para tales fines (ver [1] para un análisis detallado) con las utilizadas usualmente en AR, como por ejemplo redes neuronales y redes neuronales recurrentes respectivamente.

Referencias

[1] Marco Dorigo and Hugues Bersini. A comparison of Q-learning and classifier systems. In *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, pages 248–255, 1994.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization amd Machine Learning*. Addison Wesley, 1989.

[3] George H. John. When the best move isn't optimal: Q-learning with exploration. Unpublished manuscript, available through URL ftp://starry.stanford.edu/pub/gjohn/papers/rein-nips.ps, 1995.

[4] G. Roberts. Dynamic planning for classifier systems. In *Proceedings of Fitfth International Conference on Genetic Algorithms*, pages 231–237, San Mateo, CA, 1993. Morgan Kaufmann.

[5] G. A. Rummery and M.Ñiranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, 1994.

[6] Satinder Singh, Tommi Jaakola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning Journal*, 1998.

[7] Richard S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[8] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, and Hasselmo M. E., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pages 1038–1044. MIT Press, 1996.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. The MIT Press, 1998.

[10] K. Twardowski. Credit assignment for pole balancing with learning classifier systems. In *Proceedings of Fitfth International Conference on Genetic Algorithms*, pages 238–245, San Mateo, CA, 1993. Morgan Kaufmann.

[11] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[12] Stewart W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.