

# IIC3675: Tarea 3

Bruno Cerda Mardini

a)

**Q-learning** es *off-policy* ya que en su función de actualización, el valor futuro de la state-action value function ( $Q(S', A')$ ) se elige de manera *greedy*. En específico, se elige la acción que maximiza dicho valor, como se ve en la fórmula:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

**SARSA** es *on-policy* ya que, a diferencia de Q-learning, el valor futuro de la state-action value function ( $Q(S', A')$ ) se determina eligiendo la siguiente acción según la política que se está optimizando actualmente:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

b)

Si ambos algoritmos empiezan a elegir acciones de manera greedy, entonces se vuelven muy similares, pero de todas formas **no van a ser equivalentes**.

Esto debido al orden en que se toman las acciones. En **SARSA**, se elige una acción ( $A'$ ) para el siguiente estado ( $S'$ ), luego se hace la actualización, y después esta misma acción ( $A'$ ) se utiliza en el siguiente paso. En cambio, **Q-learning** NO elige una acción ( $A'$ ) y se hace la actualización basado en la acción que simplemente maximice el valor de Q ( $\max_a Q(S', a)$ ), de esta manera Q-learning no se compromete con ninguna acción, luego en el siguiente paso Q-learning es capaz de elegir una acción pero con la tabla Q ya actualizada.

Un ejemplo claro en donde ambos algoritmos podrían tomar decisiones distintas es en caso de empate de valores, debido a que la política greedy no podría elegir un claro ganador, es una situación en donde ambos algoritmos podrían diverger en resultados y la toma de acciones, el ejemplo es el siguiente:

Supongamos que un agente se encuentra en el estado  $S$ , toma la acción  $A$  y debido a esto ahora se encuentra en el estado  $S'$ . Supongamos también que en la tabla Q, el valor para el estado  $S'$  tiene un empate:

- $Q(S', A_1) = -1$

- $Q(S', A_2) = -1$

Debido a que hay un empate, la política greedy no puede elegir una acción que sea claramente mejor, por lo que lo tiene que hacer de manera uniforme. **SARSA** va a actualizar  $Q(S, A)$ , para esto va a elegir la siguiente acción  $A'$ , pero como hay un empate, supongamos que va a terminar eligiendo  $A_1$ , por lo que en el siguiente paso va a estar obligado a ejecutar  $A_1$ . **Q-Learning** va a actualizar  $Q(S, A)$  usando la acción que maximice el valor de retorno, pero como hay un empate, supongamos que va a tomar la acción  $A_2$  como la acción que maximice el valor de  $Q(S', a)$  (**Pero no la guarda para el siguiente paso**). La actualización para  $Q(S, A)$  de ambos algoritmos va a ser la misma, pero **eligieron acciones distintas**, por lo que no son equivalentes.

c)

En la **Figura 1** podemos observar que SARSA y 4-step SARSA son mejores que Q-learning. En específico, en los primeros episodios, 4-step SARSA obtiene retornos mejores que SARSA y Q-learning, pero después SARSA converge al mismo valor que 4-step SARSA en términos de retorno promedio para los últimos episodios, mientras que Q-learning se mantiene claramente más abajo con peores valores. La principal razón por la cual obtenemos estos resultados es debido a que SARSA y 4-step SARSA son **on-policy**, y Q-learning es **off-policy**.

**Q-learning** va a explorar de manera  $\epsilon$ -greedy, pero va a actualizar sus valores  $Q(S, A)$  de manera greedy. Debido a esto, el algoritmo va a aprender a tomar el camino más corto, pero al mismo tiempo el más riesgoso, ya que con probabilidad  $\epsilon$  se va a caer al *cliff*, donde va a obtener una recompensa de -100. A diferencia de Q-learning, **SARSA** y **4-step SARSA** son **on-policy**, por lo que van a explorar de una manera  $\epsilon$ -greedy, y también actualizarán sus valores  $Q(S, A)$  en base a esa misma política. Lo anterior implica que, al explorar los estados cercanos al *cliff*, estos se actualizarán con malos valores  $Q(S, A)$ , ya que con probabilidad  $\epsilon$  el agente se va a caer. Es por esto que el agente va a terminar aprendiendo un camino más largo y seguro.

Finalmente, la razón por la cual 4-step SARSA obtiene mejores retornos promedio de manera mucho más rápida que SARSA es debido a que las actualizaciones de  $Q(S, A)$  consideran 4 pasos más adelante, por lo que logra aprender de manera más rápida que es mejor alejarse del *cliff* para evitar la recompensa de -100. Considerando todo lo anterior, el resultado que se puede observar en la figura tiene mucho sentido.



Figure 1: Comparación de retorno promedio entre Q-learning, SARSA y 4-step SARSA en Cliff Walking.

d)

En la Tabla 1 se comparan los rendimientos de Dyna-Q y RMax. En específico, se pueden observar los valores de retorno medio por episodio, donde Dyna-Q tiene múltiples columnas dependiendo de los planning steps y RMax tiene su propia columna.

En el enunciado preguntan si Dyna-Q es equivalente a RMax si se tiene un número lo suficientemente grande. La respuesta es que no, sin importar el número de planning steps, ambos algoritmos van a converger a valores distintos. En específico, RMax va a converger a una política near-optimal, mientras que Dyna-Q va a converger a una política óptima en el límite si explora con  $\epsilon - greedy$ .

Table 1: Dyna y Rmax: Retorno medio por episodio.

Episodio	n=0	n=1	n=10	n=100	n=1000	n=10000	RMax
1	-1806.4	-1910.8	-2715.4	-2467.0	-3145.6	-2832.4	-1522.0
2	-1599.4	-2627.2	-1763.2	-3126.6	-3753.2	-4420.0	-6922.0
3	-2013.4	-1677.6	-2497.4	-2721.8	-1394.4	-885.0	-4843.0
4	-1969.0	-2270.0	-2053.0	-1604.8	-2015.2	-1195.6	-804.0
5	-2422.0	-1515.0	-1298.4	-1831.2	-1272.4	-919.8	-272.0
6	-1057.4	-1021.4	-1118.0	-2468.0	-156.0	-288.8	-232.0
7	-1097.2	-801.8	-598.6	-974.2	-98.0	-93.8	-4123.0
8	-976.4	-926.4	-815.8	-201.4	-85.6	-431.4	-487.0
9	-1106.2	-752.0	-848.8	-189.2	-86.2	-81.6	-285.0
10	-996.4	-797.2	-1266.0	-510.0	-73.6	-203.8	-67.0
11	-908.6	-1092.4	-374.6	-442.6	-93.2	-89.4	-67.0
12	-910.4	-897.0	-575.0	-178.4	-79.4	-95.6	-67.0
13	-902.6	-805.4	-1764.8	-127.2	-91.4	-169.4	-67.0
14	-827.2	-1034.6	-426.4	-120.4	-86.8	-90.4	-67.0
15	-980.6	-860.2	-573.6	-125.6	-82.8	-100.4	-67.0
16	-852.2	-683.6	-352.6	-116.2	-91.6	-87.4	-67.0
17	-912.0	-1171.2	-1174.0	-109.0	-85.0	-96.2	-67.0
18	-1059.6	-747.0	-512.8	-134.2	-83.6	-87.8	-67.0
19	-1002.2	-636.6	-2148.8	-92.0	-82.2	-83.4	-67.0
20	-1158.0	-692.2	-304.8	-82.6	-91.6	-90.8	-67.0