Reinforcement Learning para el *Cacho*

Javier Maass Martínez y Juan Pablo Sepúlveda

MA4402 Simulación Estocástica

7 de diciembre de 2022

Índice

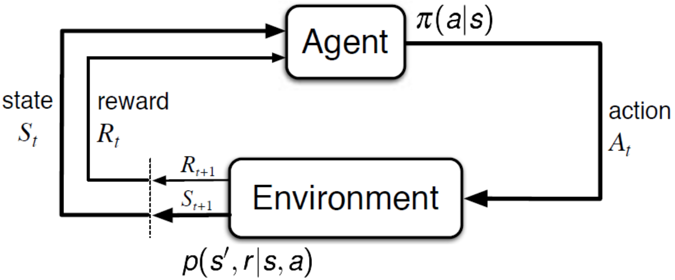
Juego de *Cacho*

El juego

- 5 dados por persona al inicio.
- Juego compuesto por rondas. En cada ronda el perdedor pierde un dado.
- Se comienza con una apuesta inicial con respecto al número de dados de alguna pinta, y se puede aumentar la apuesta (siguiendo ciertas reglas), apostar a que la predicción es exacta (calzar) o apostar a que la apuesta es mayor a la cantidad real (dudar).
- Calzar y dudar terminan la ronda.
- Calzar correctamente te devuelve un dado y quita uno al oponente.
- Calzar/dudar mal hace perder un dado.
- Dudar bien hace que el otro pierda un dado.
- El último jugador al que le quede al menos un dado gana.

Reinforcement Learning

Estados $s \in \mathscr{S}$. **Acciones** $a \in \mathscr{A}(s)$ (según **Policy** $\pi := \pi(a|s) \in \mathscr{P}(\mathscr{A}(s))$. *Función de transición del ambiente:* $p(s', r|s, a)$.



Este proceso repetido iterativamente genera una secuencia $(S_t, A_t, R_t)_{t\in\mathbb{N}}$

Modelamiento del Juego como un MDP a *controlar*

- El agente es el jugador virtual a entrenar.
- El environment es el "ambiente" que determina la naturaleza de la partida (número de jugadores, tipo de NPC, etc.). Para efectos prácticos, es una descripción de las reglas/especificaciones de la partida.
- El espacio de estados consiste en una tupla que contiene: La mano del agente, el número total de dados en juego, la última apuesta hecha en la **ronda**.
- El espacio de acciones es: $\{\text{Dudar}\} \cup \{\text{Calzar}\} \cup \bigcup_{n \in [10], p \in \{\text{Pintas}\}} \{n\} \times \{p\}$
- Modelamos las reward como 1 si se gana en ese *paso*, -1 si se pierde y 0 si sólo se sube la apuesta.

Objetivo del RL

Queremos encontrar la **política óptima** π^*, que maximice la *recompensa descontada* (de factor γ). Es decir, queremos:

$$\pi^* \in \arg\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \right]$$

Objetivo del RL

Queremos encontrar la **política óptima** π^*, que maximice la *recompensa descontada* (de factor γ). Es decir, queremos:

$$\pi^* \in \arg\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \right]$$

Programación Dinámica

Bajo **conocimiento perfecto** de la *función de transición* del *ambiente*, podemos usar **programación dinámica** para resolver el problema de forma exacta. Basta *guardar en memoria* las funciones de *valor* y de *acción-valor*:

$$v_\pi(s) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s \right] \quad \textbf{y} \quad q_\pi(s,a) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s, A_0 = a \right]$$

Se generan las **ecuaciones de Bellman** y ¡podemos encontrar π^* !

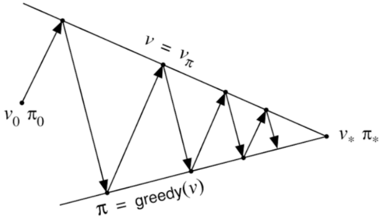
Generalized Policy Iteration (GPI)

- **[Value Iteration]** Dada una π **fija**, la **ecuación de Bellman** se usa como algoritmo para *iterativamente mejorar la estimación de la función valor* v_π (y también de q_π) desde cualquier estimación v_0 inicial.
- **[Policy Iteration]** Teniendo una función valor v_π (y la q_π asociada), la política determinista $\pi'(s) = \arg\max_a[q_\pi(s,a)]$, **siempre** es **uniformemente mejor** que π ($\forall s \in \mathscr{S},\ v_\pi(s) \leq v_{\pi'}(s)$).

Generalized Policy Iteration (GPI)

- **[Value Iteration]** Dada una π **fija**, la **ecuación de Bellman** se usa como algoritmo para *iterativamente mejorar la estimación de la función valor* v_π (y también de q_π) desde cualquier estimación v_0 inicial.
- **[Policy Iteration]** Teniendo una función valor v_π (y la q_π asociada), la política determinista $\pi'(s) = \arg\max_a [q_\pi(s,a)]$, **siempre** es **uniformemente mejor** que π ($\forall s \in \mathscr{S},\ v_\pi(s) \leq v_{\pi'}(s)$).

Se demuestra que, partiendo desde una v_0 y π_0 arbitraria; la iteración de: *aproximar la función valor real* v_π y *encontrar una policy* π' *golosa c/r a* v_π; **converge a la política óptima** π^* (y a su función de valor v_*)



Información Imperfecta

NO podemos usar las ecuaciones de Bellman (desconocemos $p(s', r|s, a)$), pero si disponemos de M *trayectorias* $\{\{(s_t^m, a_t^m, r_t^m)\}_{t=1}^{T_m}\}_{m=1}^{M}$ estimamos:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| S_0 = s, A_0 = a\right] \approx \frac{1}{C_{(s,a)}}\sum_{m=1}^{M}\sum_{\tau=0}^{T_m-1}\mathbb{1}_{\substack{s_\tau^m=s\\ a_\tau^m=a}}\left(\sum_{k=0}^{\infty}\gamma^k r_{\tau+k}^m\right)$$

Información Imperfecta

NO podemos usar las ecuaciones de Bellman (desconocemos $p(s', r|s, a)$), pero si disponemos de M *trayectorias* $\{\{(s_t^m, a_t^m, r_t^m)\}_{t=1}^{T_m}\}_{m=1}^{M}$ estimamos:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| S_0 = s, A_0 = a\right] \approx \frac{1}{C_{(s,a)}}\sum_{m=1}^{M}\sum_{\tau=0}^{T_m-1}\mathbb{1}_{\substack{s_\tau^m=s\\ a_\tau^m=a}}\left(\sum_{k=0}^{\infty}\gamma^k r_{\tau+k}^m\right)$$

Esto se transforma en una *regla de actualización* (a lo largo de la trayectoria):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \frac{1}{C(s_t^m)}\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k}^m - Q(s_t^m, a_t^m)\right)$$

Con que, tras procesar los M *episodios* (con M grande), tendremos un **buen estimador Q de la función** q_π (y por ende, también de v_π)

Información Imperfecta

NO podemos usar las ecuaciones de Bellman (desconocemos $p(s', r|s, a)$), pero si disponemos de M *trayectorias* $\{\{(s_t^m, a_t^m, r_t^m)\}_{t=1}^{T_m}\}_{m=1}^{M}$ estimamos:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s, A_0 = a\right] \approx \frac{1}{C_{(s,a)}} \sum_{m=1}^{M} \sum_{\tau=0}^{T_m-1} \mathbb{1}_{\substack{s_\tau^m = s \\ a_\tau^m = a}} \left(\sum_{k=0}^{\infty} \gamma^k r_{\tau+k}^m\right)$$

Esto se transforma en una *regla de actualización* (a lo largo de la trayectoria):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k}^m - Q(s_t^m, a_t^m)\right)$$

Con que, tras procesar los M *episodios* (con M grande), tendremos un **buen estimador Q de la función** q_π (y por ende, también de v_π)

Información Imperfecta

NO podemos usar las ecuaciones de Bellman (desconocemos $p(s', r|s, a)$), pero si disponemos de M *trayectorias* $\{\{(s_t^m, a_t^m, r_t^m)\}_{t=1}^{T_m}\}_{m=1}^{M}$ estimamos:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| S_0 = s, A_0 = a\right] \approx \frac{1}{C_{(s,a)}}\sum_{m=1}^{M}\sum_{\tau=0}^{T_m-1}\mathbb{1}_{\substack{s_\tau^m=s\\ a_\tau^m=a}}\left(\sum_{k=0}^{\infty}\gamma^k r_{\tau+k}^m\right)$$

Esto se transforma en una *regla de actualización* (a lo largo de la trayectoria):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k}^m - Q(s_t^m, a_t^m)\right)$$

Con que, tras procesar los M *episodios* (con M grande), tendremos un **buen estimador Q de la función** q_π (y por ende, también de v_π)

Información Imperfecta

NO podemos usar las ecuaciones de Bellman (desconocemos $p(s', r|s, a)$), pero si disponemos de M *trayectorias* $\{\{(s_t^m, a_t^m, r_t^m)\}_{t=1}^{T_m}\}_{m=1}^{M}$ estimamos:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s, A_0 = a\right] \approx \frac{1}{C_{(s,a)}} \sum_{m=1}^{M} \sum_{\tau=0}^{T_m-1} \mathbb{1}_{\substack{s_\tau^m = s \\ a_\tau^m = a}} \left(\sum_{k=0}^{\infty} \gamma^k r_{\tau+k}^m\right)$$

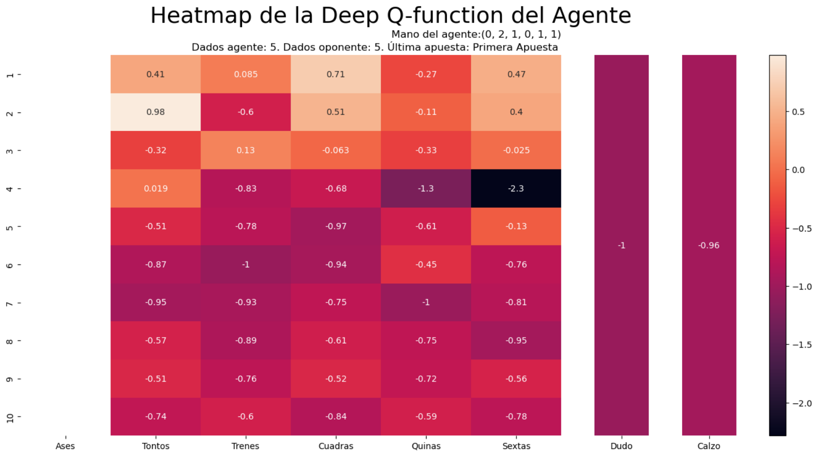
Esto se transforma en una *regla de actualización* (a lo largo de la trayectoria):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k}^m - Q(s_t^m, a_t^m)\right)$$

Con que, tras procesar los M *episodios* (con M grande), tendremos un **buen estimador Q de la función q_π** (y por ende, también de v_π)

Aplicamos **Generalized Policy Iteration** como antes, pero elegimos políticas **aleatorias, ε-*greedy*, que permiten la exploración**: En cierto estado, con probabilidad $1 - \varepsilon$ elegimos la acción *golosa* (y con ε, es al azar).

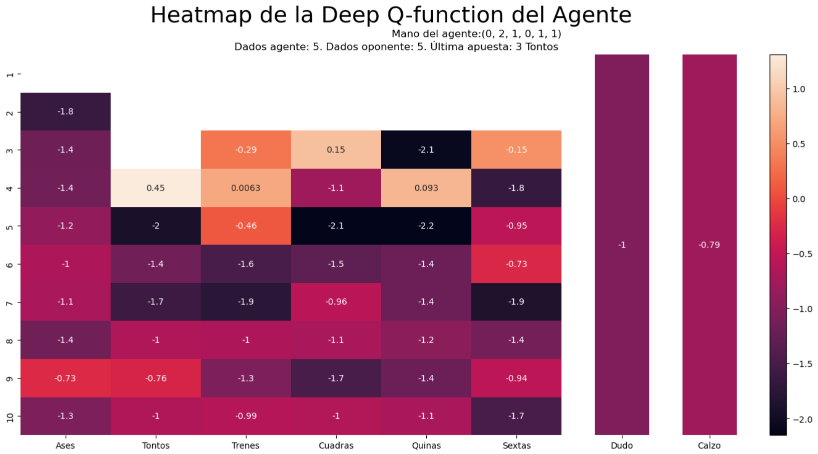
El Agente Comienza la Partida:



Viendo su *Q-Table*, decide jugar **2 Tontos**.

Ejemplo de la Evolución de las Q-Tables

El Oponente Responde **3 Tontos**:



Viendo su *Q-Table*, el Agente responde **4 Tontos**.

El Oponente **Duda** de manera Incorrecta. Las manos eran:

- **[Agente]: 2 Tontos**, 1 Tren, 1 Quina y 1 Sexta.
- **[Oponente]: 1 As**, **1 Tonto**, 1 Tren, 1 Quina y 1 Sexta.

Por lo tanto, sí habían **4 Tontos**.

SARSA (1-step Temporal Difference Learning)

En vez de *esperar a simular toda una trayectoria* para actualizar el estimador Q, se hace en **cada paso** (estimando la *cola* de la suma con Q):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha \left((r_t^m + \gamma Q(s_{t+1}^m, a_{t+1}^m)) - Q(s_t^m, a_t^m) \right)$$

SARSA (1-step Temporal Difference Learning)

En vez de *esperar a simular toda una trayectoria* para actualizar el estimador Q, se hace en **cada paso** (estimando la *cola* de la suma con Q):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha \left((r_t^m + \gamma Q(s_{t+1}^m, a_{t+1}^m)) - Q(s_t^m, a_t^m) \right)$$

Q-Learning

Cambiamos la regla de actualización, para *mejorar el estimador del target*:

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha \left((r_t^m + \gamma \max_a Q(s_{t+1}^m, a)) - Q(s_t^m, a_t^m) \right)$$

SARSA (1-step Temporal Difference Learning)

En vez de *esperar a simular toda una trayectoria* para actualizar el estimador Q, se hace en **cada paso** (estimando la *cola* de la suma con Q):

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha \left((r_t^m + \gamma Q(s_{t+1}^m, a_{t+1}^m)) - Q(s_t^m, a_t^m) \right)$$

Q-Learning

Cambiamos la regla de actualización, para *mejorar el estimador del target*:

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha \left((r_t^m + \gamma \max_a Q(s_{t+1}^m, a)) - Q(s_t^m, a_t^m) \right)$$

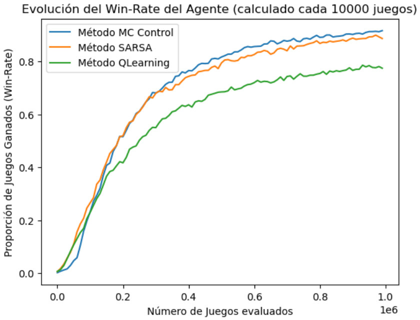
Deep Q-Learning

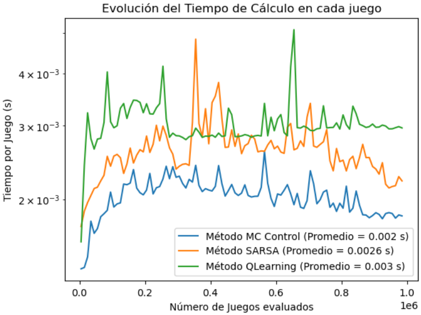
Con muchos *estados y acciones*, **guardar** Q **como una tabla** es **muy ineficiente**. Podemos aproximar la *función* $Q : \mathscr{S} \to \mathbb{R}^{\mathscr{A}}$ con una **red neuronal**: $Q_\theta : \mathscr{S} \to \mathbb{R}^{\mathscr{A}}$ que entrenaremos con SGD en la *memoria*.

Resultados

Comparación de Modelos

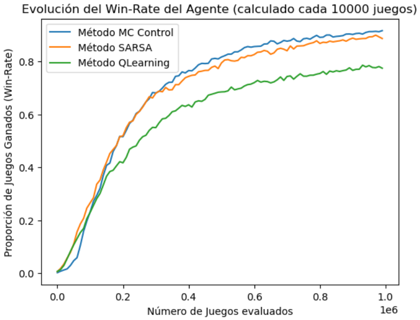
Evolución de la Performance de los modelos

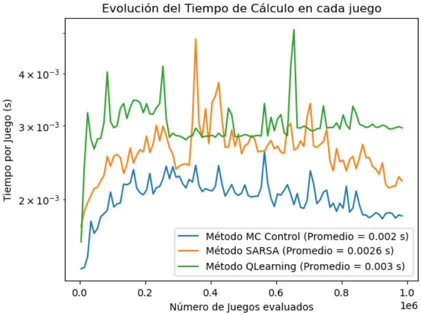




Comparación de Modelos

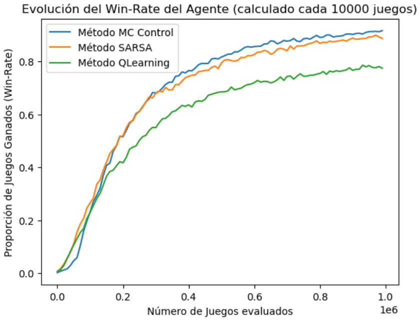
Evolución de la Performance de los modelos

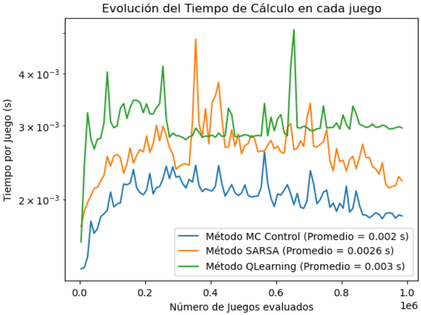




Estadísticas	WR Final (10^4 juegos)	Tiempo Medio por Juego
MC Control	0,82	2,030 ms
SARSA	0,91	2,624 ms
QLearning	0,79	3,036 ms

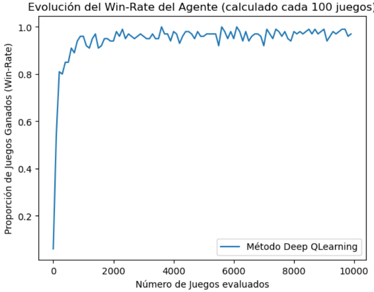
Evolución de la Performance de los modelos

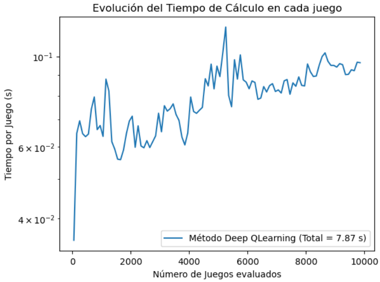




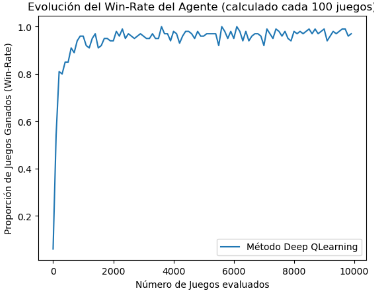
- QLearning en el *corto plazo* es el más efectivo (pero se *sesga* a la larga).
- En el largo plazo, SARSA o MC Control se comportaron mejor.
- Requieren demasiadas muestras para lograr rendimientos *aceptables*.
  - Espacio de estados demasiado extenso para memorizar.
  - En consecuencia, altos tiempos de entrenamiento.

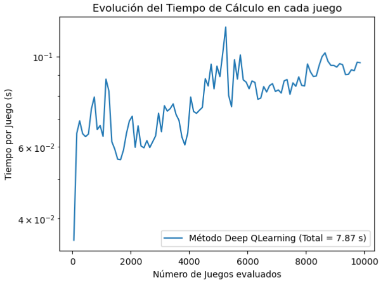
Evolución de la Performance del modelo Deep QLearning





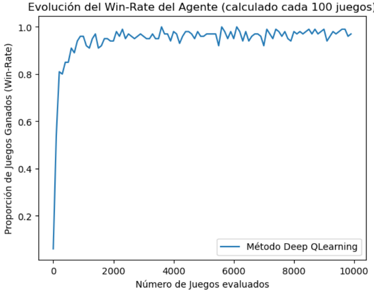
Evolución de la Performance del modelo Deep QLearning

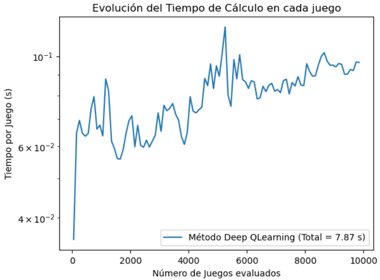




Estadísticas	WR Final (10^4 juegos)	Tiempo Medio por Juego
MC Control	0,82	2,030 ms
SARSA	0,91	2,624 ms
QLearning	0,79	3,036 ms
Deep Q-Learning	0,97	79,593 ms

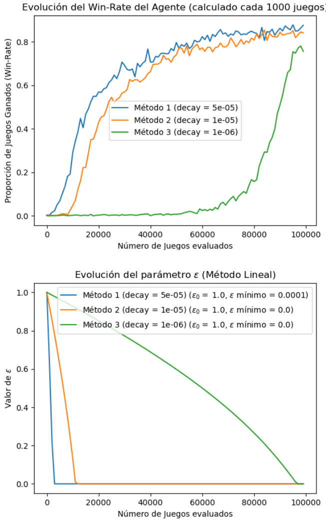
Evolución de la Performance del modelo Deep QLearning





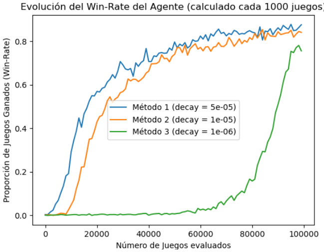
- Aumento muy veloz del winrate.
- La componente profunda del algoritmo aumenta los costos temporales (en global, todas las técnicas son *comparables*).
- No se exploró mayormente en arquitecturas de NN subyacentes.
- Se podría extender la definición de *estado* sin mayores problemas.

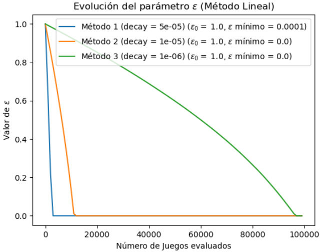
Lineal, $\varepsilon_0 = 1$



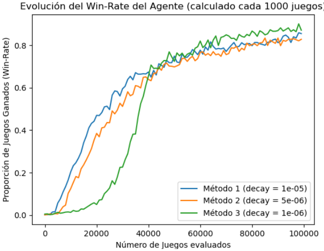
Parámetro de Exploración

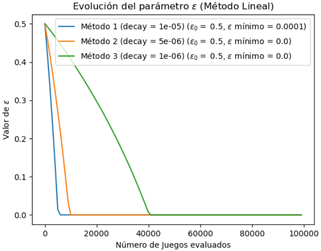
Lineal, $\varepsilon_0 = 1$



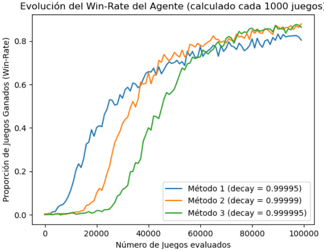


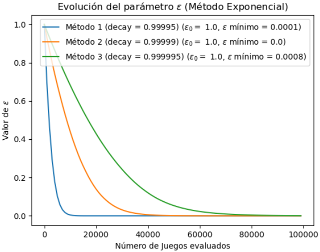
Lineal, $\varepsilon_0 = 0{,}5$





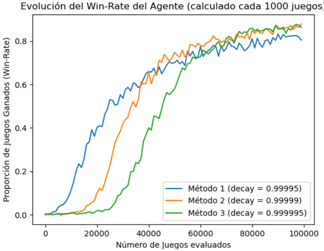
Exponencial, $\varepsilon_0 = 1$

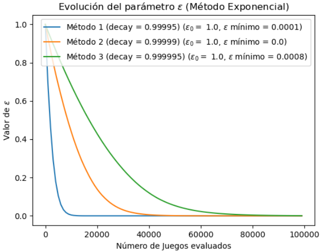




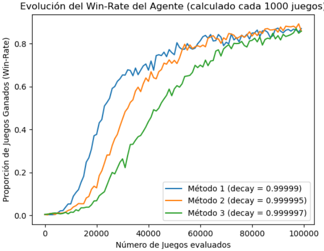
Parámetro de Exploración

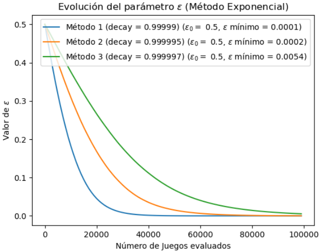
Exponencial, $\varepsilon_0 = 1$





Exponencial, $\varepsilon_0 = 0{,}5$





Tendencias Generales

- **Exploration-Explotation Trade-Off**
  - Muchísima exploración *ralentiza* el aprendizaje.
  - Muy poca exploración *sesga* el aprendizaje.
- No parece haber mayor diferencia entre un decaimiento *Lineal* y uno *Exponencial*, salvo quizás en la *suavidad* de las curvas de aprendizaje.
- En general, los resultados no son dramáticamente diferentes en términos de Win-Rate (tras 100 000 juegos simulados):

$$\textbf{[Lineal]:}\ 0{,}89; 0{,}86; 0{,}76 \mid 0{,}81; 0{,}78; 0{,}89$$

$$\textbf{[Exponencial]:}\ 0{,}83; 0{,}87; 0{,}88 \mid 0{,}87; 0{,}91; 0{,}82$$

Conclusiones

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Exploration-Exploitation Trade-Off

- Conviene fijar un ε que decaiga a una tasa *intermedia*.

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Exploration-Exploitation Trade-Off

- Conviene fijar un ε que decaiga a una tasa *intermedia*.

Próximas Aristas

- Técnicas de *Importance Sampling* para implementar métodos *off-policy*.

Conclusiones

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Exploration-Exploitation Trade-Off

- Conviene fijar un ε que decaiga a una tasa *intermedia*.

Próximas Aristas

- Técnicas de *Importance Sampling* para implementar métodos *off-policy*.
- Más RL por explorar: *Policy Gradients*, *Actor Critic Methods*, etc.

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Exploration-Exploitation Trade-Off

- Conviene fijar un ε que decaiga a una tasa *intermedia*.

Próximas Aristas

- Técnicas de *Importance Sampling* para implementar métodos *off-policy*.
- Más RL por explorar: *Policy Gradients*, *Actor Critic Methods*, etc.
- Complejización del juego: *más jugadores*, *estados más complejos*, etc.

Conclusiones

Métodos de Entrenamiento

- Métodos Tabulares son **limitados** cuando $|\mathscr{S} \times \mathscr{A}|$ es muy grande.
- Métodos *profundos* son mucho más efectivos, pero *menos explicables* y más complejos de simular.

Exploration-Exploitation Trade-Off

- Conviene fijar un ε que decaiga a una tasa *intermedia*.

Próximas Aristas

- Técnicas de *Importance Sampling* para implementar métodos *off-policy*.
- Más RL por explorar: *Policy Gradients*, *Actor Critic Methods*, etc.
- Complejización del juego: *más jugadores*, *estados más complejos*, etc.

¡El Reinforcement Learning es una herramienta muy potente !

¡Gracias por su atención!

Reinforcement Learning para el *Cacho*

Javier Maass Martínez y Juan Pablo Sepúlveda

MA4402 Simulación Estocástica

7 de diciembre de 2022