Evaluación de políticas bajo ruido Markoviano mediante el algoritmo de Online Bootstrap Inference

Ana María Patrón Piñerez

10 de agosto de 2023

Introducción

- El objetivo es encontrar la función valor para una política dada, en casos donde el ambiente provee información insuficiente o sea de dimensiones grandes.
- La estrategia es aproximar linealmente la función valor [Sutton and Barto, 2018].
- Con aproximar linealmente una función nos referimos a aplicar Aproximación Lineal Estocástica (LSA). En el caso estándar, el ruido es una diferencia de martingalas; pero con nuestra estrategia, esto se incumple y ahora el ruido es Markoviano. [Ramprasad et al., 2022, Liang, 2010]

- Los métodos de Diferencias temporales (TD), para el caso on-policy, y de Gradiente de Diferencias Temporales (GTD), para el caso off-policy, son métodos clásicos de LSA para hacer evaluación de políticas bajo nuestra estrategia. Sin embargo, sus resultados son sesgados [Sutton and Barto, 2018, Maei, 2011].
- El algoritmo de Online Bootstrap plantea mejorar esto. Consiste en agregarle a los métodos clásicos un bootstrap estadístico, en el que para cada período de tiempo se realizan B iteraciones penalizadas del parámetro de LSA [Ramprasad et al., 2022].
- A partir de estas, se construyen intervalos de confianza para la función valor.

Contribución

- Reconstruir el contexto teórico en el que se construye el algoritmo de Online Bootstrap y compararlo con el escenario clásico.
- Robustecer el análisis para caso off-policy, usando Diferencias Temporales con corrección de Gradiente (TDC).
- Estudiar computacionalmente el desempeño del Online Bootstrap respecto a la política escogida.

Índice

Marco General

La Aproximación Lineal Estocástica (LSA) es un método iterativo para encontrar las raíces de una función lineal desconocida $f(x)$, a partir de unas observaciones $g(x_t, \eta_{t+1})$, donde el ruido η_{t+1} perturba las observaciones y se cumple que $f(x) = E[g(x, \eta)]$

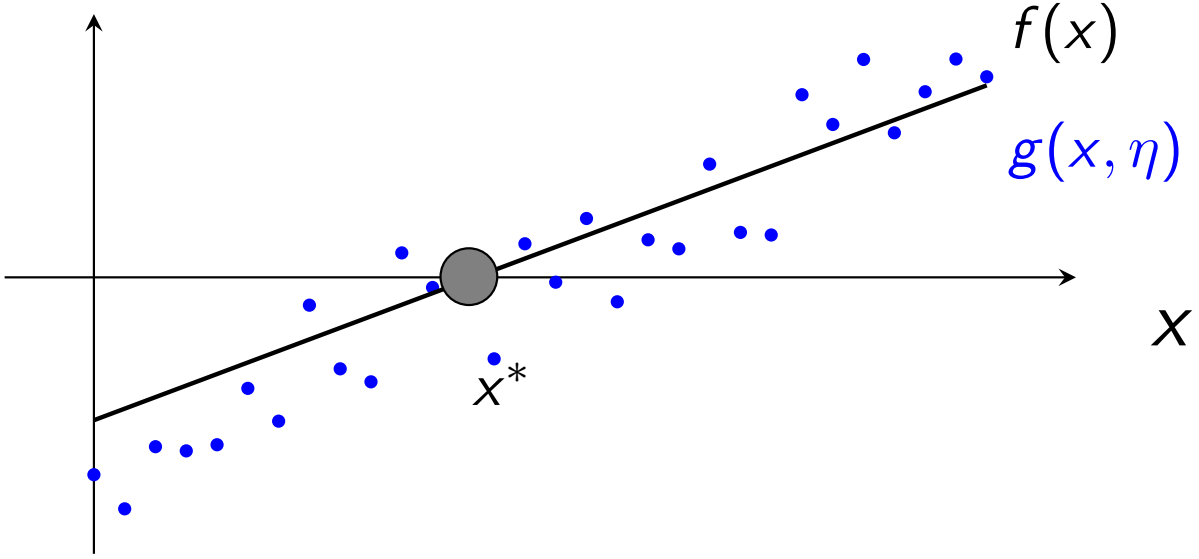


Figura: Representación gráfica de *LSA*.

Notacionalmente, tomamos

$$f(x_t) = \overline{A}x_t - \overline{b}, \qquad g(x_t, \eta_{t+1}) = \tilde{A}_{t+1}x_t - \tilde{b}_{t+1}$$

donde

- $\overline{A}, \tilde{A}_t \in \mathbb{R}^{d\times d}$, $E[\tilde{A}_t] = \overline{A}$
- $\overline{b}, \tilde{b}_t, x_t \in \mathbb{R}^d$, $E[\tilde{b}_t] = \overline{b}$

Y buscamos encontrar x^* que soluciona la ecuación $\overline{A}x = \overline{b}$

Iteración de LSA (Robbins Monro)

Formalmente, la actualización está dada por:

$$x_{t+1} = x_t + \alpha_{t+1}[\tilde{A}_t x_t - \tilde{b}_t] \tag{1}$$

- x_t es el parámetro de LSA
- $\tilde{A}_t x_t - \tilde{b}_t$ es una observación con ruido de $\overline{A}x_t - \overline{b}$, donde el ruido $\{e_{t+1}\} = \{(\tilde{A}_{t+1} - \overline{A})x_t - (\tilde{b}_{t+1} - \overline{b})\}$ es una diferencia de martingalas
- α_{t+1} es un escalar positivo (step-size function en la literatura)

Teoría asintótica para LSA

Teorema 1 (Convergencia casi siempre del algoritmo de Robbins Monro)

1. *Sea* $e_{t+1} = (\tilde{A}_{t+1} - \overline{A})x_t - (\tilde{b}_{t+1} - \overline{b})$, *donde* $\{e_t\}_{t\geq 0}$ *es una sucesión de diferencia de martingalas; es decir,* $\mathbb{E}[e_{t+1}|\mathcal{F}_t] = 0$, *donde* $\mathcal{F}_t$ *es el conjunto de información.*

2. *La sucesión* $\{\tilde{A}_{t+1}x_t - \tilde{b}_{t+1}\}_{t\geq 0} \in L^2$ *con* $\mathbb{E}[||\tilde{A}_{t+1}x_t - \tilde{b}_{t+1}||^2|\mathcal{F}_t] \leq K(1 + ||x_t||^2)$ *c.s para algún* $K > 0$

3. $\sum_{t=0}^{\infty} \alpha_t = \infty$ *y* $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$

4. *La sucesión* $\{x_t\}_{t\geq 0}$ *está acotada,* $sup_{t\geq 0}||x_t||_2 < \infty$

entonces $lim_{t\to\infty} x_t = x^*$ *c.s*

Teoría asintótica para LSA

Teorema 2 (Normalidad asintótica de Robbins Monro)

Suponga que las condiciones 1-4 del teorema 1 valen. Además que

- $Q = lim_{t\to\infty}\mathbb{E}[e_t e_t^T]$
- $\overline{A}$ *es una matriz Hurwitz, es decir que todos sus valores propios tiene parte real estrictamente negativa.*

entonces

$$\sqrt{t}(x_t - x^*) \xrightarrow{d} N(0, \overline{A}^{-1} Q (\overline{A}^{-1})^T)$$

Corolario 3 (Polyak-Ruppert)

La convergencia casi siempre y la normalidad asintótica también valen para la iteración del estimador promediado $\overline{x}_t = \frac{1}{t}\sum_{i=1}^{t} x_i$.

Construcción de LSA con ruido Markoviano

Definición 4 (Cadena de Markov)

Una cadena de Markov es un proceso estocástico discreto $\{X_t\} = \{X_1, X_2, ..., \}$ tal que

$$\mathbb{P}[X_{t+1}|X_t] = \mathbb{P}[X_{t+1}|X_1, ..., X_t]$$

Su kernel de transición $\mathcal{P}$ es una matriz con las probabilidades de transición de un evento a otro.

Teorema 5

Una cadena ergódica de Markov tiene distribución límite μ, que es única y es igual a su distribución estacionaria.

Definición 6 (LSA con ruido markoviano)

Sea $\{\tilde{A}(X_t), \tilde{b}(X_t)\}_{t\geq 1}$ una secuencia de observaciones, donde $\{X_t\}$ es una cadena ergódica de Markov con espacio de estados $\mathcal{X}$ y distribución estacionaria μ. Además $\tilde{A} : \mathcal{X} \to \mathbb{R}^{d\times d}, \tilde{b} : \mathcal{X} \to \mathbb{R}^d$ y $\mathbb{E}_\mu[\tilde{A}(X_{t+1})] = \overline{A}, \mathbb{E}_\mu[\tilde{b}(X_{t+1})] = \overline{b}$. Entonces la actualización dada por

$$x_{t+1} = x_t + \alpha_{t+1}(\tilde{A}(X_{t+1})x_t - \tilde{b}(X_{t+1})) \tag{2}$$

corresponde al algoritmo de LSA con ruido Markoviano, donde $x \in \mathbb{R}^d$ es el parámetro de LSA y $\{\alpha_t\}_{t\geq 1}$ es una sucesión de pasos decrecientes, dada por $\alpha_t = \frac{\alpha_0}{t^\eta}$ para algún $\alpha_0 > 0, \eta \in (\frac{1}{2}, 1)$

Teoría asintótica para LSA con ruido Markoviano

- Ahora, $\epsilon_{t+1} = (\tilde{A}(X_{t+1}) - \overline{A})x_t - (\tilde{b}(X_{t+1}) - \overline{b})$ y no se puede garantizar que sea una diferencia de martingalas
- Pero podemos escribir $\epsilon_t = e_t + \nu_t + \zeta_t$, donde e_t es la diferencia de martingalas de LSA en el caso general, ν_t y ζ_t son términos residuales decrecientes [Liang, 2010].
- Si para la definición 6, además se tiene que $\overline{A}$ es de rango completo, Hurwitz y que existen A_{max}, b_{max} tales que $sup_{x\in\mathcal{X}}||\tilde{A}(x)||_F \leq A_{max}, \qquad sup_{x\in\mathcal{X}}||\tilde{b}(x)||_2 \leq b_{max},$ entonces:

Proposición 1

La convergencia casi siempre y la normalidad asintótica valen para la estimación de LSA con ruido Markoviano, en particular, $\sqrt{t}(x_t - x^*) \xrightarrow{d} N(0, \overline{A}^{-1} Q (\overline{A}^{-1})^T)$ *donde* $Q = lim_{t\to\infty} \mathbb{E}[e_t e_t^T]$

- Aún así, en la práctica no podemos obtener valores para cada componente de $\epsilon_t \Rightarrow Q$ es desconocido.
- Una alternativa es usar el algoritmo de Online Bootstrap Inference. [Ramprasad et al., 2022]

El algoritmo de Online Bootstrap Inference

- Su aporte es adicionar una iteración perturbada paralela a de la iteración promedio de LSA:

$$\hat{x}^b_{t+1} = \hat{x}^b + \alpha_{t+1} W^b_{t+1}(\tilde{A}(X_{t+1})\hat{x}^b - \tilde{b}(X_{t+1}))$$
$$\overline{\hat{x}}^b_{t+1} = \frac{1}{t+1}\sum_{i=1}^{t+1} \hat{x}^b_i \quad (3)$$

  donde W^b_t son variables aleatorias acotadas i.i.d con media y varianza 1.

- ¿La razón? se puede mostrar que $\sqrt{t}(\overline{\hat{x}}_t - \overline{x}_t) \to N(0, \overline{A}^{-1} Q (\overline{A}^{-1})^T)$, $t \to \infty$ (Teo. 4.2 [Ramprasad et al., 2022]) ... ¡Es la misma de $\sqrt{t}(\overline{x}_t - x^*)$!
- Podemos aproximar la distribución de $\sqrt{t}(\overline{\hat{x}}_t - \overline{x}_t)$ con B muestras bootstraps de $\overline{\hat{x}}_t - \overline{x}_t$ para cada t, es decir (3).

Codificación del algoritmo Online Bootstrap

Algoritmo 1 Online Bootstrap Inference

Input: Número de muestras bootstrap B, $\alpha_0, \eta \in (\frac{1}{2}, 1)$ y estimadores iniciales $x_0 = \hat{x}_0^b,\ b = 1, ..., B$

for $t = 0, 1, 2, ...$ **do**
  Observar $\tilde{A}(X_{t+1}),\ \tilde{b}(X_{t+1})$
  Computar $\alpha_{t+1} = \alpha_0 (t+1)^{-\eta}$
  Actualizar $x_{t+1} \leftarrow x_t + \alpha_{t+1}(\tilde{A}(X_{t+1}))x_t - \tilde{b}(X_{t+1}))$
  Actualizar $\overline{x}_{t+1} \leftarrow \frac{1}{1+t}(t\, x_t + x_{t+1})$
  for $b = 1, 2, ..., B$ **do**
    Actualizar $\hat{x}_{t+1}^b \leftarrow \hat{x}_t^b + \alpha_{t+1} W_{t+1}^b (\tilde{A}(X_{t+1})\hat{x}_t^b - \tilde{b}(X_{t+1}))$
    Actualizar $\overline{\hat{x}}_{t+1}^b \leftarrow \frac{1}{1+t}(t\, \overline{\hat{x}}_t^b + \hat{x}_{t+1}^b)$
  end for
end for

Output Estimadores bootstrap $\{\overline{\hat{x}}_{t+1}^b\}_{b=1}^B$

Evaluación de políticas en RL

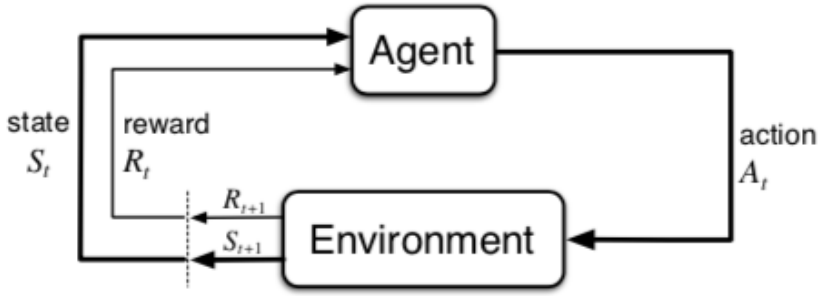


Figura: Figura tomada de [Sutton and Barto, 2018]

- Durante cada $t = 1, 2, ..$ un agente se encuentra en un estado s_t.
- Toma una acción a_t, de las disponibles para ese s_t.
- Dependiendo de la configuración del ambiente, recibe una recompensa r_{t+1} y llega a un estado s_{t+1}.

MDP y MRP

- Un **Proceso de Decisión de Markov (MDP)** es formalizar el ambiente de RL. Se denota $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ donde $\mathcal{S}, \mathcal{A}$ son los espacios de estados y acciones, respectivamente; $\mathcal{P}$ es el kernel de transición y $\mathcal{R}$ la función de recompensas.
- Una política π es un vector de probabilidades que para cada s_t, determina las probabilidades de tomar cada una de las acciones a_t disponibles.
- Un MDP junto con una política π inducen un **Proceso de Recompensa de Markov (MRP)**, que se denota $\mathcal{M}^\pi = (\mathcal{M}, \pi)$.

Función valor

La función valor asociada a π es la suma descontada de recompensas esperadas partiendo de s y siguiendo π:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}^{\pi}(s_t) | s_0 = s, \pi], \quad s \in \mathcal{S} \\ &= \mathcal{R}^{\pi}(s) + \gamma \mathcal{P}^{\pi}(s) V^{\pi}(s) \\ &:= TV^{\pi}(s) \quad \text{(Ecuación de Bellman)} \end{aligned}$$

donde $\mathcal{P}^{\pi}(s), \mathcal{R}^{\pi}(s)$ son matrices calculables. Como $TV^{\pi}(s)$ es una contracción, por teorema de punto fijo de Banach $V^{\pi}(s)$ es la solución única. Así:

$$V^{\pi}(s) = (\mathbb{I} - \gamma \mathcal{P}(s)^{\pi})^{-1} \mathcal{R}(s)^{\pi}. \tag{4}$$

Un obstáculo

- En RL, es común que haya incertidumbre, eso implica desconocer $\mathcal{P}^\pi$.
- Aún si hubiese certidumbre, las dimensiones del ambiente pueden ser grandes y así las dimensiones de $\mathcal{P}^\pi o \mathcal{R}^\pi$
  $\Rightarrow$ ¡No se puede usar la ecuación de Bellman!
- Una alternativa es aproximar linealmente la función valor como $V_\theta(s) = \phi(s)^T\theta$, donde $\theta \in \mathbb{R}^d$ es el parámetro de LSA y $\phi(s)^T = [\phi_1, ..., \phi_d]$con $\phi_i : S \to \mathbb{R}$ features (o funciones base)
- Cuando se aplica LSA a RL con esa aproximación de $V(s)$ el ruido de LSA es Markoviano. Acá $\{X_t\} = \{(s_t, r_{t+1}, s_{t+1})\}_{t\geq 0}$.

Métodos clásicos de LSA aplicados a RL

I.Diferencias Temporales (TD)-On policy

La actualización de la función valor está dada por:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t[V_t(s_t) - \gamma V_t(s_{t+1}) - r_{t+1}]. \quad (5)$$

Aplicando nuestra aproximación lineal de la función valor (5) es equivalente a:

$$\theta_{t+1} = \theta_t + \alpha_{t+1}[\underbrace{\phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T}_{\tilde{A}(X_t)}\theta_t - \underbrace{r_{t+1}\phi(s_t)}_{\tilde{b}(X_t)}]. \quad (6)$$

Que es la iteración de LSA con:

- $\tilde{A}(X_t) = \phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T$ y $\tilde{b}(X_t) = r_{t+1}\phi(s_t)$.
- $\overline{A} = E_{\mu_\pi}[\tilde{A}(X_t)] = \Phi^T\Xi(I - \gamma P^\pi)\Phi$ y $\overline{b} = E_{\mu_\pi}[\tilde{b}(X_t)] = \Phi^T\Xi r^\pi$, donde μ_π es la distribución estacionaria de $\{X_t\}$ bajo la política π.

Problemas de TD en el escenario off-policy

Ahora:

- En el caso on-policy, las observaciones se generan con la misma política π a evaluar.
- Pero en el caso off-policy, las observaciones se generan con una política π^b(*behavior policy*) distinta a la evaluar (*target policy*).

Los problemas:

- TD no es un gradiente $\Rightarrow$ inestabilidad.
- Hay una desconexión, π^b puede seleccionar acciones que π nunca tomaría $\Rightarrow$ inestabilidad.

Solución:

- Métodos de Gradiente de Diferencia Temporales (GTD): GTD1, GTD2 y TDC [Maei, 2011].

II.Gradiente de Diferencias Temporales (GTD)-Off policy.

A. Derivación: usando gradiente descentente,

- GTD1 minimiza la norma esperada de la actualización de TD:

$$NEU(\theta) = \mathbb{E}_{\mu^\pi}[\delta_t(\theta)\phi(s_t)]^T \mathbb{E}_{\mu^\pi}[\delta_t(\theta)\phi(s_t)]$$

  donde $\delta_t(\theta) = r_{t+1} + \gamma\theta_t^T\phi(s_{t+1}) - \theta_t^T\phi(s_t)$ es el error de TD.

- GTD2 y TDC minimizan el error cuadrático medio de Bellman projectado:

$$MSBPE(\theta) = ||V_\theta - \Pi TV_\theta||_\Xi^2$$

  donde Π es el operador de proyección, Ξ matriz cuyos elementos en la diagonal son las entradas de μ^π.

- Al calcular el gradiente, se obtienen productos de valores esperados no necesariamente independientes $\Rightarrow$ se incluye una actualización auxiliar enlazada.
- Se incluye un ratio de Important Sampling $\rho_t = \frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}$ para corregir la discrepancia entre π, π^b.
- Formalmente, tome

$$A_t = \rho_t \phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T \text{ y } b_t = \rho_t r_{t+1}\phi(s_t)$$

como estimadores insesgados de $A = \Phi^T \Xi(I - \gamma P^\pi)\Phi$ y $b = \Phi^T \Xi r^\pi$, respectivamente; entonces los algoritmos GTD se escriben como:

GTD1:

$$\begin{aligned} y_{t+1} &= y_t + \alpha_{t+1}[b_t + A_t\theta_t - I_d y_t] \\ \theta_{t+1} &= \theta_t + \alpha_{t+1} A_t^T y_t \end{aligned} \tag{7}$$

GTD2:

$$\begin{aligned} y_{t+1} &= y_t + \alpha_{t+1}[b_t + A_t\theta_t - \phi(s_t)^T y_t \phi(s_t)] \\ \theta_{t+1} &= \theta_t + \alpha_{t+1} A_t^T y_t \end{aligned} \tag{8}$$

TDC:

$$\begin{aligned} y_{t+1} &= y_t + \lambda_t\alpha_{t+1}[b_t + A_t\theta_t - \phi(s_t)^T y_t \phi(s_t)] \\ \theta_{t+1} &= \theta_t + \alpha_{t+1}[b_t + A_t\theta_t - \gamma\rho_t\phi(s_{t+1})\phi(s_t)^T y_t] \end{aligned} \tag{9}$$

B.Compactificando como problema de LSA:

GTD1 y GTD2:

$$\tilde{A}_t = \begin{bmatrix} 0 & -A_t^T \\ A_t & M_t \end{bmatrix} \qquad \tilde{b}_t = \begin{bmatrix} 0 \\ b_t \end{bmatrix} \qquad \Theta_t = \begin{bmatrix} \theta_t \\ y_t \end{bmatrix} \tag{10}$$

donde $M_t = I_d$ para GTD1 y $M_t = \phi(s_t)\phi^T(s_t)$ para GTD2.

TDC:

$$\tilde{A}_t = \begin{bmatrix} A_t & (\rho M_t - A_t)^T \\ \lambda_t A_t & \lambda_t M_t \end{bmatrix} \qquad \tilde{b}_t = \begin{bmatrix} b_t \\ \lambda_t b_t \end{bmatrix} \qquad \Theta_t = \begin{bmatrix} \theta_t \\ y_t \end{bmatrix} \tag{11}$$

donde $M_t = \phi(s_t)\phi^T(s_t)$, λ_t proxy empírica de $\lambda > 0$, donde λ es tal que es mayor al negativo del mínimo valor propio de la matriz $(\Phi^T \Phi)^{-1} \frac{(A+A^T)}{2}$.

Algorítmicamente

Para un estado particular s; $V_{\overline{\theta}_t}(s) = \phi^T(s)\overline{\theta}_t$, donde $\overline{\theta}_t$ se obtiene de:

Algoritmo 2 Método clásico

Input: $\alpha_0, \eta \in (\frac{1}{2}, 1)$ y estimador inicial θ_0

for $t = 0, 1, 2, \ldots$ **do**

Observar $\tilde{A}(X_{t+1})$, $\tilde{b}(X_{t+1})$

Computar $\alpha_{t+1} = \alpha_0(t+1)^{-\eta}$

Actualizar $\theta_{t+1} \leftarrow \theta_t + \alpha_{t+1}(\tilde{A}(X_{t+1})\theta_t - \tilde{b}(X_{t+1}))$

Actualizar $\overline{\theta}_{t+1} \leftarrow \frac{1}{1+t}(t\,\theta_t + \theta_{t+1})$

end for

Output Estimador $\overline{\theta}_{t+1}$

Algoritmo de Online Boostratp aplicado a RL

1. Inicialmente, de la parte I, del algoritmo Online Bootstrap obteníamos $\{\bar{\hat{\theta}}^{b}_{t+1}\}_{b=1}^{B}$
2. Ajustando el algoritmo a nuestro escenario de RL, para cada s tenemos que $V_{\bar{\theta}_t}(s) = \phi^T(s)\bar{\theta}_t$. Así, ahora construimos

$$V^{\pi}_{\bar{\theta}_t}(\nu) = \int_{s\in\mathcal{S}} V_{\bar{\theta}_t}(s)\nu(ds)$$

3. Definimos los bootstraps de valores estimados como

$$\left\{ V^{\pi}_{\bar{\hat{\theta}}^{(b)}_t} - V^{\pi}_{\bar{\theta}_t} \right\}_{b=1}^{B} \tag{12}$$

4. Construimos intervalos de confianza.

- $IC_{cuantil}(V^{\pi}_{\theta_t}(\nu)) = [V^{\pi}_{\theta_t}(\nu) + q_{\frac{\alpha}{2}}, V^{\pi}_{\theta_t}(\nu) + q_{1-\frac{\alpha}{2}}]$, donde q_α es el $\alpha-$ésimo cuantil de la distribución (12).

- $IC_{SE}(V^{\pi}_{\theta_t}(\nu)) = [V^{\pi}_{\theta_t}(\nu) + z_{\frac{\alpha}{2}}\tilde{\sigma}, V^{\pi}_{\theta_t}(\nu) + z_{1-\frac{\alpha}{2}}\tilde{\sigma}]$, donde z hace referencia a la distribución normal y $\tilde{\sigma} = \sqrt{\frac{s^2}{B}}, s^2$ es la varianza de (12).

Experimentos

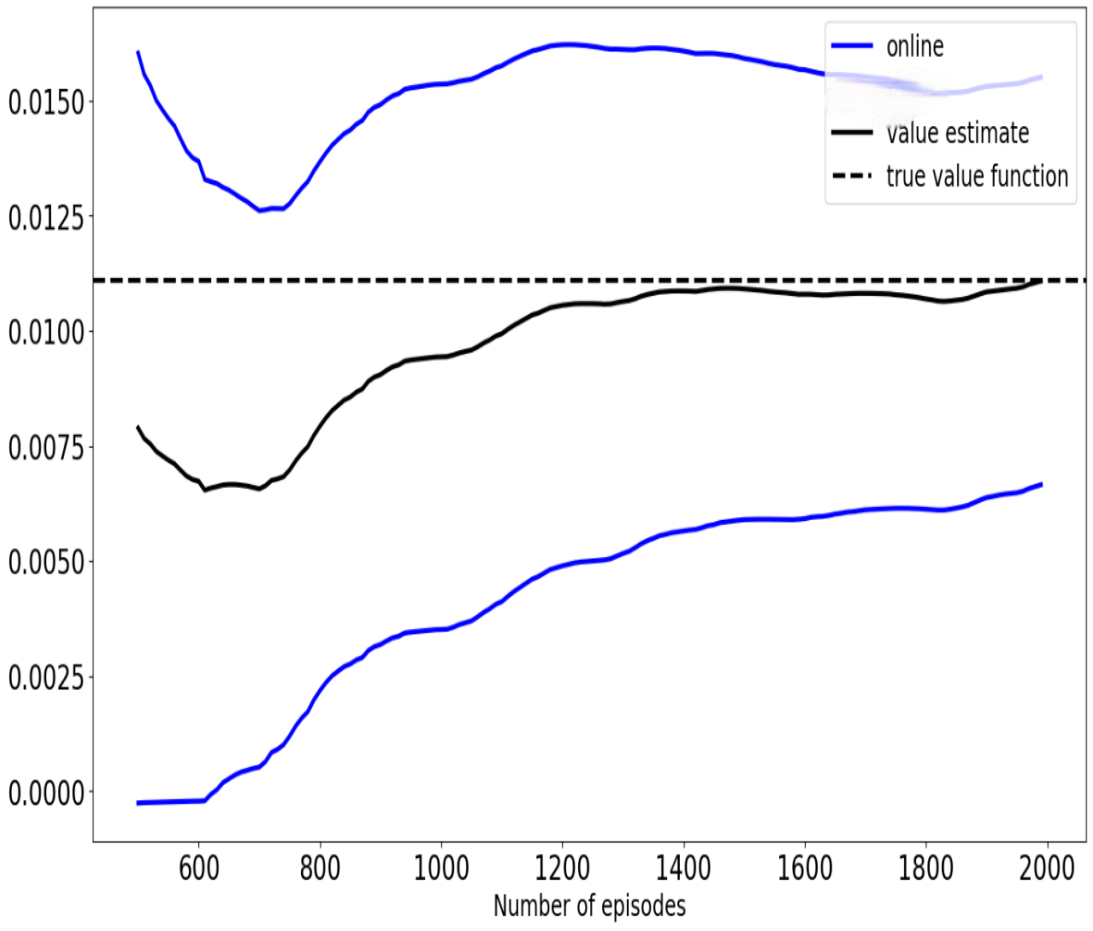
- Evaluamos el desempeño del Online Bootstrap respecto al método clásico y su sensibilidad respecto a a la política evaluada.
- El código que se emplea es principalmente construido por [Ramprasad et al., 2022]. Adaptamos este para el caso con TDC y para variaciones en la política.
- Los intervalos de confianza son al 95 %; $B = 200$, $\eta = \frac{2}{3}, \theta_0 = \vec{0}$. Además, $W_t \sim U[1 - \frac{1}{\sqrt{3}}, 1 + \frac{1}{\sqrt{3}}]$.
- Analizamos un caso on-policy y otro off-policy.

On policy

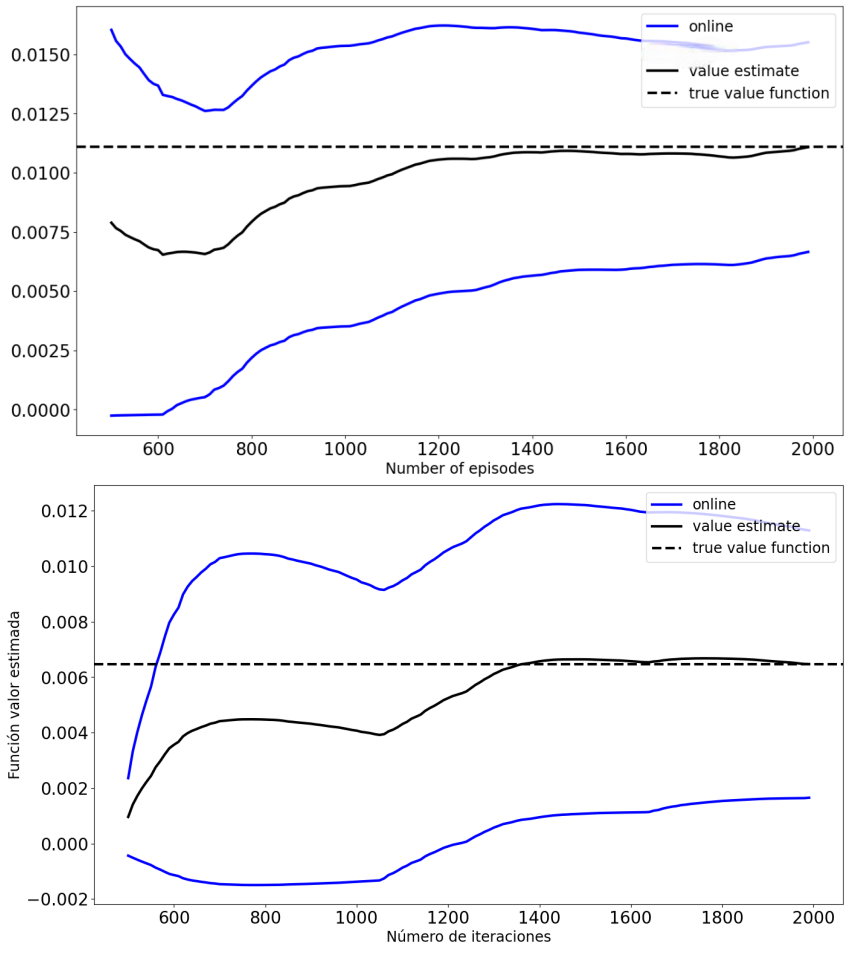
- Se usa el ambiente FrozenLake de OpenAI gym para generar el MDP.
- Hay 64 estados, cada estado es una cuadricula de una grilla 8×8; el espacio de acciones es $\mathcal{A} = \{izquierda, derecha, arriba, abajo\}$.
- El objetivo es llegar al último estado (casilla con coordenadas 8×8).
- La recompensa es 1 en el último estado y 0 de lo contrario.
- La política π, con la que se generan las observaciones, es una Q-política $\epsilon-$greedy con $\epsilon = 0{,}2$; y entre mayor sea ϵ, hay más aleatoriedad.

Online Bootstrap vs Método clásico

- Función valor verdadera $\approx 0{,}011 (\in$ IC siempre)
- TD se acerca a mayor número de iteraciones

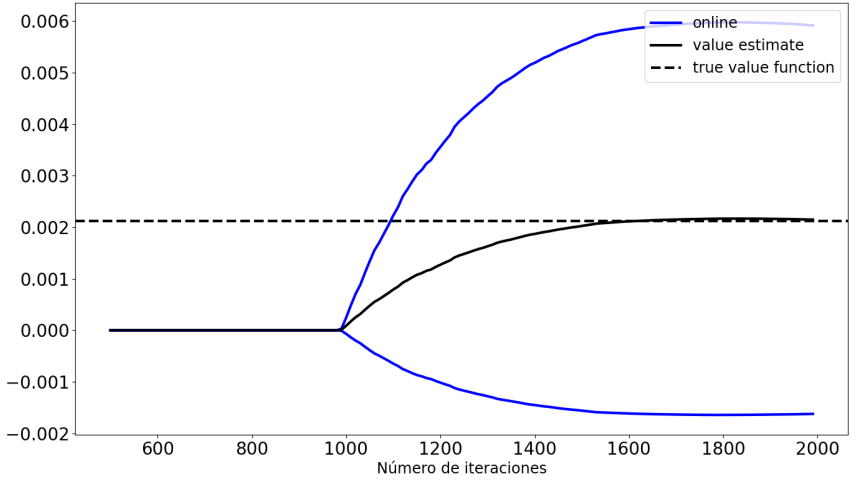


Sensibilidad frente a la política



$\epsilon = 0{,}3$

- Hipótesis: Mayor distancia entre el kernel de transición de X_t bajo π aleatoria y su distribución estacionaria se relaciona con convergencia más lenta

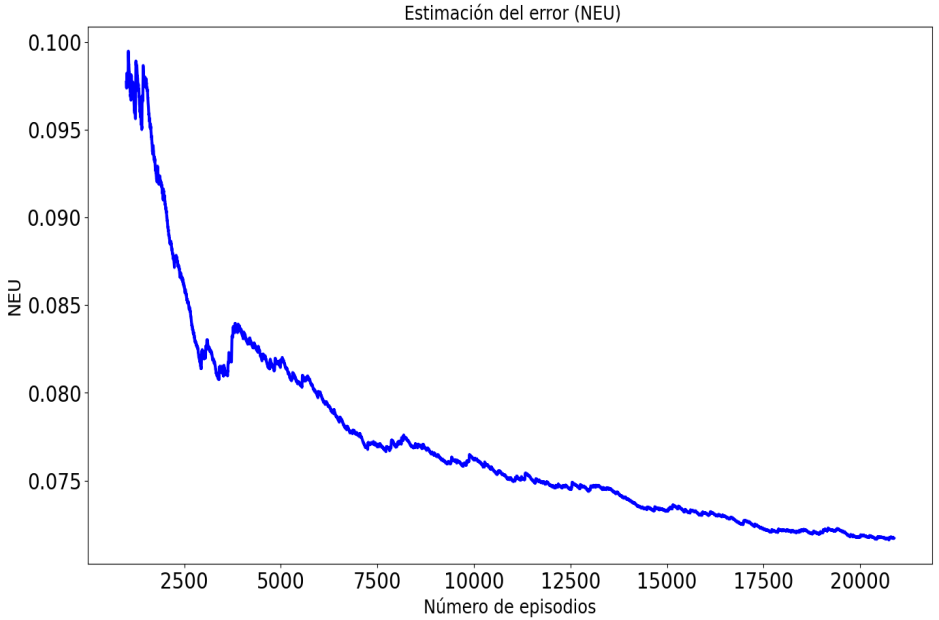


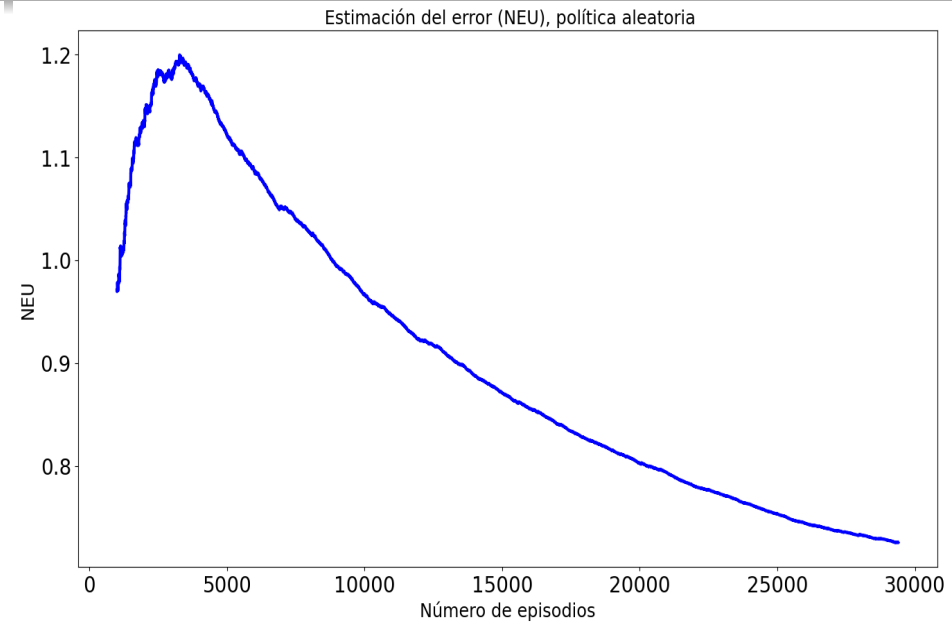
$\epsilon = 0{,}5$

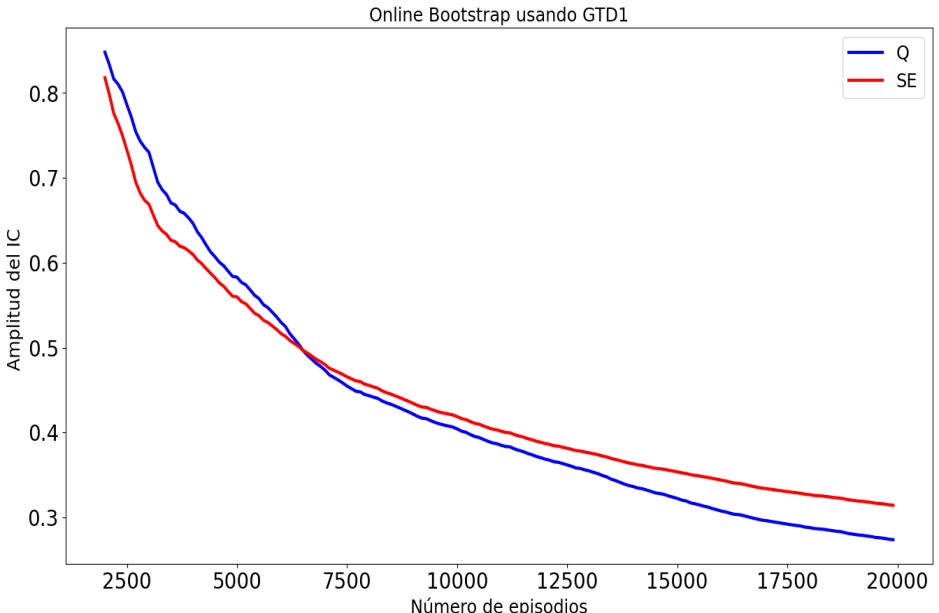
Off policy

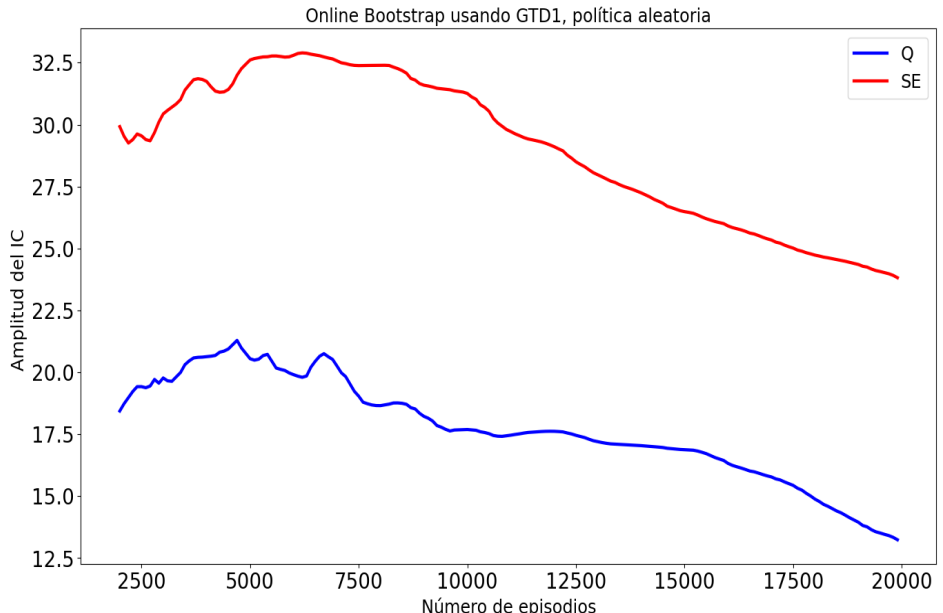
- Simulador del manejo de la sepsis (Oberst y Sontag, 2019).
- Este simula 4 signos del paciente: frecuencia cardiaca, presión sanguínea, concentración de oxigeno y niveles de glucosa.
- Hay 720 estados. Hay 8 acciones, que son las combinaciones de tratamientos posibles.
- La recompensa es 1 si el paciente es dado de alta, -1 si muere y 0 en cualquier otro caso.
- π es óptima bajo Q-learning y π^b es una Q-política $\epsilon-$greedy con $\epsilon = 0{,}05$.

Política ϵ-greedy vs alteatoria: GTD1

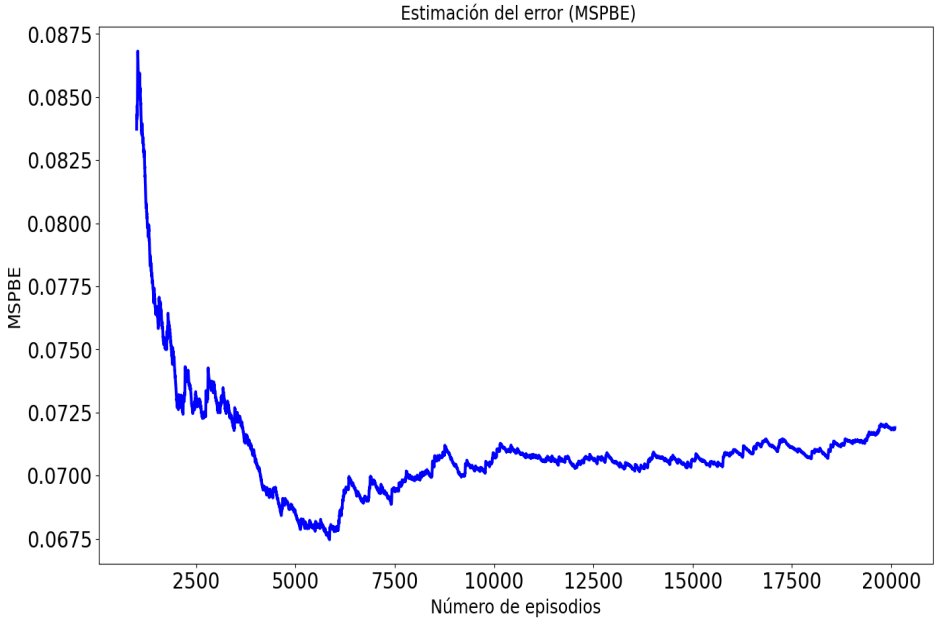


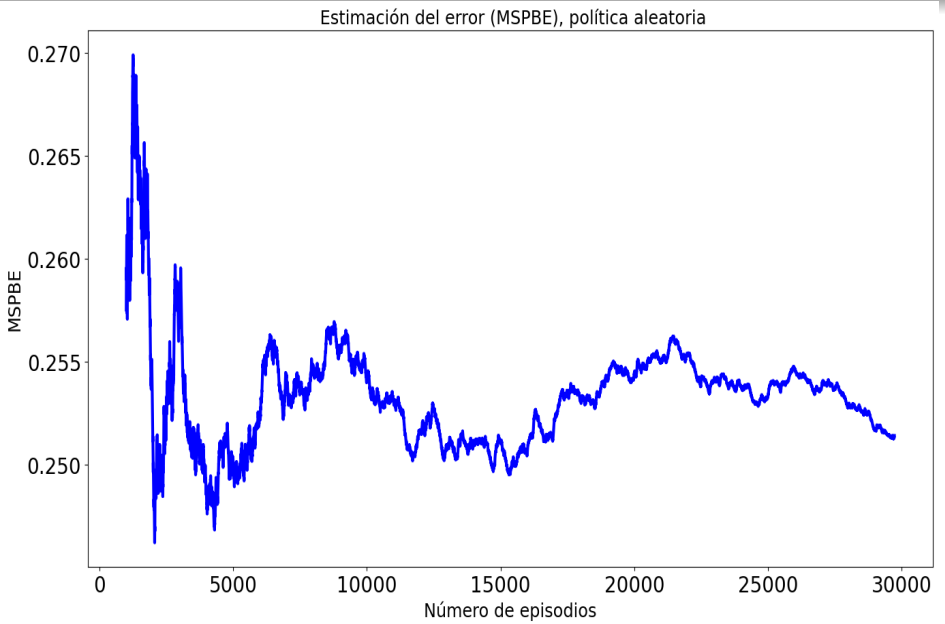


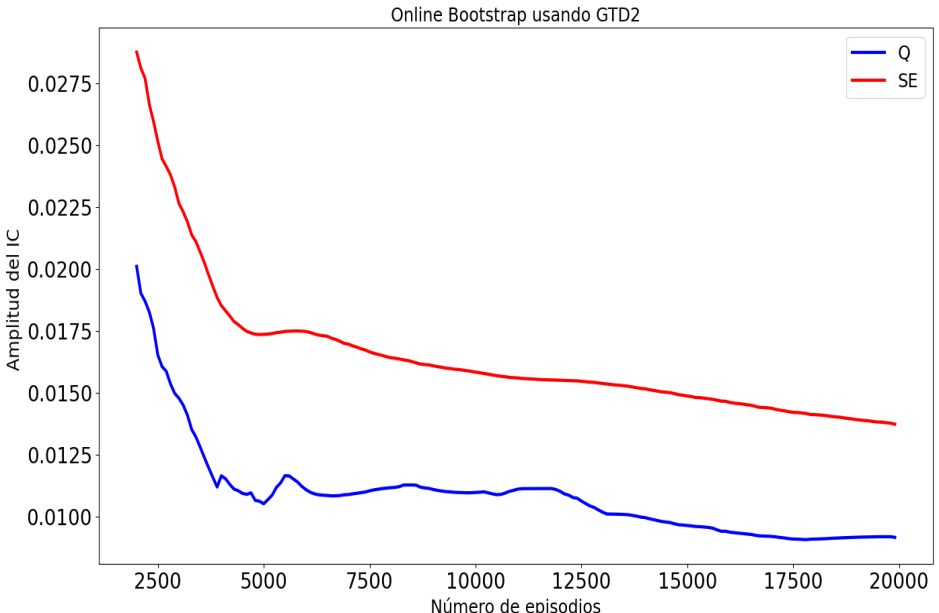


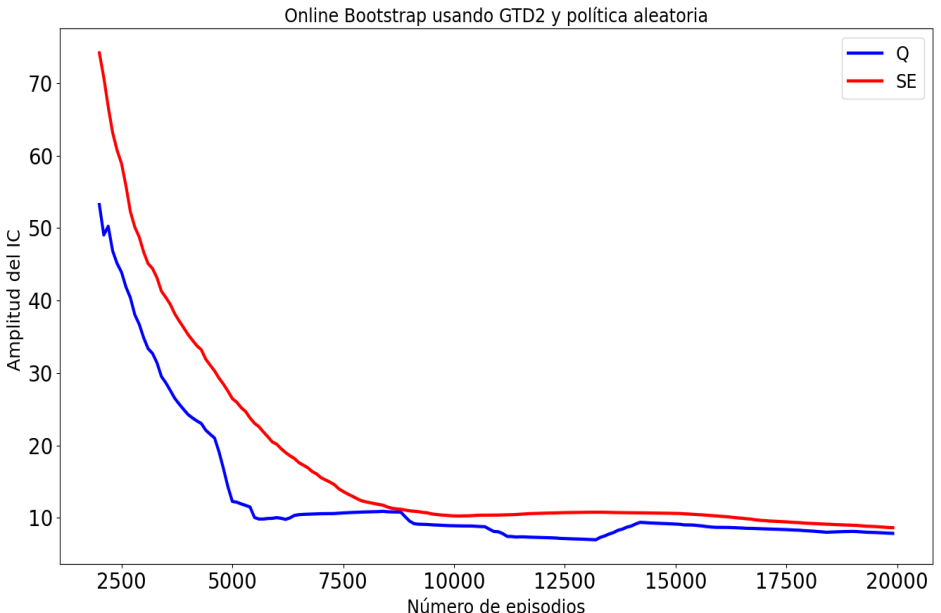


Política ϵ-greedy vs alteatoria: GTD2

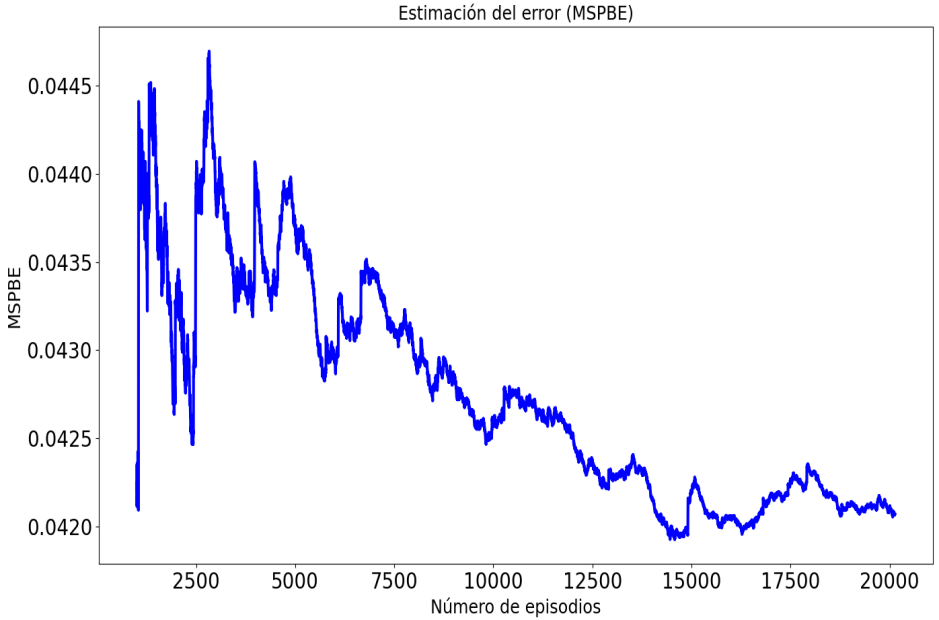


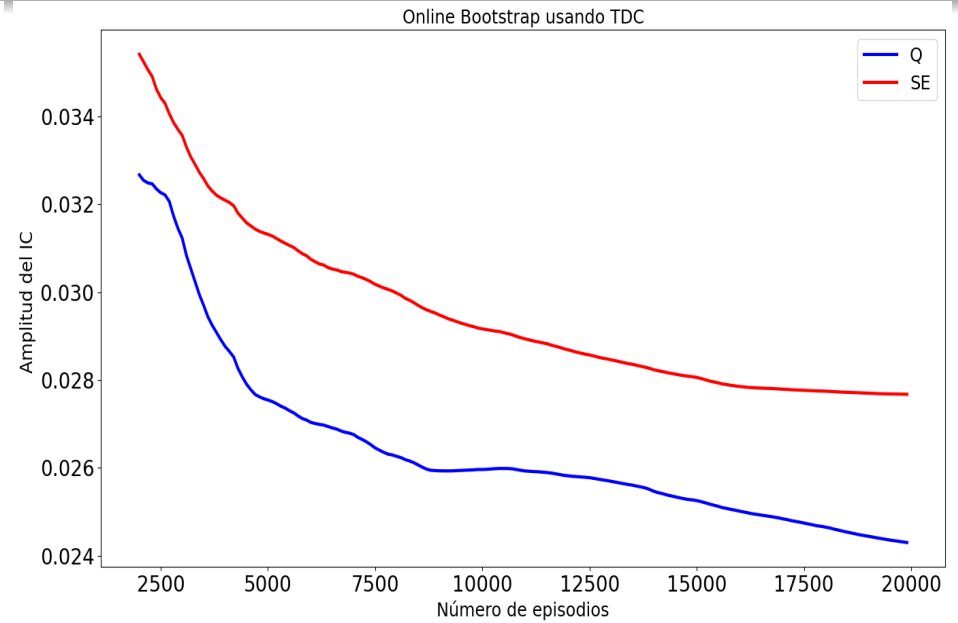


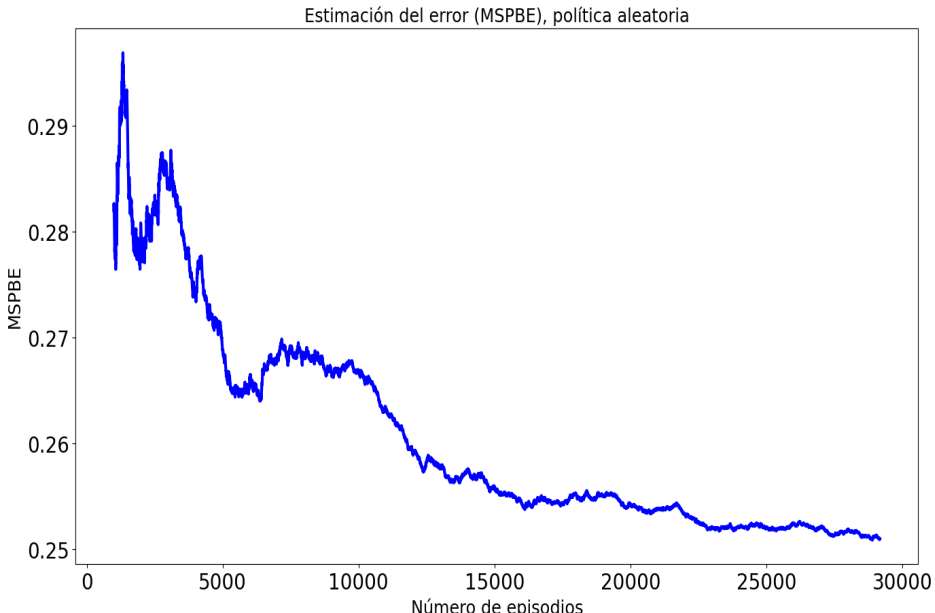


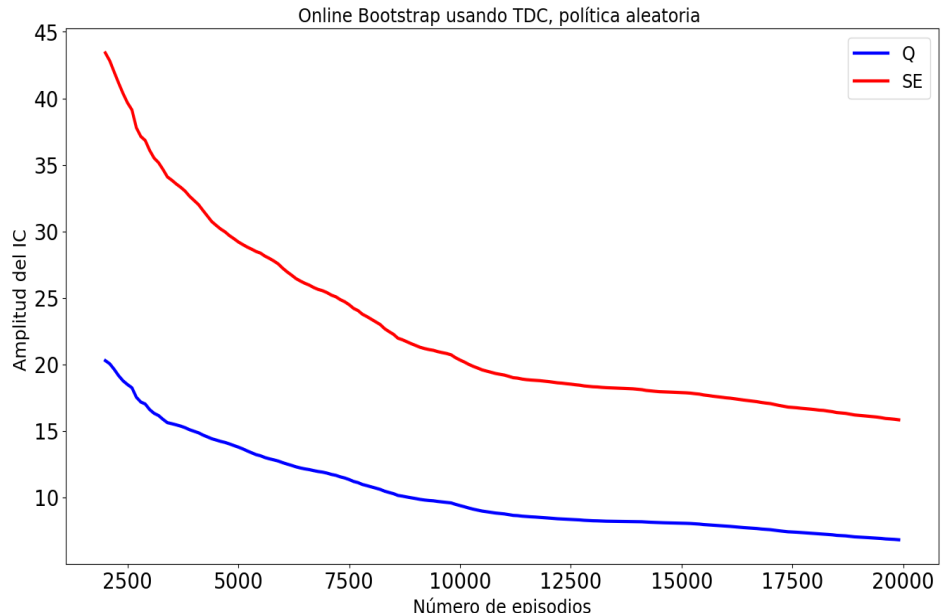


Política ϵ-greedy vs alteatoria: TDC









Conclusiones

- El Online Bootstrap tiene una motivación clara. Esta es la necesidad de evaluar políticas usando una estrategia conveniente y fácilmente implementable (en nuestro caso es aproximar linealmente la función valor), pero penalizando el ruido subyacente que distorsiona los resultados.
- El Online Bootstrap permite garantizar que la función valor verdadera estará dentro de un intervalo siempre. Aunque, TD tiene un comportamiento aceptable, su estimación no siempre es precisa.

- El comportamiento de los métodos GTD es muy parecido. El que mejor se comporta es GTD2, le sigue TDC y por último GTD1. Es probable que estos efectos se deban a la función objetivo que están minimizando.
- El algoritmo de Online Bootstrap es sensible ante cambios en la política evaluada. Aunque la convergencia se da, esta es más lenta y el tiempo computacional requerido es mayor.

[Arfé, 2017] Arfé, A. (2017).
Stochastic approximation and martingale methods.

[Bach et al., 2016] Bach, F., Liu, Y., and Li, R. (2016).
Statistical machine learning and convex optimization.
Département d'Informatique de l'ENS (DI ENS).

[Bercu, 2019a] Bercu, B. (2019a).
Asymptotic behavior of stochastic algorithms with statistical applications. part i.

[Bercu, 2019b] Bercu, B. (2019b).
Asymptotic behavior of stochastic algorithms with statistical applications. part ii.

[Borkar, 2006] Borkar, V. S. (2006).
Stochastic approximation with 'controlled markov noise'.
Systems and Control Letters, 55(2):139–145.

[Borkar, 2008] Borkar, V. S. (2008).
Stochastic approximation a dynamical systems viewpoint.
Cambridge University Press.

[Haskell, 2011] Haskell, W. B. (2011).
Introduction to dynamic programming.
University of Alberta.

[Karmakar, 2020] Karmakar, P. (2020).
Stochastic approximation with markov noise: Analysis and applications in reinforcement learning.

[Kushner and Yin, 2003] Kushner, H. and Yin, G. (2003).
Stochastic approximation and recursive algorithms and applications.
Springer.

[Levin et al., 2017] Levin, D., Peres, Y., and Wilmer, E. L. (2017).
Markov chains and mixing times.
American Mathematical Society.

[Liang, 2010] Liang, F. (2010).
Trajectory averaging for stochastic approximation mcmc algorithms.
The Annals of Statistics.

[Maei, 2011] Maei, H. R. (2011).
Gradient temporal-difference learning algorithms.
University of Alberta.

[Oberst and Sontag, 2019] Oberst, M. and Sontag, D. (2019).
Counterfactual off-policy evaluation with gumbel-max structural causal models.
Proceedings of the 36th International Conference on Machine Learning.

[Ramprasad et al., 2022] Ramprasad, P., Li, Y., Yang, Z., Wang, Z., Sun, W., and Cheng, G. (2022).
Online bootrstrap inference for policy evaluation in reinforcement learning.
Journal of the American Statistical Association.

[Robbins and Monroe, 1951] Robbins, H. and Monroe, S. (1951).
A stochastic approximation method.
The Annals of Mathematical Statistics.

[Sutton and Barto, 2018] Sutton, R. and Barto, A. (2018).
Reinforcement learning: An introduction.
The MIT Press.

[Xu et al., 2020] Xu, T., Wang, Z., Zhou, Y., and Liang, Y. (2020).
Reanalysis of variance reduced temporal difference learning.
International Conference on Learning Representations.