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 $g(x_t; \ _{t+1})$, donde el ruido $_{t+1}$ perturba las observaciones y se cumple que $f(x) = E[g(x; \ )]$

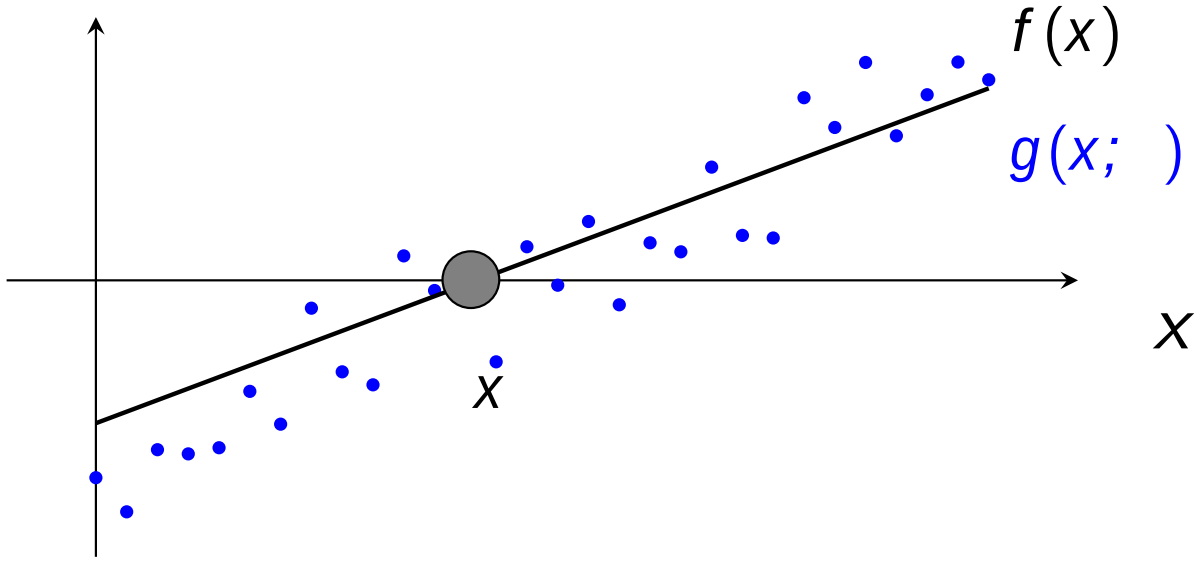

Figura: Representación gráfica de *LSA*.

Notacionalmente, tomamos

$$f(x_t) = \overline{A}x_t \quad \overline{b}; \qquad g(x_t;\ _{t+1}) = \tilde{A}_{t+1}x_t \quad \tilde{b}_{t+1}$$

donde

- $\overline{A}; \tilde{A}_t\ 2\ \mathrm{R}^{d\ \ d}; E[\tilde{A}_t] = \overline{A}$
- $\overline{b}; \tilde{b}_t; x_t\ 2\ \mathrm{R}^d; E[\tilde{b}_t] = \overline{b}$

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

Iteración de LSA (Robbins Monro)

Formalmente, la actualización está dada por:

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

- x_t es el parámetro de LSA
- $\tilde{A}_t x_t \quad \tilde{b}_t$ es una observación con ruido de $\overline{A} x_t \quad \overline{b}$, donde el ruido $fe_{t+1}g = f(\tilde{A}_{t+1} \quad \overline{A})x_t \quad (\tilde{b}_{t+1} \quad \overline{b})g$ es una diferencia de martingalas
- $_{t+1}$ 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* $e_{t+1} = (\tilde{A}_{t+1} \quad \overline{A})x_t \quad (\tilde{b}_{t+1} \quad \overline{b})$, *donde* $fe_t g_{t \ 0}$ *es una sucesion de diferencia de martingalas; es decir,* $\mathrm{E}[e_{t+1} j F_t] = 0$, *donde* F_t *es el conjunto de informacion.*

2. *La sucesion* $f\tilde{A}_{t+1}x_t \quad \tilde{b}_{t+1}g_{t \ 0} \ 2 \ L^2$ *con* $\mathrm{E}[jj\tilde{A}_{t+1}x_t \quad \tilde{b}_{t+1}jj^2 j F_t] \quad K(1 + jjx_t jj^2)$ *c.s para algun* $K > 0$

3. $\mathrm{P}_{t=0}^{1} \ t = 1$ *y* $\mathrm{P}_{t=0}^{1} \ \frac{2}{t} < 1$

4. *La sucesion* $fx_t g_{t \ 0}$ *esta acotada, sup*$_{t \ 0} jjx_t jj_2 < 1$

entonces $\lim_{t \ ! \ 1} x_t = x \quad c.s$

Teoría asintótica para LSA

Teorema 2 (Normalidad asintótica de Robbins Monro)

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

$Q = lim_{t \to 1} \mathrm{E}[e_t e_t^T]$

$\overline{A}$ *es una matriz Hurwitz, es decir que todos sus valores propios tiene parte real estrictamente negativa.*

entonces

$$p_{\bar{t}}(x_t \quad x) \overset{d}{!} N(0; \overline{A}^{\ 1} Q(\overline{A}^{\ 1})^T)$$

Corolario 3 (Polyak-Ruppert)

La convergencia casi siempre y la normalidad asintotica tambien valen para la iteracion 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 $fX_t g = fX_1; X_2; :::; g$ tal que

$$\mathrm{P}[X_{t+1} j X_t] = \mathrm{P}[X_{t+1} j X_1; :::; X_t]$$

Su kernel de transición P es una matriz con las probabilidades de transición de un evento a otro.

Teorema 5

Una cadena ergodica de Markov tiene distribucion l mite , que es unica y es igual a su distribucion estacionaria.

Definición 6 (LSA con ruido markoviano)

Sea $f\tilde{A}(X_t); \tilde{b}(X_t)g_{t \ 1}$ una secuencia de observaciones, donde $fX_t g$ es una cadena ergódica de Markov con espacio de estados X y distribución estacionaria . Además $\tilde{A} : X \ ! \ R^{d \ d}; \tilde{b} : X \ ! \ R^d$ y $E \ [\tilde{A}(X_{t+1})] = \overline{A}; E \ [\tilde{b}(X_{t+1})] = \overline{b}$. Entonces la actualización dada por

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

corresponde al algoritmo de LSA con ruido Markoviano, donde $x \ 2 \ R^d$ es el parámetro de LSA y $f \ _t g_{t \ 1}$ es una sucesión de pasos decrecientes, dada por $_t = \frac{^0}{t}$ para algún $_0 > 0; \ 2 \ (\frac{1}{2}; 1)$

Teoría asintótica para LSA con ruido Markoviano

- Ahora, $_{t+1} = (\tilde{A}(X_{t+1}) \quad \overline{A})x_t \quad (\tilde{b}(X_{t+1}) \quad \overline{b})$ y no se puede garantizar que sea una diferencia de martingalas
- Pero podemos escribir $_t = e_t + \ _t + \ _t$, donde e_t es la diferencia de martingalas de LSA en el caso general, $_t$ y $_t$ 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 $sup_{x2X} jj\tilde{A}(x)jj_F \quad A_{max}; \quad sup_{x2X} jj\tilde{b}(x)jj_2 \quad b_{max}$, entonces:

Proposición 1

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

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

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

$$\hat{x}^b_{t+1} = \hat{x}^b + \alpha_{t+1} W^b_{t+1} (\tilde{A}(X_{t+1})\hat{x}^b - \tilde{b}(X_{t+1}))$$

$$\bar{\hat{x}}^b_{t+1} = \frac{1}{t+1} \sum_{i=1}^{t+1} \hat{x}^b_i \qquad (3)$$

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

¿La razon? se puede mostrar que $\sqrt{t}(\bar{\hat{x}}_t - \bar{x}_t) \to N(0; \bar{A}^{-1} Q (\bar{A}^{-1})^T)$; $t \to 1$ (Teo. 4.2 [Ramprasad et al., 2022]) ... <Es la misma de $\sqrt{t}(\bar{x}_t - x^*)$!

Podemos aproximar la distribucion de $\sqrt{t}(\bar{\hat{x}}_t - \bar{x}_t)$ con B muestras bootstraps de $\bar{\hat{x}}_t - \bar{x}_t$ para cada t, es decir (3).

Algoritmo 1 Online Bootstrap Inference

Input: Numero de muestras bootstrap B, α_0; $\alpha \in (\frac{1}{2}; 1)$ y estimadores iniciales $x_0 = \hat{x}_0^b$; $b = 1; :::; B$ linenosize=

for $t = 0; 1; 2; :::$ do
  Observar $\tilde{A}(X_{t+1})$; $\tilde{b}(X_{t+1})$
  Computar $\alpha_{t+1} = \alpha_0(t+1)^{-\alpha}$
  Actualizar $x_{t+1} \leftarrow x_t + \alpha_{t+1}(\tilde{A}(X_{t+1}))x_t - \tilde{b}(X_{t+1}))$
  Actualizar $\bar{x}_{t+1} \leftarrow \frac{1}{1+t}(t\,x_t + x_{t+1})$
  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 $\bar{\hat{x}}_{t+1}^b \leftarrow \frac{1}{1+t}(t\,\bar{\hat{x}}_t^b + \hat{x}_{t+1}^b)$
  end for
end for

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

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

Durante cada $t = 1, 2, \ldots$ un agente se encuentra en un estado s_t.

Toma una accion a_t, de las disponibles para ese s_t.

Dependiendo de la con guracion del ambiente, recibe una recompensa r_{t+1} y llega a un estado s_{t+1}.

Un Proceso de Decision de Markov (MDP) es formalizar el ambiente de RL. Se denota $M = (S; A; P; R;)$ donde $S; A$ son los espacios de estados y acciones, respectivamente; P es el kernel de transicion y R la funcion de recompensas.

Una politica 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 politica inducen un Proceso de Recompensa de Markov (MRP) , que se denota $M = (M ;)$.

La funcion valor asociada a es la suma descontada de recompensas esperadas partiendo de y siguiendo :

$$V(s) = E[\sum_{t=0}^{1} {}^{t} R(s_t) | s_0 = s;]; \quad s \in S$$

$$= R(s) + P(s) V(s)$$

$$:= TV(s) \quad \text{(Ecuacion de Bellman)}$$

donde $P(s); R(s)$ son matrices calculables. Como $TV(s)$ es una contraccion, por teorema de punto jo de Banach $V(s)$ es la solucion unica. As:

$$V(s) = (I \quad P(s))^{1} R(s) : \qquad (4)$$

En RL, es comun que haya incertidumbre, eso implica desconocer P .

Aun si hubiese certidumbre, las dimensiones del ambiente pueden ser grandes y as las dimensiones P de R) <No se puede usar la ecuacion de Bellman!

Una alternativa es aproximar linealmente la funcion valor como $V(s) = (s)^T$, donde $2 R^d$ es el parametro de LSA y $(s)^T = [_1; :::; _d]$ con $_i : S ! R$ features (o funciones base)

Cuando se aplica LSA a RL con esa aproximacion $V_t(s)$ el ruido de LSA es Markoviano. Ad $fX_t g = f(s_t; r_{t+1}; s_{t+1}) g_{t 0}$.

I.Diferencias Temporales (TD)-On policy

La actualizacion de la funcion valor esta dada por:

$$V_{t+1}(s_t) = V_t(s_t) + \quad _t[V_t(s_t) \quad V_t(s_{t+1}) \quad r_{t+1}]: \qquad (5)$$

Aplicando nuestra aproximacion lineal de la funcion valor (5) es equivalente a:

$$_{t+1} = \quad _t + \quad _{t+1}[\underbrace{|\ (s_t)(\ (s_t) \quad (s_{t+1}))^T}_{\tilde{A}(X_t)} \quad _t \quad \underbrace{r_{t+1}\{z\ (s_t)}_{\tilde{b}(X_t)}]: \qquad (6)$$

Que es la iteracion de LSA con:

$\tilde{A}(X_t) = \quad (s_t)(\ (s_t) \quad (s_{t+1}))^T$ y $\tilde{b}(X_t) = r_{t+1} \quad (s_t)$.

$\bar{A} = E \quad [\tilde{A}(X_t)] = \quad ^T(\ I \quad P\ )$ y

$\bar{b} = E \quad [\tilde{b}(X_t)] = \quad ^T r$, donde es la distribucion estacionaria de $fX_t g$ bajo la politica .

Ahora:

- En el caso on-policy, las observaciones se generan con la misma politica a evaluar.
- Pero en el caso o -policy, las observaciones se generan con una politica ^{b} (behavior policy) distinta a la evaluar t (target policy).

Los problemas:

- TD no es un gradiente) inestabilidad.
- Hay una desconexion, ^{b} puede seleccionar acciones que nunca tomara) inestabilidad.

Solucion:

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

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

A. Derivacion: usando gradiente descentente,

GTD1 minimiza la norma esperada de la actualizacion de TD:

$$NEU( ) = E [_t( ) (s_t)]^T E [_t( ) (s_t)]$$

donde $_t( ) = r_{t+1} + _t^T (s_{t+1}) _t^T (s_t)$ es el error de TD.

GTD2 y TDC minimizan el error cuadratico medio de Bellman projectado:

$$MSBPE( ) = jjV TV jj^2$$

donde es el operador de proyeccion, matriz cuyos elementos en la diagonal son las entradas de

Al calcular el gradiente, se obtienen productos de valores esperados no necesariamente independientes, se incluye una actualizacion auxiliar enlazada.

Se incluye un ratio de Important Sampling $= \frac{(a_t|s_t)}{b(a_t|s_t)}$ para corregir la discrepancia entre , b:

Formalmente, tome

$$A_t = \quad _t \quad (s_t)(\quad (s_t) \qquad (s_{t+1}))^T \text{ y } b_t = \quad _t r_{t+1} \quad (s_t)$$

como estimadores insesgados de $A = \quad ^T \quad (\quad I \qquad P \quad)$ y $b = \quad ^T \quad r$, respectivamente; entonces los algoritmos GTD se escriben como:

GTD1:

$$\begin{aligned} y_{t+1} &= y_t + \quad _{t+1}[b_t + A_t \quad _t \quad I_d y_t] \\ \quad _{t+1} &= \quad _t + \quad _{t+1} A_t^T y_t \end{aligned} \tag{7}$$

GTD2:

$$\begin{aligned} y_{t+1} &= y_t + \quad _{t+1}[b_t + A_t \quad _t \quad (s_t)^T y_t \quad (s_t)] \\ \quad _{t+1} &= \quad _t + \quad _{t+1} A_t^T y_t \end{aligned} \tag{8}$$

TDC:

$$\begin{aligned} y_{t+1} &= y_t + \quad _t \quad _{t+1}[b_t + A_t \quad _t \quad (s_t)^T y_t \quad (s_t)] \\ \quad _{t+1} &= \quad _t + \quad _{t+1}[b_t + A_t \quad _t \quad _t \quad (s_{t+1}) \quad (s_t)^T y_t] \end{aligned} \tag{9}$$

B.Compacti cando como problema de LSA:

GTD1 y GTD2:

$$\tilde{A}_t = \begin{matrix} 0 & A_t^T \\ A_t & M_t \end{matrix} \qquad \tilde{b}_t = \begin{matrix} 0 \\ b_t \end{matrix} \qquad {}_t = \begin{matrix} t \\ y_t \end{matrix} \qquad (10)$$

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

TDC:

$$\tilde{A}_t = \begin{matrix} A_t & (M_t \quad A_t)^T \\ {}_t A_t & {}_t M_t \end{matrix} \qquad \tilde{b}_t = \begin{matrix} b_t \\ {}_t b_t \end{matrix} \qquad {}_t = \begin{matrix} t \\ y_t \end{matrix} \qquad (11)$$

donde $M_t = (s_t)\ ^T(s_t)$, ${}_t$ proxy emprica de > 0, donde es tal que es mayor al negativo del mnimo valor propio de la matriz $(\ ^T \ )\ \ 1\frac{(A+A^T)}{2}$.

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

Algoritmo 2 Metodo clasico

Input: θ_0; $\alpha \in (\frac{1}{2}; 1)$ y estimador inicial θ_0
for $t = 0; 1; 2; \ldots$ do
  Observar $\tilde{A}(X_{t+1})$; $\tilde{b}(X_{t+1})$
  Computar $\alpha_{t+1} = \alpha_0(t+1)^{-\alpha}$
  Actualizar $\theta_{t+1} \leftarrow \theta_t + \alpha_{t+1}(\tilde{A}(X_{t+1})\theta_t - \tilde{b}(X_{t+1}))$
  Actualizar $\bar{\theta}_{t+1} \leftarrow \frac{1}{1+t}(t\bar{\theta}_t + \theta_{t+1})$
end for
Output Estimador $\bar{\theta}_{t+1}$

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

$$\bar{V}_t(\cdot) = \int_{s \in S} \bar{V}_t(s)\, \mu(ds)$$

3. De nimos los bootstraps de valores estimados como

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

4. Construimos intervalos de con anza.

$IC_{cuantil}(V_t(\ )) = [\ V_t(\ ) + q_{\frac{}{2}}; V_t(\ ) + q_{1\ \ \frac{}{2}}]$,
donde$q$ es el esimo cuantil de la distribucion (12).

$IC_{SE}(V_t(\ )) = [\ V_t(\ ) + z_{\frac{}{2}} \sim;\ V_t(\ ) + z_{1\ \ \frac{}{2}} \sim]$,
donde$z$ hace referencia a la distribucion normal y
$\sim = \sqrt{\frac{s^2}{B}}$; s^2 es la varianza de (12).

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 %, $B = 200$, $\alpha = \frac{2}{3}$, $\theta_0 = 10$. Además, $W_t \sim U[1 - \frac{1}{\sqrt{3}}, 1 + \frac{1}{\sqrt{3}}]$.

Analizamos un caso on-policy y otro o -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 A = fizquierda; derecha; arriba; abajog.

El objetivo es llegar al ultimo estado (casilla con coordenadas 8 8).

La recompensa es 1 en el ultimo estado y 0 de lo contrario.

La poltica , con la que se generan las observaciones, es una Q-poltica greedy con = 0;2; y entre mayor sea, hay mas aleatoriedad.

Funcion valor verdadera

0;011(2 IC siempre)

TD se acerca a mayor numero de iteraciones

Hipotesis: Mayor distancia entre el kernel de transicion de X_t bajo aleatoria y su distribucion estacionaria se relaciona con convergencia mas lenta

$= 0;3$

$= 0;5$

Simulador del manejo de la sepsis (Oberst y Sontag, 2019).

Este simula 4 signos del paciente: frecuencia cardiaca, presion sanguinea, concentracion 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.

es optima bajo Q-learning y ^{b} es una Q-politica greedy con $= 0;05$.

El Online Bootstrap tiene una motivacion clara. Esta es la necesidad de evaluar politicas usando una estrategia conveniente y facilmente implementable (en nuestro caso es aproximar linealmente la funcion valor), pero penalizando el ruido subyacente que distorsiona los resultados.

El Online Bootstrap permite garantizar que la funcion valor verdadera estara dentro de un intervalo siempre. Aunque, TD tiene un comportamiento aceptable, su estimacion no siempre es precisa.

El comportamiento de los metodos GTD es muy parecido. El que mejor se comporta es GTD2, le sigue TDC y por ultimo GTD1. Es probable que estos efectos se deban a la funcion objetivo que estan minimizando.

El algoritmo de Online Bootstrap es sensible ante cambios en la poltica evaluada. Aunque la convergencia se da, esta es mas lenta y el tiempo computacional requerido es mayor.

[Arfe, 2017] Arfe, 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.