Aprendizaje Reforzado

Maestría en Data Mining, Universidad Austral

Javier Kreiner

Plan de la clase

- Métodos libres de modelo (model-free)
- Evaluación Monte Carlo
- Evaluación por Diferencia Temporal (TD)
- Control Monte Carlo

Predicción libre de modelo

$$v_\pi(s) = \mathcal{R}_s^\pi + \sum_{s'} v_\pi(s') p_{s,s'}^\pi$$

En general NO CONOCEMOS $p_{s,s'}^\pi$

¿Podemos hacer evaluación, aprendiendo tan sólo de la experiencia?

Predicción/Evaluación Monte Carlo

- Los métodos Monte Carlo aprenden de la experiencia
- Son ‘libre de modelo’
- Utilizan episodios completos
- Usa la idea más simple posible: función de valor ≅ retorno total promedio
- Sólo lo podemos usar con tareas periódicas

Recordemos

Función de recompensa $\mathcal{R}_s := E[R_{t+1}|S_t = s]$

Retorno $G_t := R_{t+1} + \gamma R_{t+2} + \cdot$

Función de Valor $v(s) = E[G_t|S_t = s]$

En esta última expresión podemos usar la media empírica.

Monte Carlo ‘primera visita’

- Para estimar la función de valor para el estado s
- Tomar la primera vez que el estado s es visitado en un episodio
- Incrementar el contador de s: N(s) = N(s) + 1
- Incrementar la sumas de los retornos totales: S(s) = S(s) + G_t
- El valor estimado es la media empírica del retorno: V(s) = S(s)/N(s)
- Por la ley de los grandes números V(s) converge a $v_\pi(s)$ si N(s) tiende a infinito:

Ley de los grandes números

$$\frac{\sum_{i=1}^{n} X_i}{n} \rightarrow E[X], \qquad X_i,\ iid$$

Pseudocódigo

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Detalle computacional

$$\mu_k := \frac{\sum_{j=1}^{k} x_j}{k} = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}$$

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{(G_t - V(S_t))}{N(S_t)}$$

Aprendizaje por Diferencia Temporal (TD)

$$v_\pi(s) = E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$v_\pi^{n+1}(S_t) = v_\pi^n(S_t) + \alpha[R_{t+1} + \gamma v_\pi^n(S_{t+1}) - v_\pi^n(S_t)]$$

Retorno estimado: $R_{t+1} + \gamma v_\pi^{n+1}(S_{t+1})$

Error de TD: $[R_{t+1} + \gamma v_\pi^n(S_{t+1}) - v_\pi^n(S_t)]$

Aprendizaje por Diferencia Temporal

- Este tipo de métodos aprende directamente de la experiencia
- Son libres de modelo
- Aprenden con episodios incompletos, utilizando bootstrapping
- Utilizan una estimación anterior para realizar una estimación

Comparativa

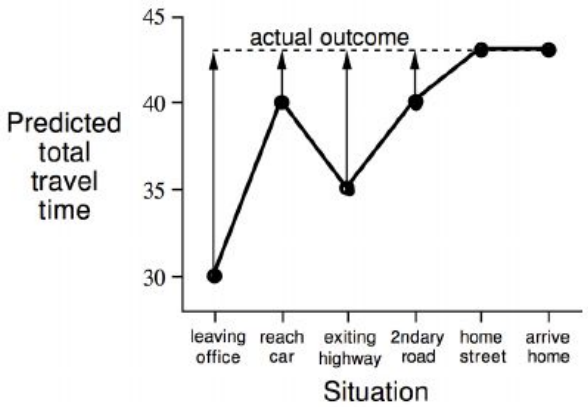
$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = S]$$

- Programación Dinámica: Actualiza directamente con las esperanzas.
- Monte Carlo: Actualiza usando como target una aproximación de la esperanza que se actualiza *sólo al final del episodio*.
- Diferencia Temporal: Utiliza otra aproximación de la esperanza, pero *se actualiza en cada paso*.
- *Bootstrapping:* El update actualiza una *estimación previa*

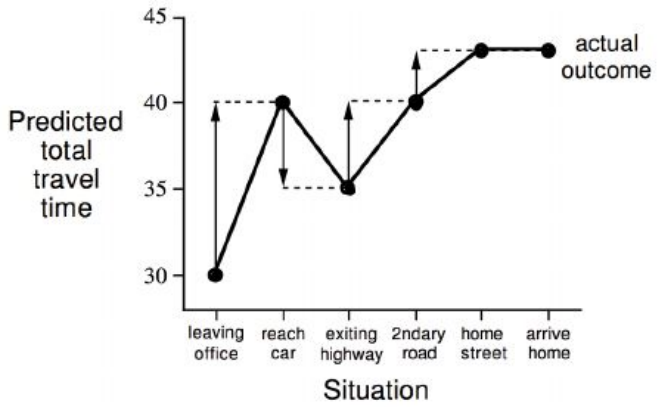
Driving Home (Ejemplo 6.1)

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

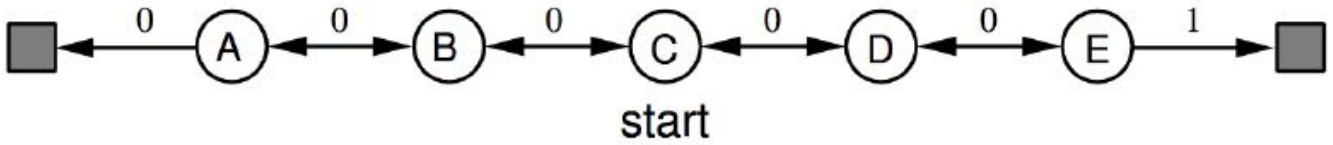
Changes recommended by Monte Carlo methods (α=1)

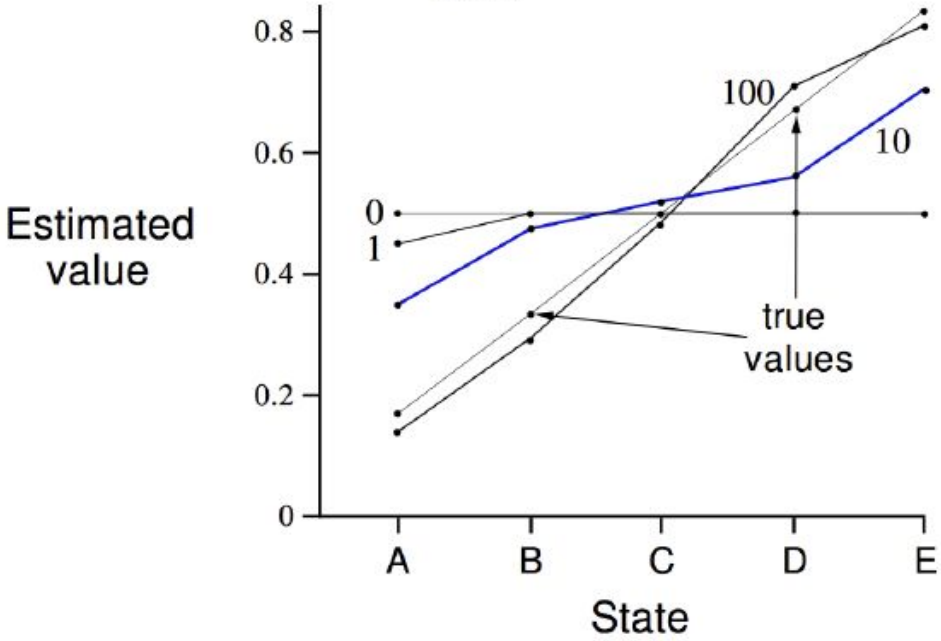


Changes recommended by TD methods (α=1)

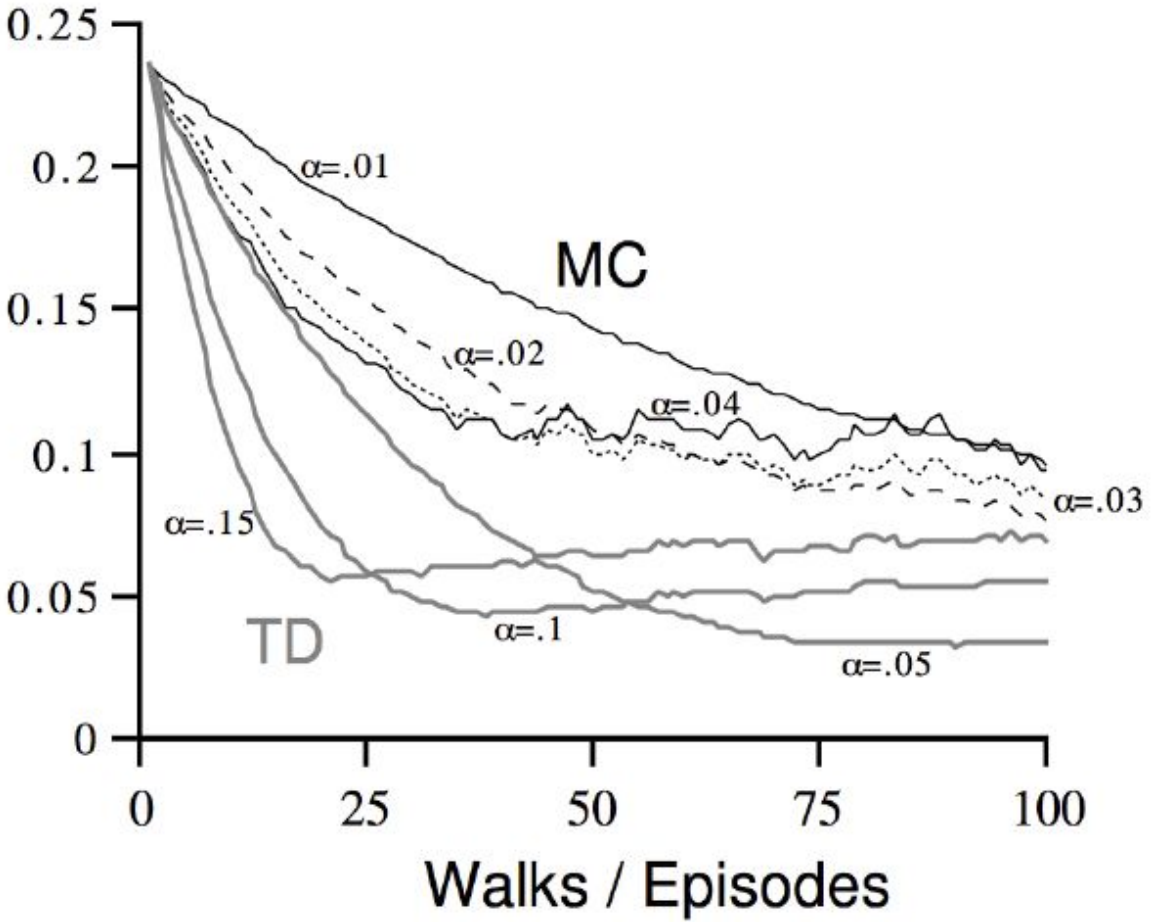


Ejemplo 6.2: Caminata aleatoria

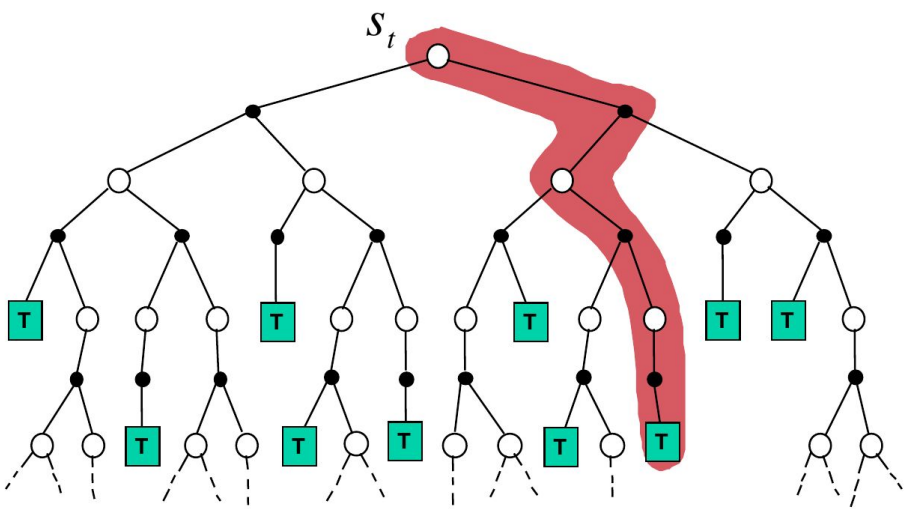




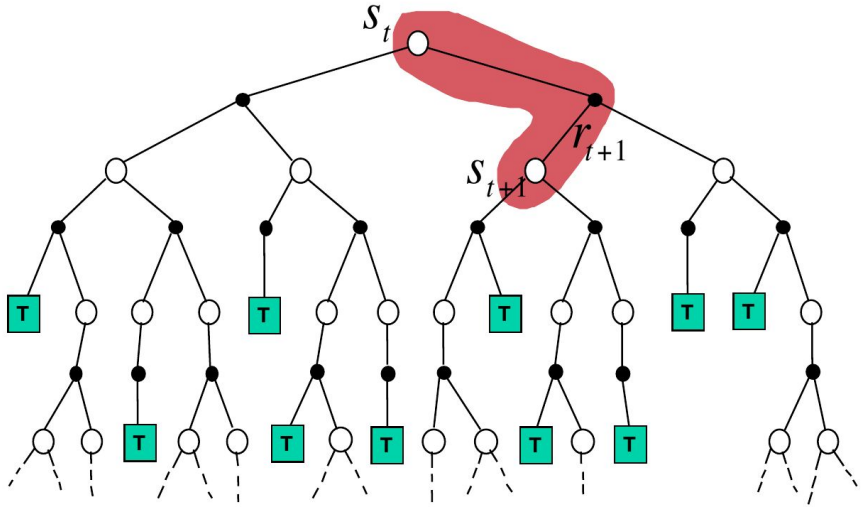
RMS error, averaged over states



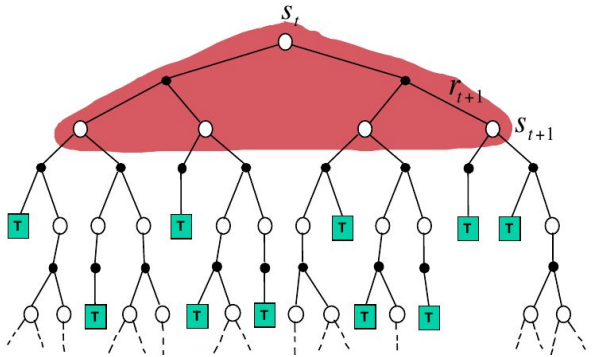
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



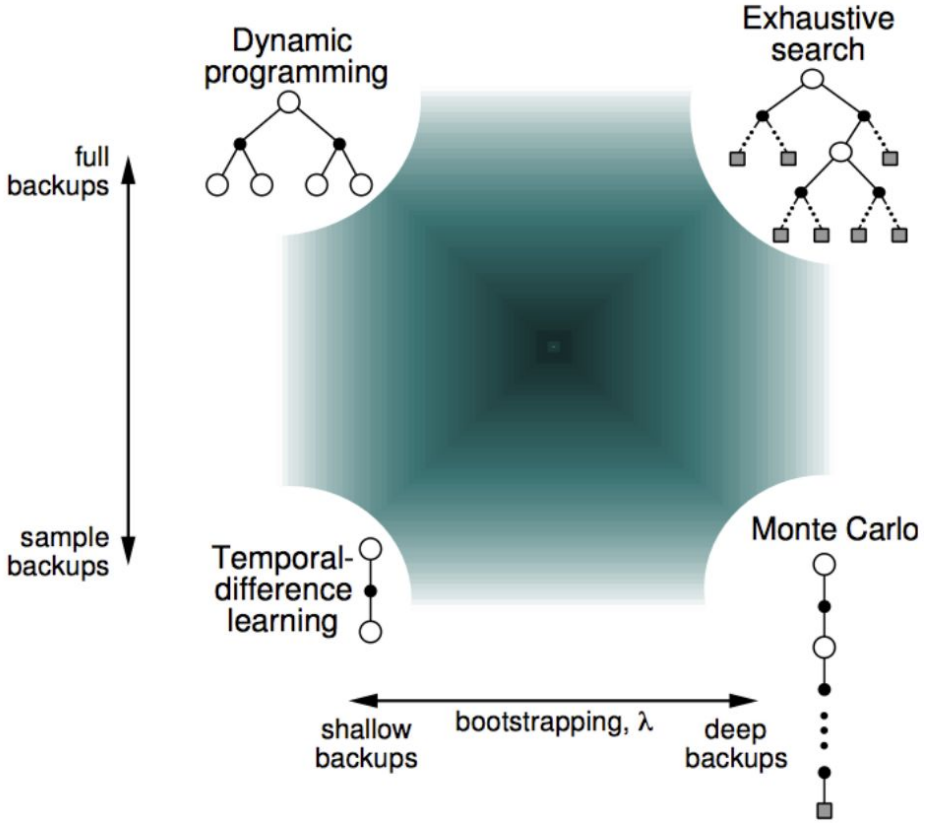
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



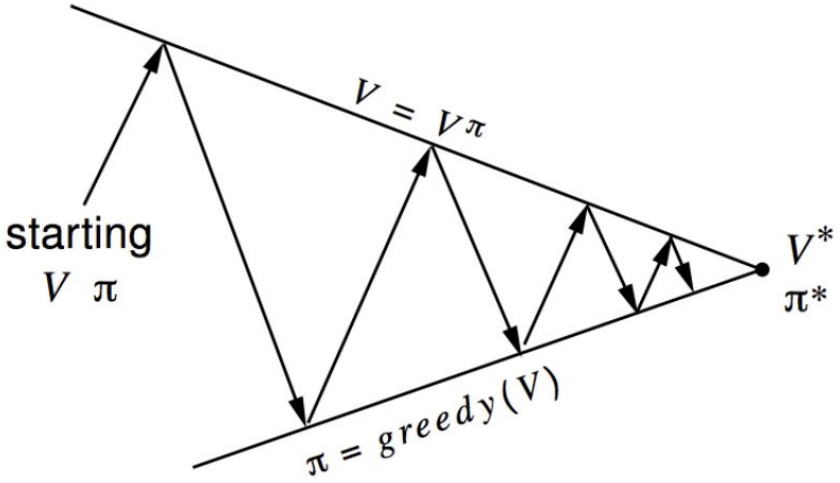
Comparativa de los métodos vistos



Monte Carlo para la función de valor estado-acción

- Ahora para cada tupla (estado, acción) queremos saber $q_\pi(s,a)$
- El problema es que si generamos episodios sólo con π y es determinística, hay algunos pares que nunca vamos a visitar
- Una forma es de alguna forma garantizarnos que vamos a tener episodios con todos los pares (s,a) (‘comienzos exploratorios’)
- Esto no siempre es posible. La otra opción es explorar continuamente. Es decir, con una pequeña probabilidad tomar cualquier acción aleatoriamente.

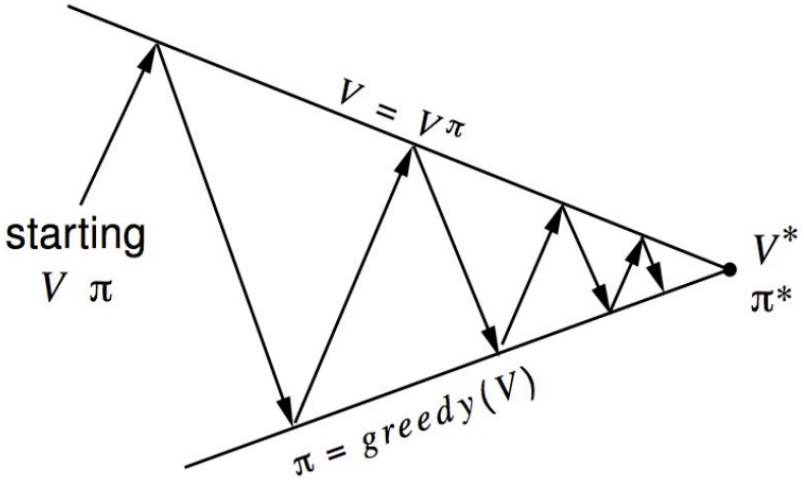
Control (Improvement) - Monte Carlo



$$q_{\pi_k}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} v_{\pi_k(s')} p_{s,s'}^a$$

$$\pi_{k+1}(s) = arg\max_a q_{\pi_k}(s,a)$$

Control (Improvement) - Monte Carlo

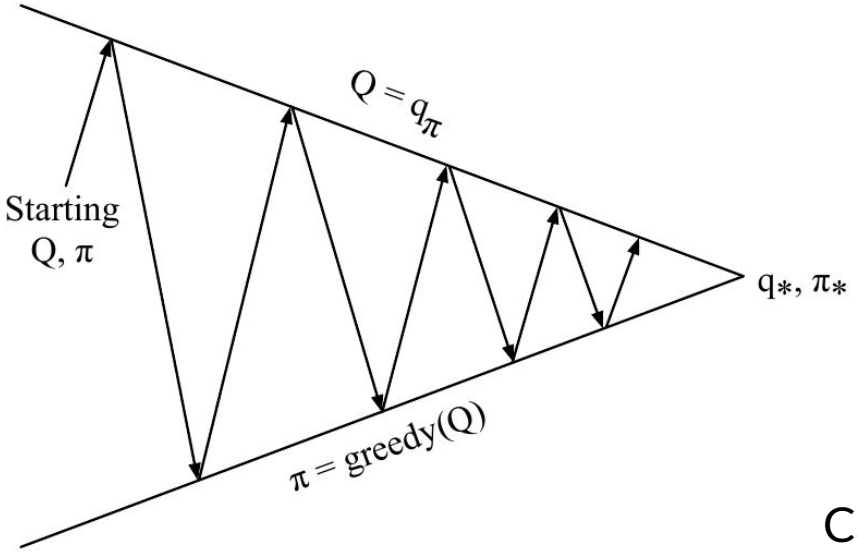


- No tengo como dato la matriz de transición. (*Model Free*)
- No tengo la esperanza *exacta*, donde están *todos* los posibles escenarios. (*Exploration vs. Explotation*)

$$q_{\pi_k}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} v_{\pi_k(s')} p_{s,s'}^a$$

$$\pi_{k+1}(s) = arg\max_a q_{\pi_k}(s,a)$$

Los parches - Dependiente del modelo (p)



Hacer evaluación MC de

$$q_{\pi_k}(s,a) =: Q_k(s,a)$$

Cambio la esperanza que aproximo.

Probar un poco todo (epsilon - greedy policy)

$$\pi^{\varepsilon}(a|s) = \begin{cases} (1-\varepsilon) + \varepsilon \frac{1}{|\mathcal{A}|} & \text{si } a = arg\max_a q_\pi(s,a) \\ \varepsilon \frac{1}{|\mathcal{A}|} & \text{caso contrario} \end{cases}$$

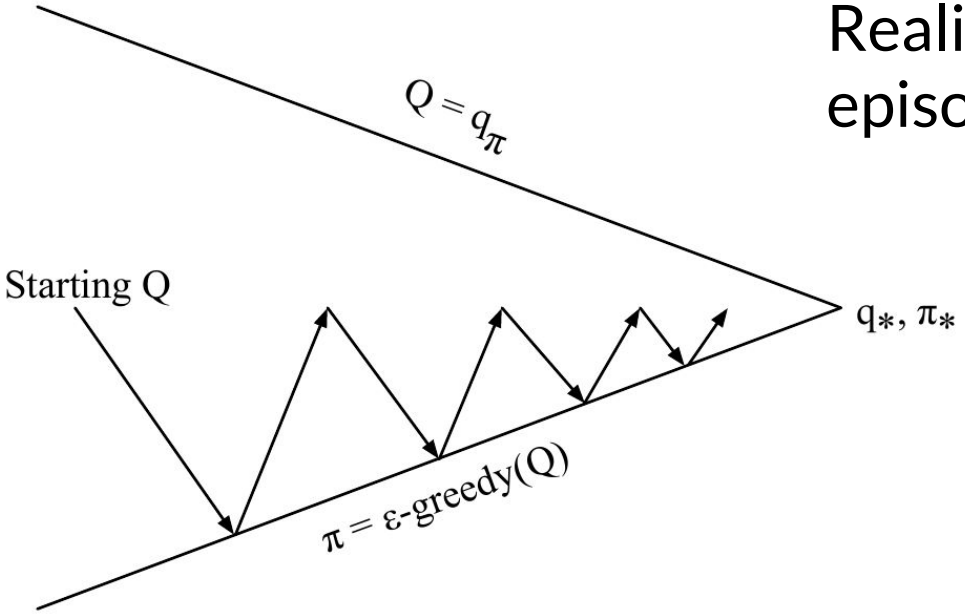
Teorema:

Si π^{ε} una política ε-greedy, $\pi'(s) := arg\max_a q_{\pi^{\varepsilon}}(s,a)$.

Entonces:

$$v_{\pi^{\varepsilon}}(s) \leq v_{\pi'^{\varepsilon}}(s)$$

Algunas mejoras



Realizar la actualización en cada episodio.

GLIE Monte Carlo

- Simular el episodio k utilizando la política π_k^{ε}: $\{S_1^k, A_1^k, R_2^k, \ldots, S_T^k\}$.
- Para cada par (s, a) del episodio

$$N^{k+1}(s,a) = N^k(s,a) + 1$$

$$Q^{k+1}(s,a) = Q^k(s,a) + \frac{1}{N^{k+1}(s,a)}(G^{k+1}(s,a) - Q^k(s,a))$$

- $$\varepsilon = \frac{1}{k}, \qquad \pi_{k+1}^{\varepsilon} = \varepsilon - \text{greedy}(Q(s,a))$$

Monte Carlo (programación)

- Ejemplo de Blackjack
- Predicción Monte Carlo
- Predicción TD
- Control Monte Carlo on-policy con políticas epsilon greedy

Ejercicio - Leer:

That any ε-greedy policy with respect to q_π is an improvement over any ε-soft policy π is assured by the policy improvement theorem. Let π' be the ε-greedy policy. The conditions of the policy improvement theorem apply because for any $s \in \mathcal{S}$:

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s,a) \\ &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \max_a q_\pi(s,a) \qquad (5.2) \\ &\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon} q_\pi(s,a) \end{aligned}$$

(the sum is a weighted average with nonnegative weights summing to 1, and as such it must be less than or equal to the largest number averaged)

$$\begin{aligned} &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + \sum_a \pi(a|s) q_\pi(s,a) \\ &= v_\pi(s). \end{aligned}$$

Thus, by the policy improvement theorem, $\pi' \geq \pi$ (i.e., $v_{\pi'}(s) \geq v_\pi(s)$, for all $s \in \mathcal{S}$).

Ejercicio (programación)

- Completar el código de predicción del valor de una política con Diferencia Temporal
- Completar el código de control Monte-Carlo y correrlo para obtener la política óptima para el BlackJack
- Obtener la política óptima y la función de valor para esa política para el ambiente Gridworld (visto en la parte de programación dinámica) utilizando control Monte Carlo, ¿cuántos episodios simulados se necesitan para obtener un resultado con 2 dígitos de precisión para todos los estados en la función de valor óptima?
- Recordar líneas que permiten usar el codigo en colab:
    - `!git clone https://github.com/javkrei/aprendizaje-reforzado-austral`
    - `%cd 'aprendizaje-reforzado-austral'`
    - `%cd 'clase 4'`

Lista de papers para el proyecto final

- Para el proyecto final (50% de la nota), en grupos de 3/4 personas:
  - elegir un paper de la siguiente lista: https://spinningup.openai.com/en/latest/spinningup/keypapers.html
  - preparar una presentación explicando el problema central del paper, el abordaje utilizado y las conclusiones
  - la presentación debe durar 20/25 minutos y se realizará en la penúltima y última clases (dependiendo del grupo)
  - mandar lista de grupos y papers elegidos de acá a dos semanas (**o sea el 29 de noviembre**). **IMPORTANTE: No elegir papers que tengan que ver con policy gradient. También es posible elegir papers que no estén en la lista.**

Lectura recomendada

- Problemas de reproducibilidad en Deep RL: Deep Reinforcement Learning that Matters, https://arxiv.org/abs/1709.06560