Aprendizaje Reforzado

Maestría en Data Mining, Universidad Austral

Javier Kreiner

Plan de la clase

- Ejemplo reciente
- Repaso
- Aprendizaje on-policy vs. off-policy
- Sarsa
- Q-learning
- Monte Carlo off-policy con importance sampling
- Introducción a aproximación de función de valor

Ejemplo de aplicación reciente de RL

- https://openai.com/blog/solving-rubiks-cube/
- El mayor desafío fue simular ambientes suficientemente variados que capturen la variabilidad del mundo físico
- *Automatic Domain Randomization* (ADR), genera ambientes progresivamente más desafiantes para el algoritmo de manera automática y aleatoria
- Evitar tener que contar con una simulación perfecta del ambiente y permite transferir lo aprendido de la simulación al robot real

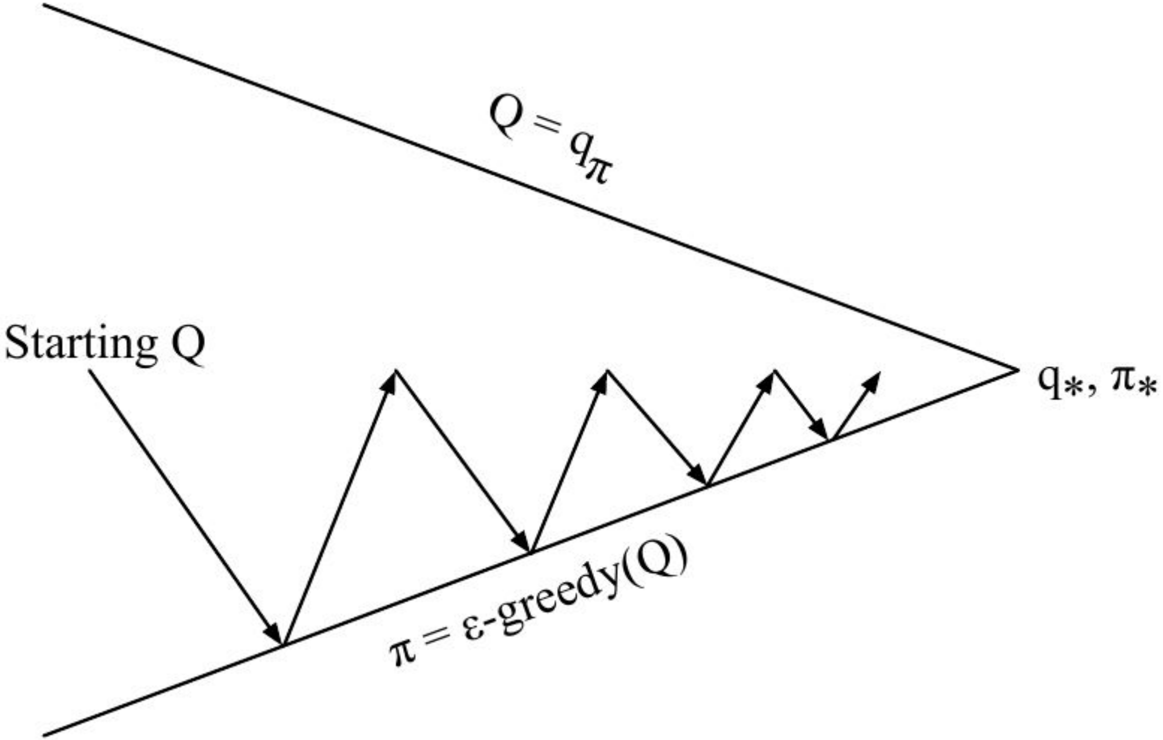
Repaso clase anterior

- Vimos cómo evaluar una política con Monte Carlo
- Lo vimos también con TD
- Además vimos un algoritmo que usa Monte Carlo para calcular la política óptima
- Este método se puede usar para, por ejemplo, calcular la política óptima para jugar al Black Jack

Aprendizaje On-policy vs. aprendizaje Off-policy

- Métodos on-policy aprenden la función de valor de la política que están utilizando
- Vimos que estos métodos en realidad actúan según una política suave (soft-policy) que una parte del tiempo siempre explora
- En los métodos off-policy hay dos políticas, una la cual se actúa (política de comportamiento, con la cual se explora) y otra de la cual se aprende la función de valor y se convierte en la política óptima (política objetivo o target)
- Los métodos off-policy son más generales y poderosos, pero hay que tener cuidado de ajustar los cálculos para estimar correctamente la política target, tienen mayor varianza y demoran más en converger

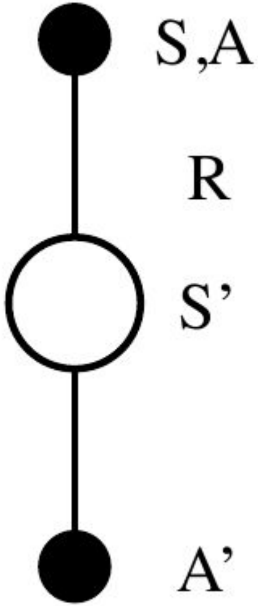
Control (improvement) on-policy con Sarsa



On-policy: tomo acciones con la misma política que estoy mejorando

Sarsa (on-policy + TD)

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S',A') - Q(S,A)\right)$$



Misma idea que TD pero para función de acción-valor.

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

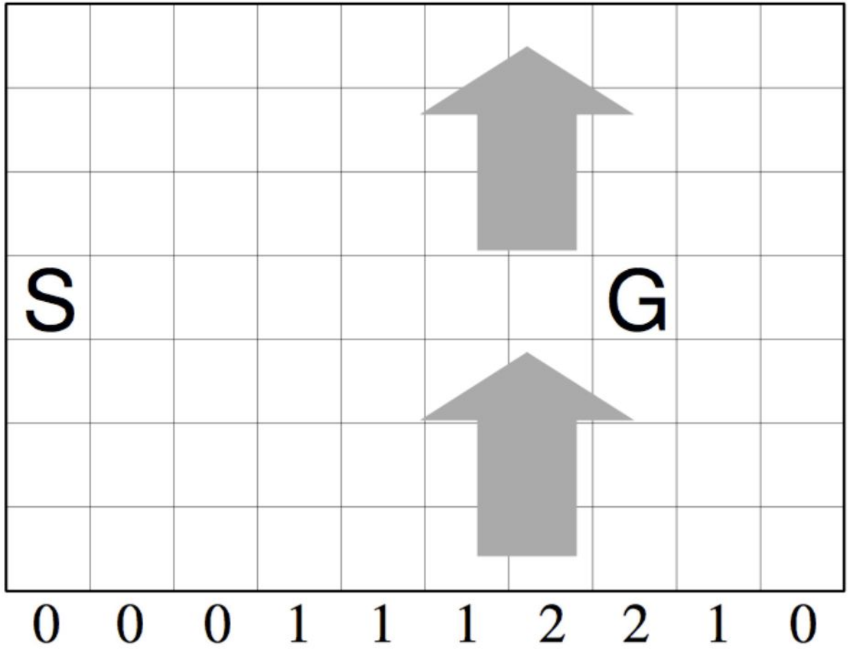
Sarsa - pseudocódigo

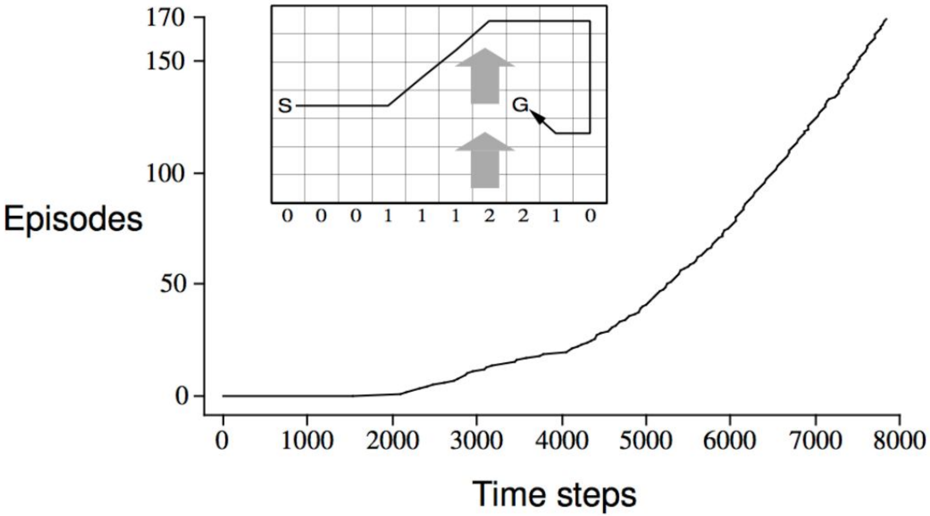
Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Gridworld con viento





Aprendizaje off-policy

Utilizo una política *exploratoria* $\mu(a|s)$ para mejorar la política *óptima* $\pi(a|s)$.

Aprendo observando la experiencia de otros agentes.

¿Cómo mezclar las dos experiencias?

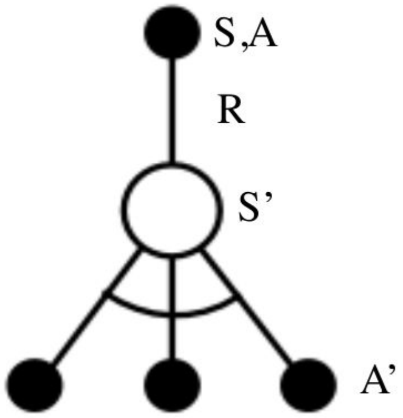
Q-learning

Los episodios los genero con μ pero la estimación del retorno esperado la calculo con una acción tomada con π.

$$A_{t+1} \sim \mu(\cdot|S_t)$$
$$A' \sim \pi(\cdot|S_t)$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t) \right)$$

Off-Policy, Q-learning



Dada $Q^k(s, a)$:

$$\pi_{k+1}(s) = \arg\max_{a'} Q^k(S_t, a'), \qquad \mu_{k+1}(a|s) = \pi^{\varepsilon}_{k+1}.$$

$$Q^{k+1}(S, A) = Q^k(S, A) + \alpha\left(R^+ + \gamma \max_{a'} Q^k(S^+, a') - Q^k(S, A)\right)$$

Q-learning (off-policy + TD)

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Diferencias Temporales (programación)

- Sarsa
- Q-learning

Importance Sampling - Off-policy MC

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

$$V(S_t) \leftarrow V(S_t) + \alpha \left({\color{red} G_t^{\pi/\mu}} - V(S_t) \right)$$

Control Monte Carlo off-policy

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
- $Q(s,a) \in \mathbb{R}$ (arbitrarily)
- $C(s,a) \leftarrow 0$
- $\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
- $b \leftarrow$ any soft policy
- Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- $W \leftarrow 1$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$
  - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
  - If $A_t \neq \pi(S_t)$ then exit For loop
  - $W \leftarrow W \frac{1}{b(A_t|S_t)}$

El espacio de estados...

- El espacio de estados puede ser *gigante*: Backgammon - 10^{20} estados.
- Espacio de estados *continuo*.

⚠ Difícil o imposible guardar $v_\pi(s)$ para todo s!

Idea:

$$v_\pi(s) \approx \hat{v}(s; w)$$

Diferentes aproximantes

- Combinación lineal de features.
- Redes neuronales
- Fourier

En general, varias de las herramientas vistas en supervisado.

A tener en cuenta:
diferenciabilidad y datos no iid.

Gradiente Descendente Estocástico

Busco w tal que

$$J(w) := E_\mu[(v_\pi(S) - \hat{v}(S;w))^2],$$

sea mínimo (μ distribución sobre $\mathcal{S}$).

$$\nabla_w J(w) = -2E_\mu[(v_\pi(S) - \hat{v}(S;w))\nabla_w \hat{v}(S;w)]$$

Stochastic Gradient Descent

$$\begin{aligned} w^{k+1} &= w^k + \Delta w^{k+1} \\ &= w^k + \alpha(v_\pi(S) - \hat{v}(S;w^k))\nabla_w \hat{v}(S;w^k), \end{aligned}$$

$S \sim \mu$.

Repaso

Un paso de evaluación, uno de mejora

Sarsa (on-policy)

$$Q^{k+1}(S,A) = Q^k(S,A) + \alpha(R^+ + \gamma Q^k(S^+,A^+) - Q^k(S,A)),$$

Q-learning (off-policy)

$$Q^{k+1}(S,A) = Q^k(S,A) + \alpha(R^+ + \gamma \max_{a'} Q^k(S^+,a') - Q^k(S,A))$$

con S^+ proveniente de tomar la acción A^+ con la política $\pi_{k+1} = \varepsilon - greedy(Q^k)$.

Aproximación de función de valor

$$J(w) := E_\mu[(v_\pi(S) - \hat{v}(S;w))^2] = \sum_{s \in \mathcal{S}} \mu(s)(v_\pi(s) - \hat{v}(s;w))^2,$$

En lugar de calcular $v_\pi(s), \quad \forall s$, *aproximamos globalmente* controlando los parámetros w.

Recuerdo: Regresión Lineal

$$J(\beta) = E[(Y - f_\beta(X))^2] \approx \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\beta(x_i))^2$$

El cual se puede minimizar realizando Descenso por Gradiente Estocástico (*Batch*)

Descenso por Gradiente Estocástico (SGD)

- Reemplazar una esperanza por una realización
- Reemplazar la función por el target

$$w^{k+1} = w^k + \Delta w^{k+1}$$

TARGET

$$\Delta w^{k+1} = \alpha(q_\pi(S_t, A_t) - \hat{q}(S_t, A_t; w))\nabla_w \hat{q}(S_t, A_t; w)$$

Sarsa- on-policy

$$\approx \alpha(R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) - \hat{q}(S_t, A_t; w))\nabla_w \hat{q}(S_t, A_t; w)$$

Q-learning- off-policy

$$\approx \alpha(R_{t+1} + \gamma \max_{a'} q_\pi(S_{t+1}, a') - \hat{q}(S_t, A_t; w))\nabla_w \hat{q}(S_t, A_t; w)$$

Podemos usar la experiencia que tengamos en más de una pasada de SGD!

Problema (programación)

- Si vamos al casino a jugar al Black Jack y usamos la política óptima, cuál es la ganancia/pérdida de largo plazo?
- Implementar expected-Sarsa, Sutton sección 6.6, es igual Q-learning, pero la ecuación de update es

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big],
\end{aligned}
$$