

APRENDIZAJE REFORZADO (PROFUNDO) Y DEEP Q-NETWORK (DQN)

EL7021: Seminario de robótica y sistemas autónomos

Francisco Leiva² Javier Ruiz-del-Solar^{1,2}

¹Departamento de Ingeniería Eléctrica, Universidad de Chile

²Advanced Mining Technology Center (AMTC), Universidad de Chile

Abril, 2023

Aprendizaje Reforzado

Reinforcement Learning (RL)

Preguntas respondidas en las clases anteriores:

- ▶ ¿Qué es el aprendizaje reforzado?
- ▶ ¿Cuál es el objetivo del aprendizaje reforzado?
- ▶ ¿En qué se diferencia el aprendizaje reforzado del aprendizaje supervisado?
- ▶ ¿Qué es un MDP?
- ▶ ¿Qué es la programación dinámica?

Preguntas adicionales:

- ▶ ¿Como aprender a través de interacciones agente-ambiente?
- ▶ ¿Cuáles son las limitaciones del aprendizaje reforzado?
- ▶ ¿Cómo estas limitaciones pueden ser superadas?

Aprendizaje Reforzado

Reinforcement Learning (RL)

- ▶ ¿Cómo aprender a actuar sin tener acceso a un modelo perfecto del MDP?
 - ▶ Mediante interacciones con el ambiente.

- ▶ ¿Cómo hacer esto?
 - ▶ Métodos de Monte Carlo.
 - ▶ Aprendizaje por diferencia temporal.
 - ▶ n -step Bootstrapping.
 - ▶ Aprender un modelo del ambiente mediante interacciones y luego usarlo para planificar.
 - ▶ Otros...

En lo que sigue se va a presentar brevemente una introducción a métodos de Monte Carlo y de aprendizaje por diferencia temporal.

Métodos de Monte Carlo

Monte Carlo methods

- ▶ La idea general de estos métodos es emplear repetidamente muestreos aleatorios para realizar alguna estimación.
- ▶ En el contexto de RL, se pueden usar para estimar funciones de valor usando las experiencias obtenidas por la interacción agente-ambiente, y usar esto para derivar una política.
- ▶ No requiere conocimiento completo del MDP (función de transición de estados).

- ▶ Como en programación dinámica, los siguientes problemas serán revisados:
 - ▶ *Policy evaluation* (predicción).
 - ▶ *Policy improvement*.
 - ▶ *Policy Iteration* (control).

Monte Carlo Policy Evaluation

Algoritmo 1: Monte Carlo Policy Evaluation (first-visit)

Dar como entrada una política π

Inicializar $V^\pi(s)$ aleatoriamente

Inicializar $S(s) \leftarrow 0$ y $N(s) \leftarrow 0$ para cada $s \in \mathcal{S}$

for $\text{episodio} = 1, N$ **do**

 Generar *rollout* siguiendo política π : $s_1, a_1, r_1, s_2, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T$

$R \leftarrow 0$

for $t=T-1, 1$ **do**

$R \leftarrow r_t + \gamma R$

if $s_t \notin \{s_1, \dots, s_{t-1}\}$ **then**

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

$S(s_t) \leftarrow S(s_t) + R$

end

end

$V^\pi(s) \leftarrow \frac{S(s)}{N(s)}$ para todo $s \in \mathcal{S}$

end

Monte Carlo Policy Evaluation

- ▶ Sin conocer el modelo del ambiente, $V^\pi(s)$ no es suficiente para derivar una política.
- ▶ No obstante, sin conocer un modelo del ambiente, es posible derivar π a partir de $Q^\pi(s, a)$.

- ▶ Para estimar $Q^\pi(s, a)$ usando el enfoque Monte Carlo, se sigue el mismo procedimiento que en *Monte Carlo Policy Evaluation*, con las siguientes consideraciones:
 - ▶ Se visitan pares estado-acción (s, a) .
 - ▶ Pueden existir problemas de exploración.

Monte Carlo Control

Algoritmo 2: Monte Carlo Control (exploring starts, first-visit)

Inicializar una política $\pi(s)$

Inicializar $Q^\pi(s, a)$ aleatoriamente

Inicializar $S(s, a) \leftarrow 0$ y $N(s, a) \leftarrow 0$ para cada $s \in \mathcal{S}$, $a \in \mathcal{A}$

for episodio = 1, N **do**

 Seleccionar aleatoriamente (s_1, a_1)

 Generar *rollout* siguiendo política π , desde (s_1, a_1) : $s_1, a_1, r_1, s_2, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T$

$R \leftarrow 0$

for $t=T-1, 1$ **do**

$R \leftarrow r_t + \gamma R$

if $(s_t, a_t) \notin \{s_1, a_1, \dots, s_{t-1}, a_{t-1}\}$ **then**

$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$

$S(s_t, a_t) \leftarrow S(s_t, a_t) + R$

end

end

$Q^\pi(s, a) \leftarrow \frac{S(s, a)}{N(s, a)}$ para todo $s \in \mathcal{S}$, $a \in \mathcal{A}$

$\pi(s_t) \leftarrow \arg \max_{a \in \mathcal{A}} Q^\pi(s_t, a)$

end

Aprendizaje por diferencia temporal

Temporal-Difference Learning (TD-Learning)

- ▶ TD-Learning es una combinación de ideas entre programación dinámica y métodos de Monte Carlo:
 - ▶ No requiere el conocimiento de un modelo del ambiente.
 - ▶ Actualiza estimaciones en función de otras estimaciones.
- ▶ Como antes, se revisará el problema de predicción y control.

Aprendizaje por diferencia temporal

Temporal-Difference Learning (TD-Learning)

Revisión del problema de predicción:

- ▶ Actualización incremental Monte Carlo:

$$V_{k+1}^{\pi}(s) = V_k^{\pi}(s_t) + \alpha(R - V_k^{\pi}(s_t))$$

- ▶ Requiere experimentar episodios completos para actualizar.

- ▶ Actualización TD-Learning (TD(0)):

$$V_{k+1}^{\pi}(s) = V_k^{\pi}(s_t) + \alpha(r_t + \gamma V_k^{\pi}(s_{t+1}) - V_k^{\pi}(s_t))$$

- ▶ Puede actualizar tras un paso de tiempo.

Policy Evaluation usando TD Learning

Algoritmo 3: TD(0) Policy Evaluation

Dar como entrada una política π

Inicializar $V^\pi(s)$ aleatoriamente

Inicializar parámetro $\alpha \in (0, 1]$

for *episodio* = 1, *N* **do**

 Obtener s_1

for $t=1, T$ **do**

 Seleccionar a_t de acuerdo a la política $\pi(a_t|s_t)$

 Ejecutar acción a_t , observar r_t y s_{t+1}

$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$

end

end

SARSA (On-Policy TD control)

Algoritmo 4: SARSA ϵ -greedy

Inicializar $Q^\pi(s, a)$ aleatoriamente

Inicializar parámetro $\alpha \in (0, 1]$ y $\epsilon > 0$

for *episodio* = 1, *N* **do**

 Obtener s_1

 Con probabilidad ϵ , elegir a_1 aleatoriamente, si no, $a_1 = \arg \max_{a \in \mathcal{A}} Q_\theta(s_1, a)$

for $t=1, T$ **do**

 Ejecutar acción a_t , observar r_t y s_{t+1}

 Con probabilidad ϵ , elegir a_{t+1} aleatoriamente, si no, $a_{t+1} = \arg \max_{a \in \mathcal{A}} Q_\theta(s_{t+1}, a)$

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

end

end

Q-Learning (Off-Policy TD Control)

Algoritmo 5: Q-Learning ϵ -greedy

Inicializar $Q^\pi(s, a)$ aleatoriamente

Inicializar parámetro $\alpha \in (0, 1]$ y $\epsilon > 0$

for episodio = 1, N **do**

 Obtener s_1

for $t=1, T$ **do**

 Con probabilidad ϵ , elegir a_t aleatoriamente, si no, $a_t = \arg \max_{a \in \mathcal{A}} Q_\theta(s_t, a)$

 Ejecutar acción a_t , observar r_t y s_{t+1}

$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t))$

end

end

Aprendizaje Reforzado Profundo

Deep Reinforcement Learning (Deep RL)

- ▶ El aprendizaje reforzado profundo surge de la combinación entre el aprendizaje profundo (*deep learning*) y el aprendizaje reforzado.
- ▶ El *deep* RL se caracteriza por el uso de redes neuronales artificiales como aproximadores funcionales.
- ▶ El primer trabajo que popularizó el uso de redes neuronales en RL fue Deep Q-Network (DQN)^{1, 2}.

¹Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves et al. «Playing atari with deep reinforcement learning». En: *arXiv preprint arXiv:1312.5602* (2013).

²Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu et al. «Human-level control through deep reinforcement learning». En: *nature* 518.7540 (2015), págs. 529-533.

Aprendizaje Reforzado Profundo

Deep Reinforcement Learning (Deep RL)

- ▶ DQN puede considerarse uno de los grandes “*breakthroughs*” en “*inteligencia artificial*” de la década pasada:
 - ▶ Combinando RL y el uso de redes neuronales convolucionales, se logró entrenar políticas que alcanzaron desempeños de nivel humano en una variedad de juegos de Atari 2600.

Aprendizaje Reforzado Profundo

Deep Reinforcement Learning (Deep RL)

- ▶ Tras DQN, una serie de casos de estudio exitosos surgieron aplicando deep RL en otros juegos, por ejemplo:
 - ▶ *Alpha Go*³ y *Alpha Go Zero*⁴.
 - ▶ *OpenAI Five* (Dota II)⁵.
 - ▶ *Alpha Star* (StarCraft II)⁶.
- ▶ Además, la aplicabilidad de deep RL no se restringe a juegos de video (e.g. ver aplicaciones en robótica [navegación, manipulación, etc]).

³David Silver, Aja Huang et al. «Mastering the game of Go with deep neural networks and tree search». En: *nature* 529.7587 (2016), págs. 484-489.

⁴David Silver, Julian Schrittwieser et al. «Mastering the game of go without human knowledge». En: *nature* 550.7676 (2017), págs. 354-359.

⁵Christopher Berner et al. «Dota 2 with large scale deep reinforcement learning». En: *arXiv preprint arXiv:1912.06680* (2019).

⁶Oriol Vinyals et al. «Grandmaster level in StarCraft II using multi-agent reinforcement learning». En: *Nature* 575.7782 (2019), págs. 350-354.

Deep Q-Network (DQN)

- ▶ DQN emplea una red neuronal $Q_{\theta}(s, a)$ para aproximar $Q^*(s, a)$.
- ▶ Esta red neuronal, $Q_{\theta}(s, a)$, es entrenada empleando una variante de Q-Learning que posee dos elementos fundamentales:
 - ▶ El uso de un *Experience Replay*.
 - ▶ El uso de *Target Networks*.

Deep Q-Network (DQN)

- Como en Q-Learning, para aproximar $Q^*(s, a)$ se emplea la ecuación definida en (1) como regla de actualización.

$$Q^*(s, a) = r(s, a) + \gamma \max_{a' \in \mathcal{A}} \mathbb{E}_{s' \sim p(s'|s, a)} Q^*(s', a') \quad (1)$$

- Para lograr esto, se trata de minimizar el error cuadrático medio, calculado al forzar que $Q_\theta(s, a)$ cumpla con (1):

$$L_i(\theta_i) = \mathbb{E}_{(s, a) \sim p_\pi(s, a)} \left[\left(r(s, a) + \gamma \max_{a' \in \mathcal{A}} \mathbb{E}_{s' \sim p(s'|s, a)} Q_{\theta_{i-k}}(s', a') - Q_{\theta_i}(s, a) \right)^2 \right]$$

Deep Q-Network (DQN)

- Definiendo $y = r(s, a) + \gamma \max_{a' \in \mathcal{A}} \mathbb{E}_{s' \sim p(s'|s, a)} Q_{\theta_{i-k}}(s', a')$, entonces:

$$L_i(\theta_i) = \mathbb{E}_{(s, a) \sim p_\pi(s, a)} \left[(y - Q_{\theta_i}(s, a))^2 \right]$$

- Derivando con respecto a θ_i , es posible minimizar los costos $L_i(\theta_i)$ siguiendo la dirección del gradiente dado por:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{(s, a) \sim p_\pi(s, a)} \left[(y - Q_{\theta_i}(s, a)) \nabla_{\theta_i} Q_{\theta_i}(s, a) \right]$$

En Q-Learning, esta actualización es realizada “*on-line*”, lo que trae problemas de estabilidad.

Deep Q-Network (DQN)

Experience Replay

- ▶ Las experiencias (s_t, a_t, r_t, s_{t+1}) son almacenadas en un *buffer* finito D .
- ▶ *Batches* de experiencias son muestreados uniformemente de D para actualizar los parámetros de $Q_\theta(s, a)$:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} [(y - Q_{\theta_i}(s, a)) \nabla_{\theta_i} Q_{\theta_i}(s, a)]$$

- ▶ Lo anterior permite la reutilización de experiencias, y también romper la correlación que existe entre ellas.

Deep Q-Network (DQN)

Target Network

- ▶ Se utiliza una copia $Q_{\bar{\theta}}(s, a)$ de $Q_{\theta}(s, a)$ para el cómputo de y .
- ▶ Esta copia es actualizada copiando los parámetros de $Q_{\theta}(s, a)$ cada $k > 1$ actualizaciones.
- ▶ Este mecanismo reduce la posibilidad de oscilaciones (o divergencia) en los parámetros θ , al introducir un “*delay*” en el efecto que tienen los cambios en $Q_{\theta}(s, a)$ en el cómputo de y .

Deep Q-Network (DQN)

- ▶ Al igual que Q-Learning, DQN es “*off-policy*”.
- ▶ para explorar, emplea una estrategia ϵ -greedy:
 - ▶ Con probabilidad ϵ se selecciona una acción aleatoria.
 - ▶ Con probabilidad $1 - \epsilon$ se selecciona una acción “avara”,
$$a = \arg \max_{a' \in \mathcal{A}} Q_{\theta}(s, a').$$
- ▶ Notar que para calcular $a = \arg \max_{a' \in \mathcal{A}} Q_{\theta}(s, a')$ se requeriría evaluar $|\mathcal{A}|$ veces a la red $Q_{\theta}(s, a)$.
- ▶ DQN simplifica el proceso de selección de acciones al considerar $|\mathcal{A}|$ salidas (una por acción posible), dada una entrada s .
- ▶ Con esto, la selección de acciones solo requiere la evaluación de $Q_{\theta}(s, a)$, pues se elige aquella acción asociada a la mayor salida.

Deep Q-Network (DQN)

Algoritmo 6: Deep Q-Network

Inicializar $Q_\theta(s, a)$ con parámetros θ

Inicializar $Q_{\bar{\theta}}(s, a)$ con parámetros $\bar{\theta} \leftarrow \theta$

Inicializar *replay memory* D

for *episodio* = 1, M **do**

 Obtener s_1

for $t=1, T$ **do**

 Con probabilidad ϵ , elegir a_t aleatoriamente, si no, $a_t = \arg \max_{a \in \mathcal{A}} Q_\theta(s_t, a)$

 Ejecutar acción a_t , observar r_t y s_{t+1}

 guardar transición (s_t, a_t, r_t, s_{t+1}) en D

 Muestrear un *minibatch* de N transiciones (s_j, a_j, r_j, s_{j+1}) de D

 Calcular $y_j = \begin{cases} r_j & \text{si } s_{j+1} \text{ es un estado terminal,} \\ r_j + \gamma \max_{a'} Q_{\bar{\theta}}(s_{j+1}, a') & \text{si no.} \end{cases}$

 Actualizar $Q_\theta(s, a)$ minimizando el costo $L(\theta) = \frac{1}{N} \sum_{j=1}^N (y_j - Q_\theta(s_j, a_j))^2$

 Cada C actualizaciones de $Q_\theta(s, a)$, $\bar{\theta} \leftarrow \theta$

end







end

Deep Q-Network (DQN)

Comentarios

- ▶ No es trivial usar DQN en problemas donde se requiere control continuo.
- ▶ No escala necesariamente bien para conjuntos de acciones con alta cardinalidad.
- ▶ Existen otros algoritmos adecuados para problemas con esta clase de características, que derivan (o no) de DQN.

Referencias

-  Berner, Christopher et al. «Dota 2 with large scale deep reinforcement learning». En: *arXiv preprint arXiv:1912.06680* (2019).
-  Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves et al. «Playing atari with deep reinforcement learning». En: *arXiv preprint arXiv:1312.5602* (2013).
-  Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu et al. «Human-level control through deep reinforcement learning». En: *nature* 518.7540 (2015), págs. 529-533.
-  Silver, David, Aja Huang et al. «Mastering the game of Go with deep neural networks and tree search». En: *nature* 529.7587 (2016), págs. 484-489.
-  Silver, David, Julian Schrittwieser et al. «Mastering the game of go without human knowledge». En: *nature* 550.7676 (2017), págs. 354-359.
-  Vinyals, Oriol et al. «Grandmaster level in StarCraft II using multi-agent reinforcement learning». En: *Nature* 575.7782 (2019), págs. 350-354.