
Aprendizaje por Refuerzos

— Diplomatura en Ciencia de Datos, Aprendizaje
Automático y sus Aplicaciones - FaMAF, 2021 —

Agenda

- Métodos de Solución Aproximada y Generalización en RL
- Aproximación Funcional y SGD
- Redes Neuronales: Deep Q-networks
- Deep Q-learning
- Policy Gradient

Crédito: Libro Reinforcement Learning - An Introduction. Sutton & Barto, 2018

Métodos de Solución Aproximada y Generalización en RL

Métodos de Solución Aproximada

- Extensión de los métodos tabulares para aplicarlos a problemas con espacios de estados arbitrariamente grandes
 - Espacio de estados: Infinitos o Finitos pero demasiado grandes.
 - Espacio de acciones: Discretos y Continuos.
- No se busca una política óptima o una función de valor óptima, sino que generamos buenas soluciones aproximadas empleando recursos computacionales limitados.

Generalización en RL (1)

- El problema de los espacios de estados de gran tamaño, no es solo la memoria necesaria para almacenar las tablas de valor, sino el tiempo y los datos necesarios para llenarlas de manera correcta.

Pregunta:

¿Cómo se puede generalizar de manera útil la experiencia en base a un número limitado de ejemplos?

Generalización en RL (2)

Respuesta: **Aproximación de Funciones**

La aproximación funcional es una instancia del **Aprendizaje Supervisado (Machine Learning)**. En teoría, cualquiera de los métodos estudiados en dichas áreas podrían emplearse con algoritmos **RL**, aunque en la práctica algunos de ellos son mejores que otros.

Aproximación Funcional y SGD


Aproximación Funcional en RL


- Se utilizan para la estimación de \mathbf{V} , \mathbf{Q} o π .
- Las funciones mencionadas no se representan como tablas sino en forma de funciones parametrizadas con un vector de pesos $\mathbf{w} \in \mathbf{R}^d$.
- Como consecuencia, cuando un estado se actualiza, el cambio se generaliza desde ese estado para afectar los valores de muchos otros estados.
- $V_{\text{aprox}}(s, \mathbf{w}) \approx V_{\pi}(s)$

Stochastic gradient-descent (SGD) (1)

- El vector de pesos es un vector columna con un número fijo de componentes reales $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d)$ y la función de valor aproximada $V_{\text{aprox}}(\mathbf{s}, \mathbf{w})$ es una función diferenciable de \mathbf{w} para todo $\mathbf{s} \in \mathbf{S}$.
- SGD actualiza \mathbf{w} en cada uno de los pasos $\mathbf{t} = 0, 1, 2, 3, \dots, \mathbf{n}$ de interacción intentando minimizar el error de predicción respecto de los ejemplos provenientes de la experimentación con el entorno.

Step-size


$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \frac{1}{2}\alpha \nabla \left[v_{\pi}(S_t) - \hat{v}(S_t, \mathbf{w}_t) \right]^2 \\ &= \mathbf{w}_t + \alpha \left[v_{\pi}(S_t) - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t)\end{aligned}$$

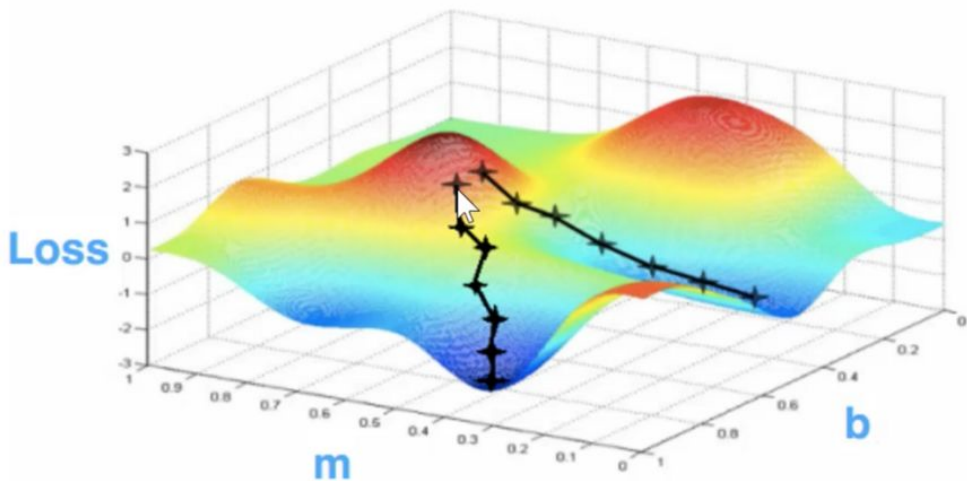


El vector de derivativas es el *gradiente* de f respecto de \mathbf{w} $\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^{\top}$

Stochastic gradient-descent (SGD) (2)

Gradient Descent

$f(x)$ = nonlinear function of x



Bellman Update

- Bellman Update:

$$NewQ(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$$

New Q value for that state and that action

Current Q value

Reward for taking that action at that state

Maximum expected future reward given the new s' and all possible actions at that new state

Learning Rate

Discount rate

$$\Delta w = \alpha [(R + \gamma \max_a \hat{Q}(s', a, w)) - \hat{Q}(s, a, w)] \nabla_w \hat{Q}(s, a, w)$$

Change in weights

learning rate

Maximum possible Qvalue for the next_state (= Q_target)

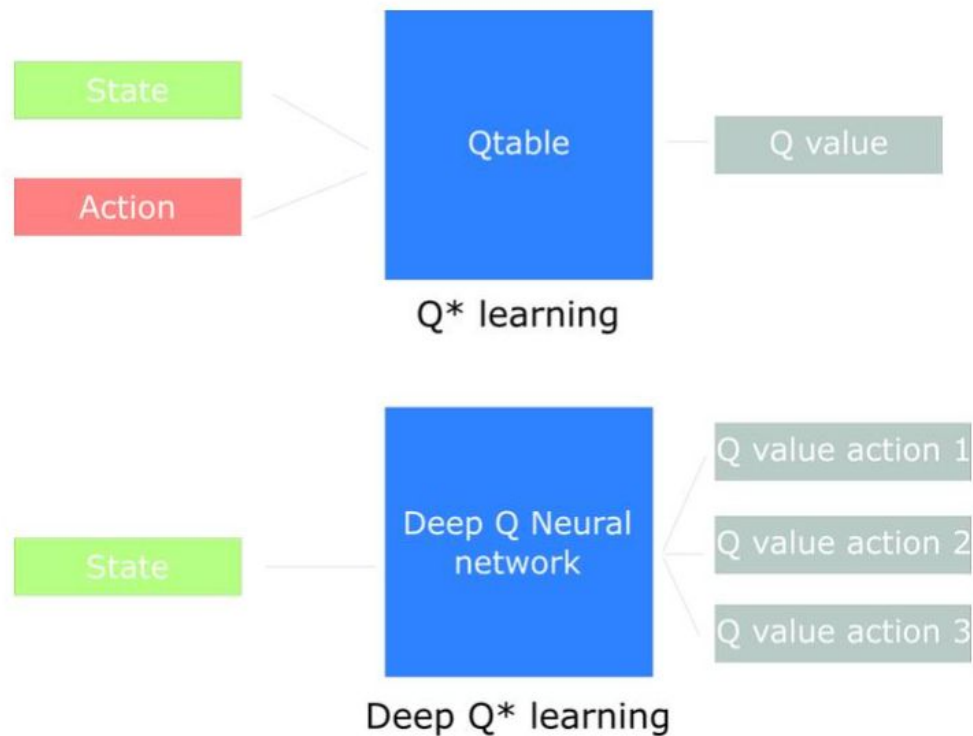
Current predicted Q-val

TD Error

Gradient of our current predicted Q-value

Deep Q-Learning

Q-Learning VS Deep Q-Learning



Deadly Triad

- **Function approximation**
- **Bootstrapping Update:** Objetivos que incluyen estimaciones existentes en lugar de depender exclusivamente de recompensas reales y retornos completos.
- **Off-policy training:** Entrenamiento en una distribución de transiciones diferente a la producida por la política objetivo.

Deep Q-Learning - Target Network (1)

The diagram illustrates the weight update equation for Deep Q-Learning, showing the relationship between the current network and the target network.

Weight Update Equation:

$$\Delta w = \alpha [(R + \gamma \max_a \hat{Q}(s', a, w)) - \hat{Q}(s, a, w)] \nabla_w \hat{Q}(s, a, w)$$

Annotations for the Weight Update Equation:

- Change in weights:** Δw
- learning rate:** α
- Maximum possible Qvalue for the next_state (= Q_target):** $R + \gamma \max_a \hat{Q}(s', a, w)$
- Current predicted Q-val:** $\hat{Q}(s, a, w)$
- TD Error:** The difference between the maximum possible Qvalue and the current predicted Q-val.
- Gradient of our current predicted Q-value:** $\nabla_w \hat{Q}(s, a, w)$
- Es una estimación!!!** (It is an estimation!!!): Points to $\hat{Q}(s', a, w)$.
- El mismo conjunto de parámetros** (The same set of parameters): Points to $\hat{Q}(s', a, w)$ and $\hat{Q}(s, a, w)$.

Target Network Equation:

$$Q(s, a) = r(s, a) + \gamma \max_a Q(s', a)$$

Annotations for the Target Network Equation:

- Q target:** $Q(s, a)$
- Reward of taking that action at that state:** $r(s, a)$
- Discounted max q value among all. possibles actions from next state.** $\gamma \max_a Q(s', a)$

Deep Q-Learning - Target Network (2)

$$\Delta w = \alpha [(R + \gamma \max_a \hat{Q}(s', a, \vec{w}^-)) - \hat{Q}(s, a, w)] \nabla_w \hat{Q}(s, a, w)$$

Change in
weights

learning
rate

Maximum possible Qvalue for the
next_state (= Q_target)

Current predicted
Q-val

TD Error

Gradient of our current
predicted Q-value

At every T steps:

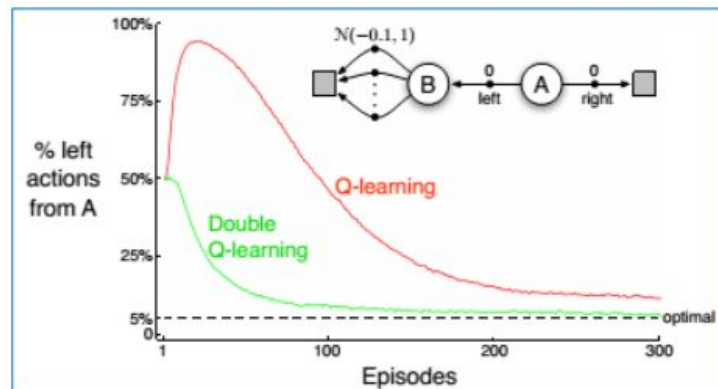
$$\vec{w}^- \leftarrow w$$

Update fixed parameters

Artículo: <http://www.nature.com/articles/nature14236>

Mejoras a Deep Q-Learning - Double DQNs (1)

- Al inicio del entrenamiento no tenemos suficiente información respecto de qué acciones elegir, por lo que tomar los mejores valores de Q puede llevar a “falsos positivos” y a demorar la convergencia a largo plazo.
- Solución:
 - Cuando calculamos el Q target, empleamos dos redes para desacoplar la selección de acciones de la generación del Q target.



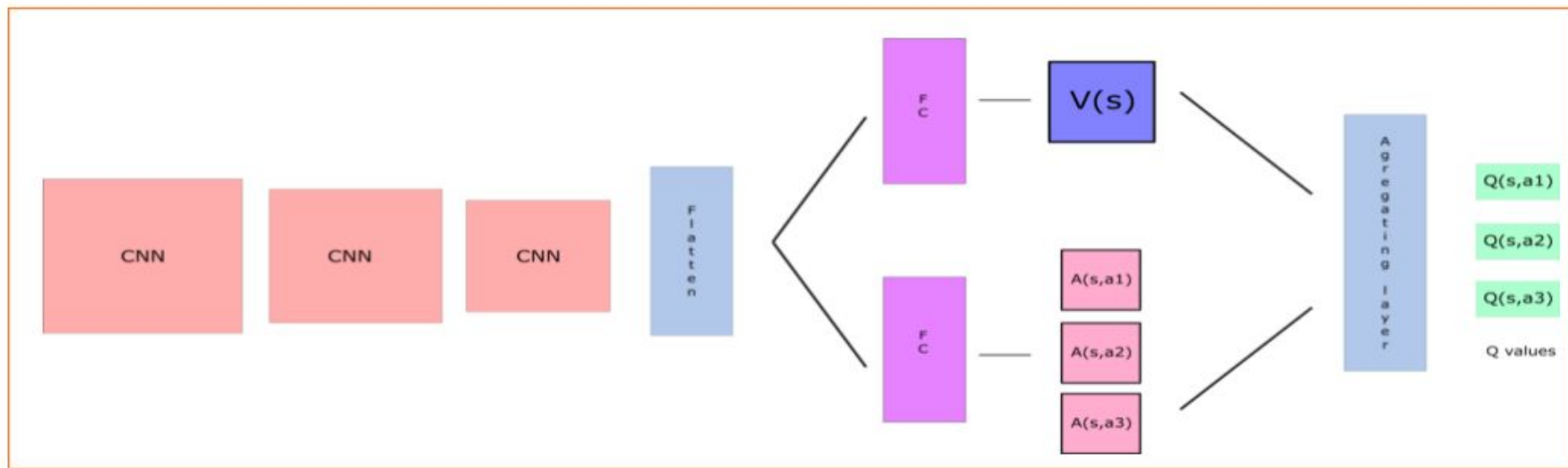
Mejoras a Deep Q-Learning - Double DQNs (2)

$$Q(s,a) = Q(s,a) + \alpha [r(s,a) + \gamma Q_{w1}(s', \operatorname{argmax}_a Q_{w2}(s',a)) - Q_{w1}(s, a)]$$

- La red Q_{w2} selecciona cuál es la mejor acción en el siguiente estado (acción con mayor valor esperado).
- La red Q_{w1} calcula el valor objetivo (target) correspondiente a tomar esa acción en el estado siguiente.

Artículo: <http://arxiv.org/abs/1509.06461>

Mejoras a Deep Q-Learning - Dueling Networks (1)



$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha) \right)$$

Mejoras a Deep Q-Learning - Dueling Networks (2)

¿Por qué calculamos $A(s,a)$ y $V(s)$ separadamente, para luego combinarlos?

- Desacoplando la estimación de $A(s,a)$ y $V(s)$, se puede aprender cuáles estados son valorables sin tener que aprender el efecto de cada acción en cada estado.
- Si el estado es de por sí “malo”, calcular el valor de las acciones no aporta al aprendizaje.
- Al calcular $V(s)$, no es necesario calcular el valor de cada acción lo que es particularmente útil en aquellos estados en que no es relevante la acción que se toma.

Artículo: <http://arxiv.org/abs/1511.06581>

Policy Gradient Methods

Policy Gradient Methods (1)

- Son métodos que “aprenden” directamente una política parametrizada $\pi(\theta)$.
- Se debe definir una métrica de performance $J(\theta)$.
- Estos métodos tratan de maximizar $J(\theta)$ por lo que utilizan gradiente ascendente (en vez de descendente)

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

- Para asegurar la exploración del entorno, $\pi(\mathbf{a} | \mathbf{s}, \theta)$ es no determinística.

Policy Gradient Methods (2)

- Ventajas:
 - $\pi(\theta)$ puede aproximar una política determinista. Por el contrario, en una política ε -greedy, siempre existe una probabilidad ε de elegir una acción aleatoriamente.
 - Al tener una política estocástica, puede existir más de 1 acción “óptima”.
- Desventajas:
 - Son métodos “On-Policy”

REINFORCE: Monte Carlo Policy Gradient (1)

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &\propto \mathbb{E}_{\pi} \left[\sum_a \pi(a|S_t, \boldsymbol{\theta}) q_{\pi}(S_t, a) \frac{\nabla \pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \right] \\ &= \mathbb{E}_{\pi} \left[q_{\pi}(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} \right] && \text{(replacing } a \text{ by the sample } A_t \sim \pi) \\ &= \mathbb{E}_{\pi} \left[G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} \right], && \text{(because } \mathbb{E}_{\pi}[G_t|S_t, A_t] = q_{\pi}(S_t, A_t))\end{aligned}$$

- Donde:
 - S_t : es el estado en el instante t .
 - A_t : es la acción seleccionada en el instante t .
 - G_t : es el retorno total hasta el instante t .

REINFORCE: Monte Carlo Policy Gradient (2)

La actualización de los parámetros θ del aproximador es:

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)}$$

Cada incremento es proporcional al producto del retorno G_t y el vector gradiente de la probabilidad de tomar la acción realmente tomada dividido por la probabilidad de tomar esa acción.

Utiliza el retorno completo hasta el instante t , que incluye todas las recompensas futuras hasta el final del episodio. En este sentido, REINFORCE es un algoritmo de tipo Monte Carlo y está bien definido sólo para problemas episódicos.

REINFORCE with baseline

El teorema del gradiente de política se puede generalizar para incluir una comparación del valor de la acción con un valor arbitrario llamado “baseline”.

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \left(G_t - b(S_t) \right) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)}.$$

- El baseline deja el valor esperado de la actualización sin cambios, pero puede tener un gran efecto en su varianza.
- Este debe variar con el estado: en algunos estados todas las acciones tienen valores altos y necesitamos una línea de base alta para diferenciar las acciones de mayor valor de las menos valoradas y viceversa.
- Una opción natural para utilizar como baseline es una estimación de la función de valor $V_{aprox}(S_t; w)$.

Actor-Critic Methods

Componentes:

- Actor: Selecciona las acciones a tomar en el estado S.
- Crítico: Evalúa la acción elegida por el Actor.

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha \left(G_{t:t+1} - \hat{v}(S_t, \mathbf{w}) \right) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \underbrace{\gamma \hat{v}(S_{t+1}, \mathbf{w})}_{\text{Bootstrapping}} - \hat{v}(S_t, \mathbf{w}) \right) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha \delta_t \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)}.\end{aligned}$$

Actor-Critic Methods (2)

Ventajas (Con respecto a Reinforce):

- Al hacer bootstrapping se introduce bias al estimador, lo cual, reduce la varianza del mismo y acelera el aprendizaje.
- Se puede utilizar en problemas continuos (no episódicos).
- Artículos Relevantes:
 - Actor-Critic with Experience Replay (ACER) - <https://arxiv.org/pdf/1611.01224>
 - Asynchronous Advantage Actor-Critic (A3C) - <http://arxiv.org/abs/1602.01783>
 - Trust Region Policy Optimization (TRPO) - <http://arxiv.org/abs/1502.05477>
 - Proximal Policy Optimization (PPO) - <https://arxiv.org/pdf/1707.06347>

Fin

¿Preguntas?