

Aprendizaje por refuerzo

Clase 4: Programación dinámica

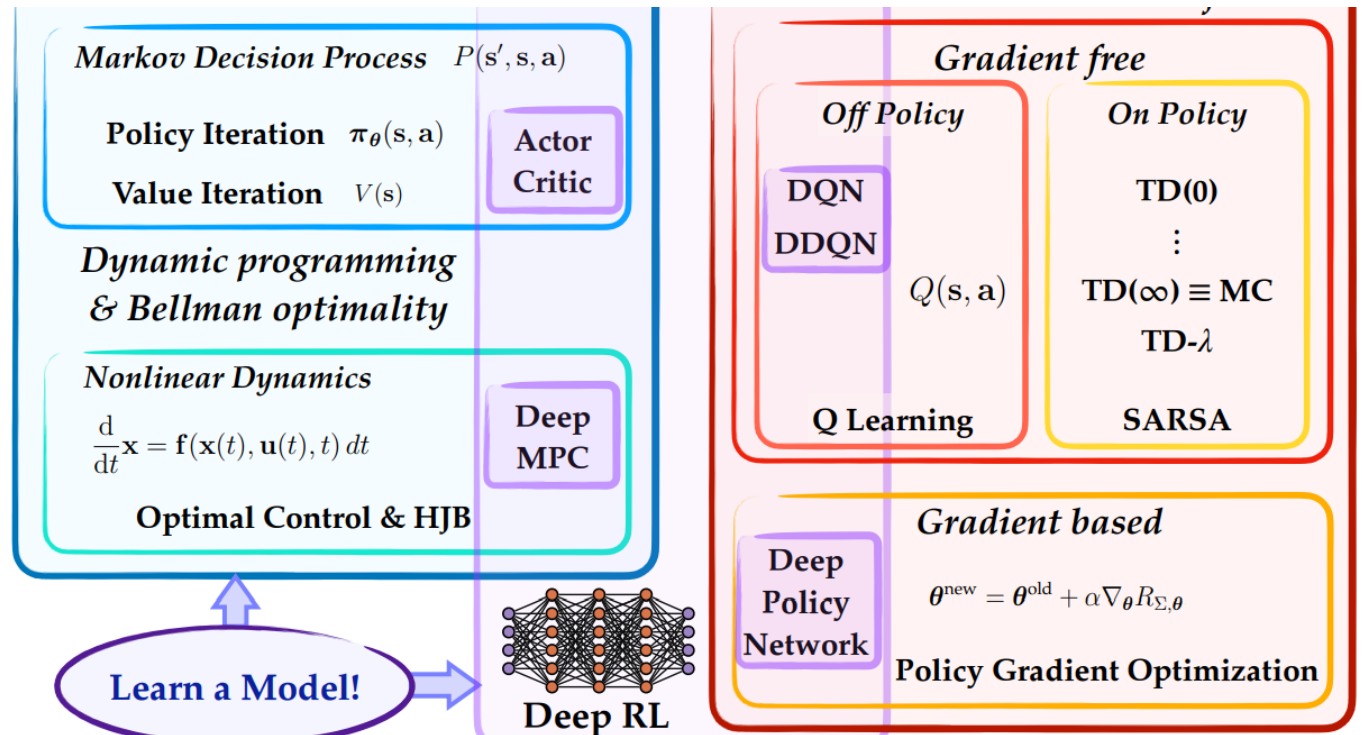


Para el día de hoy...

- Evaluación de una política
- Iteración de valor
- Iteración de política



Comencemos el camino...



Programación dinámica



Descompone los
problemas en
subproblemas

Resuelve los
subproblemas
Une las soluciones



Se refiere a un conjunto de algoritmos
que pueden ser utilizados para encontrar
políticas óptimas dado un MDP



Son métodos resolver problemas
complejos



Los métodos
consisten en dos
partes

Evaluación de una
política
Mejora de una política

Ecuaciones de Bellman (esperadas)

- Dado un MDP, $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, para cualquier política π , las funciones de valor obedecen las siguientes ecuaciones

$$v_{\pi}(s) = \sum_a \pi(a|s) [r(s, a) + \gamma \sum_{s'} p(s'|a, s) v_{\pi}(s')]$$

$$q_{\pi}(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|a, s) \sum_{a'} \pi(a'|s) q_{\pi}(s', a')$$

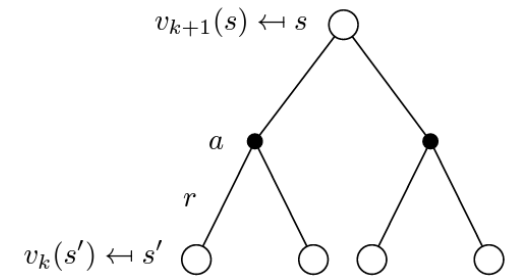
Ecuaciones de Bellman (óptimas)

- Dado un MDP, $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, para la política óptima π , las funciones de valor obedecen las siguientes ecuaciones

$$v^*(s) = \max_a \left[r(s, a) + \gamma \sum_{s'} p(s'|a, s) v^*(s') \right]$$

$$q^*(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|a, s) \max_{a' \in \mathcal{A}} q^*(s', a')$$

Evaluación de una política



Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π , the policy to be evaluated

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$




$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

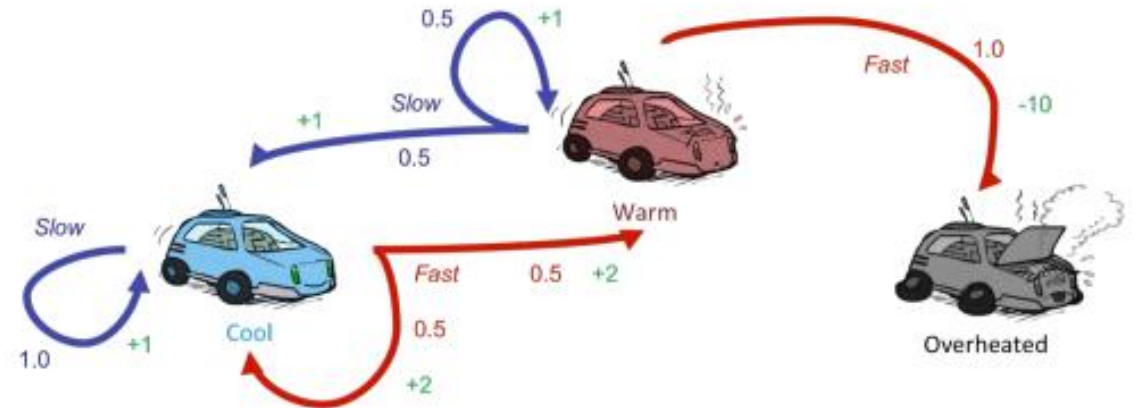
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

Veamos nuestro ejemplo

- $\forall s: v_{\pi}(s) = \sum_a \pi(a|s)[r(s, a) + \gamma \sum_{s'} p(s'|a, s)v_{\pi}(s')]$
- $\gamma = 1$
- $v_{\pi}(\text{Cool}) \rightarrow \text{Fast}; v_{\pi}(\text{Warm}) \rightarrow \text{Slow};$

			
V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0

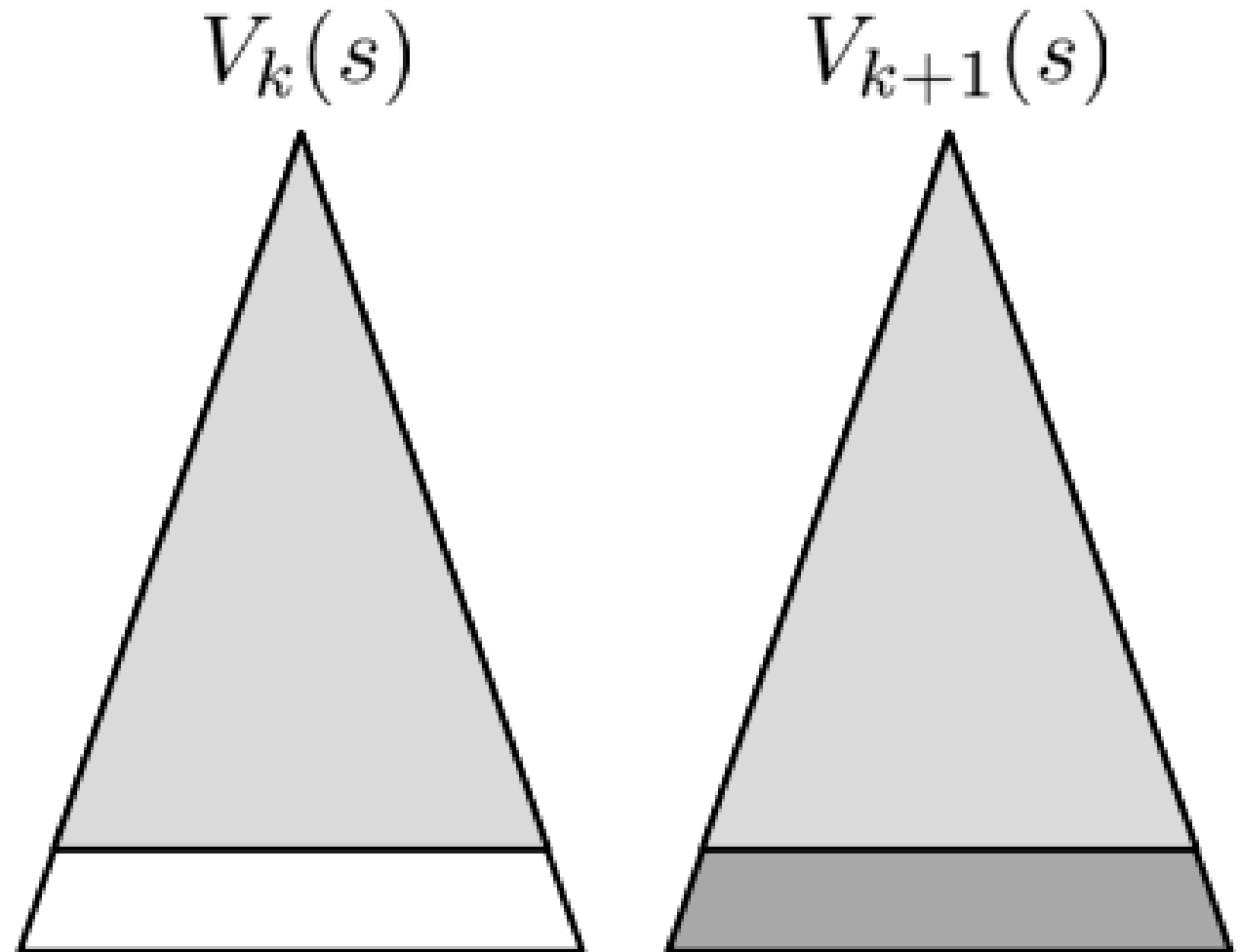


Política óptima

- Define un orden parcial entre políticas
- $\pi \geq \pi' \leftrightarrow v_{\pi}(s) \geq v_{\pi'}(s), \forall s$
- Para cualquier MDP
 - Existe una política óptima π^* tal que $\pi^* \geq \pi, \forall \pi$
 - $v^{\pi^*}(s) = v^*(s)$
 - $q^{\pi^*}(s, a) = q^*(s, a)$

Borrador de prueba de convergencia

- Caso 1: Si el árbol tiene una profundidad máxima M , entonces v_M tendrá los valores reales
- Case 2: si el descuento es menor que 1
 - Para cualquier estado V_k y v_{k+1} se puede ver como árboles casi idénticos de tamaño $k + 1$
 - En la capa del fondo v_{k+1} tiene recompensas mientras que v_k tiene ceros
 - La última capa es en el mejor de los casos R_{\max} y en el peor de los casos R_{\min}
 - Pero todo descontado por γ^k
 - Entonces, v_k y v_{k+1} son a lo más $\gamma^k \max |R|$ diferentes
 - Mientras k incrementa, los valores convergen



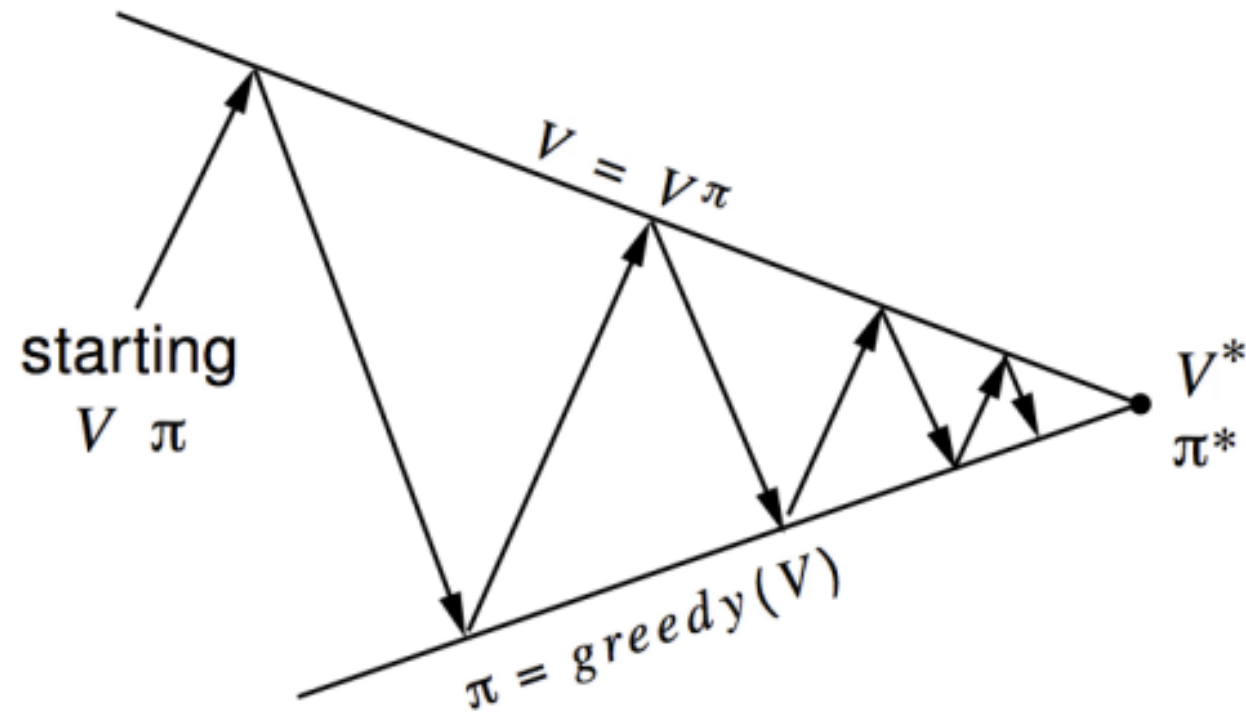
Mejorando la política

- Hasta ahora sabemos como evaluar una política π dada
- ¿Podemos utilizar esa información para mejorar la política?

Mejorando la política

- Algoritmo
 - Iterar utilizando
 - $\forall s: \pi_{new}(s) = \operatorname{argmax}_a q_{\pi(s,a)} = \operatorname{argmax}_a \mathbb{E}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s, A_t = a]$
 - Después, evaluar π_{new} y repetir
- Nota: es posible demostrar que $v_{\pi_{new}}(s) \geq v_{\pi}(s) \forall s$

Iteración de política



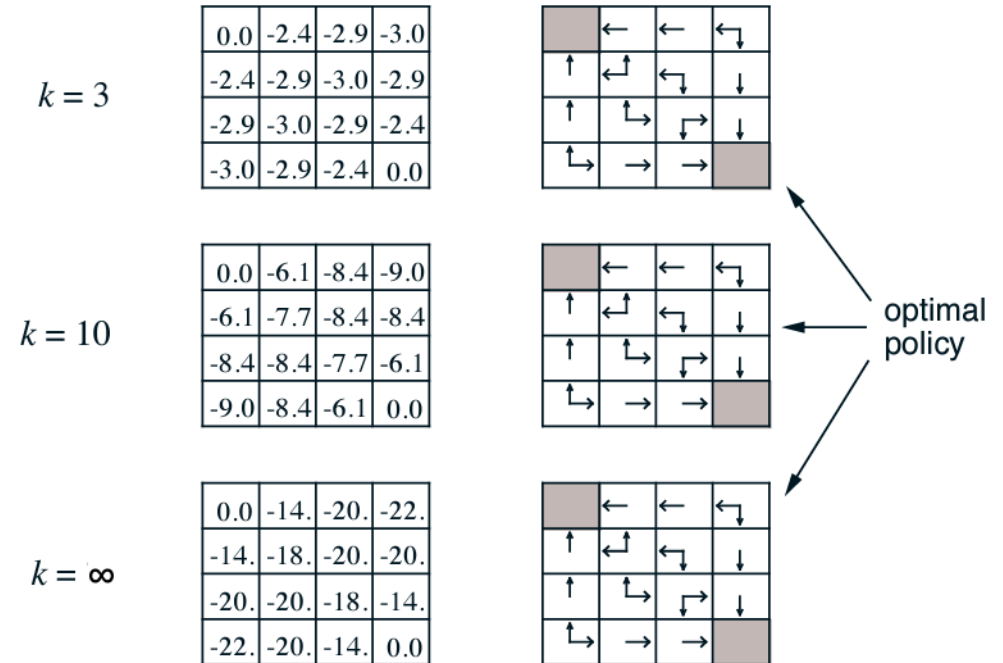
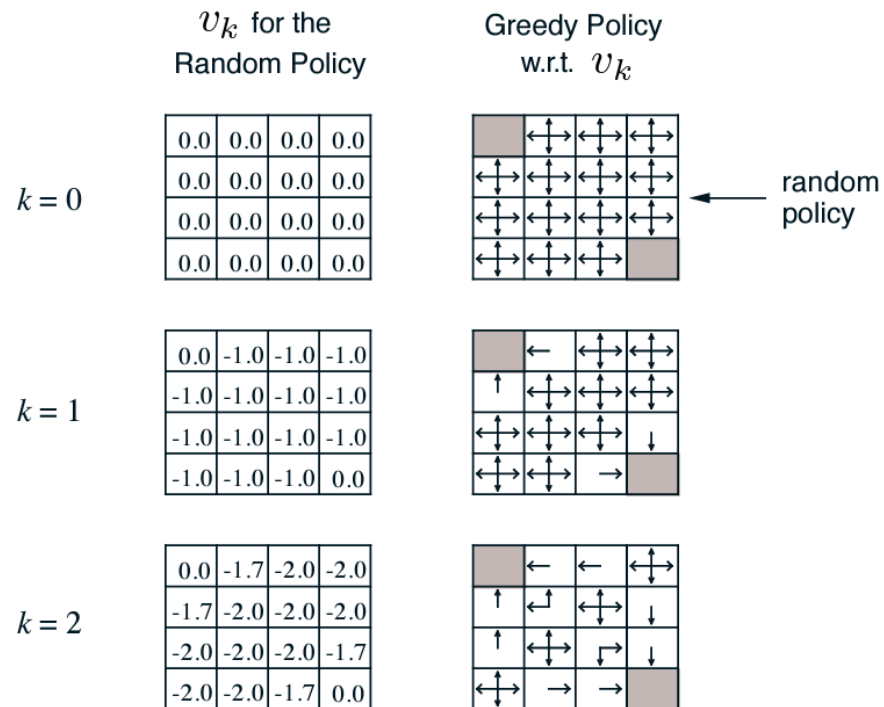
- Evaluación de política: estimar v_π
- Mejora de política: generar $\pi' \geq \pi$

Algoritmo Iteración de política

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
 $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
2. Policy Evaluation
Loop:
 $\Delta \leftarrow 0$
 Loop for each $s \in \mathcal{S}$:
 $v \leftarrow V(s)$
 $V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
 until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)
3. Policy Improvement
 policy-stable \leftarrow true
 For each $s \in \mathcal{S}$:
 old-action $\leftarrow \pi(s)$
 $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$
 If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false
 If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Ejemplo



Notas del algoritmo

- ¿Necesitamos que converja a v_π ?
- ¿Podemos detenernos antes?
 - ¿Después de 10 pasos?
 - ¿De 5?
 - ¿1?

Iteración de valor

- Podemos tomar la ecuación de Bellman y convertirla en una actualización

$$\forall s: v_{k+1}(s) \leftarrow \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a]$$

- Esto es equivalente a iteración de política con 1 paso para la evaluación de la política

Algoritmo iteración de valor

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$   
| Loop for each  $s \in \mathcal{S}$ :  
|    $v \leftarrow V(s)$   
|    $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$   
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$   
until  $\Delta < \theta$ 
```

Output a deterministic policy, $\pi \approx \pi_*$, such that
$$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

Observaciones

- Los algoritmos basados en función de valor $v_\pi(s)$ o $v^*(s)$ tienen una complejidad de $O(|\mathcal{A}||\mathcal{S}|^2)$ por iteración
- También se puede basar en funciones de acción $q_\pi(s)$ o $q^*(s)$ tienen una complejidad de $O(|\mathcal{A}|^2|\mathcal{S}|^2)$ por iteración

Problema	Ecuación de Bellman	Algoritmo
Predicción	Esperada	Evaluación de política iterativa
Control	Esperada + mejora de política	Iteración de política
Control	Óptima	Iteración de valor

Extensiones

- Hasta ahora hemos utilizado actualizaciones síncronas
- Asincronía
 - Actualizar los estados en cualquier orden
 - Puede reducir los cálculos
 - Garantiza convergencia si todos los estados son seleccionados

Programación dinámica asíncrona

Programación dinámica en lugar

Priorización

Tiempo real

Programación dinámica en lugar

- En la versión síncrona tenemos dos copias de la función de valor

$$\forall s: v_{new}(s) \leftarrow \max_a \mathbb{E}[R_{t+1} + \gamma v_{old}(S_t + 1) | S_t | s]$$

- Podemos utilizar solo una copia

$$\forall s: v(s) \leftarrow \max_a \mathbb{E}[R_{t+1} + \gamma v(S_t + 1) | S_t | s]$$

Uso de prioridades

- Utilizar la magnitud del error de Bellman para guiar la selección
$$|\max_a \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = S] - v(s)|$$
- Actualizar el estado con el error más grande
- Actualizar el error de los estados afectados en cada paso
- Requiere conocimiento de la dinámica inversa
- Puede ser implementado mediante una cola de prioridad

Programación dinámica en tiempo real

- Idea: actualizar solamente los estados relevantes para el agente
- Si un agente está en el estado S_t , actualizar ese estado y los estados en que se espera que se encuentre pronto

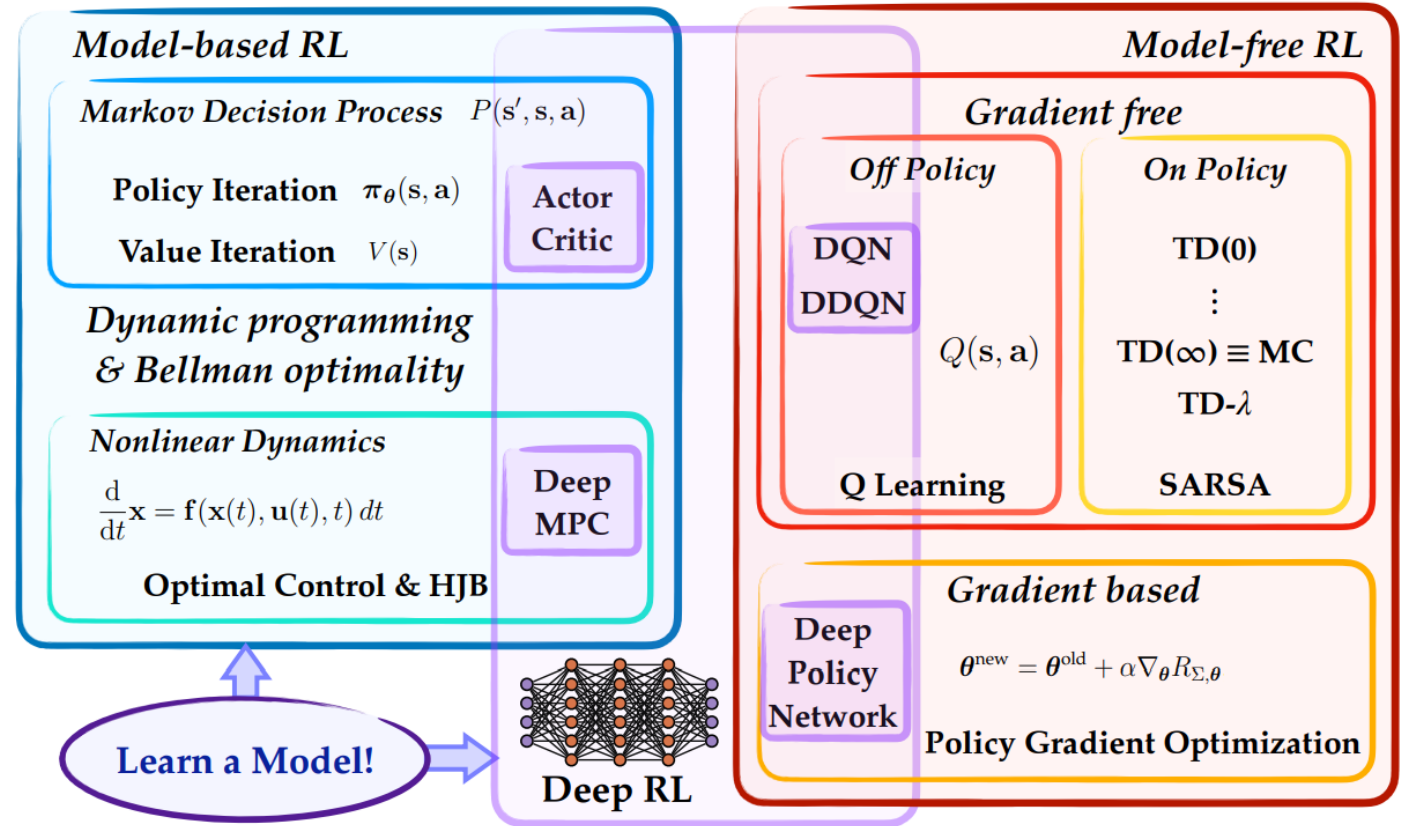
Comentarios de programación dinámica

- La versión estándar necesita actualizar el ancho del árbol
 - Cada estado sucesor y acción es considerada
 - Utiliza el modelo real y la función de recompensa
- DP es efectivo para problemas de tamaño mediado (algunos millones de estados)
- Para problemas grandes DP sufre la maldición de la dimensionalidad. El número de estados crece exponencialmente con el número de estados
- Puede ser que incluso una revisión sea muy costosa

¿Qué podemos hacer?

- Considerar nuestros de las transiciones y recompensas
- Ventajas
 - Es libre de modelo: no necesitamos el conocimiento explícito de MDP
 - Rompe la maldición de la dimensionalidad
 - El costo es constante, independiente de $|S|$

Lo que sigue



Tarea 1

- MDPs
 - MDP del gridworld
 - Estados: coordenadas cartesianas
 - Acciones: up, down, left, right
 - Recompensa: -1 por movimiento
 - $\gamma = 1$
 - MDP del juego de gato con rival aleatorio
- Algoritmos
 - Iteración de valor
 - Iteración de política
- Pruebas
 - Aplicar ambos algoritmos a los dos MDPs
 - Para cada algoritmo mostrar $v^*(s)$ y $\pi^*(a|s)$
- Fecha de entrega: 27/02/2023

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

Para la otra vez...

- Predicción libre de modelo

The End.