UNIVERSITAT DE BARCELONA

Aprendizaje

Inteligencia Artificial

Aprendizaje

- Procesos de Decisión de Márkov (MDP)
- Aprendizaje por Refuerzo (RL)

UNIVERSITAT DE BARCELONA

Bibliografía

- Capítulos 16 y 23 del libro de Russell and Norvig “Artificial Intelligence A Modern Approach”

- Sutton and Barto: “Reinforcement Learning: An introduction”
  - capítulos 3, 4 y 6 (secciones 6.1, 6.2, 6.5)

UNIVERSITAT DE BARCELONA

Procesos de Decisión de Márkov

Inteligencia Artificial

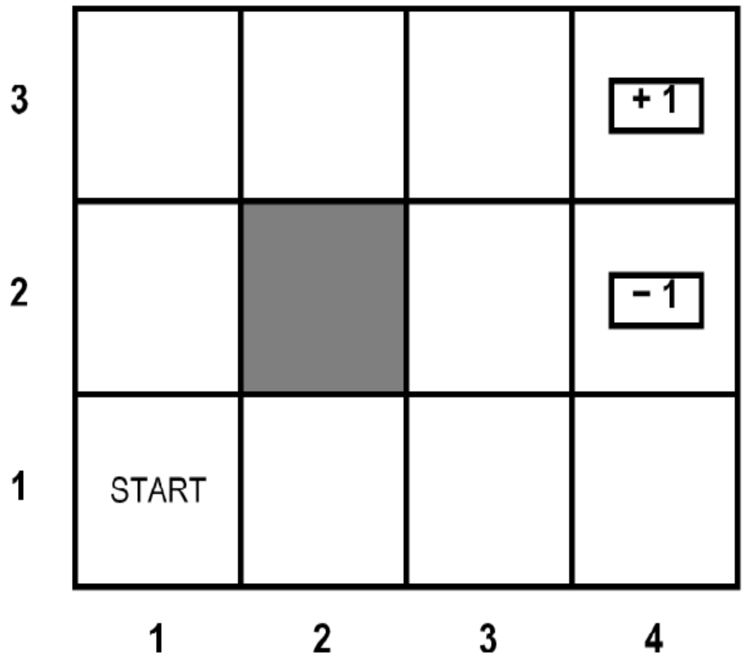
Procesos de decisión de Márkov

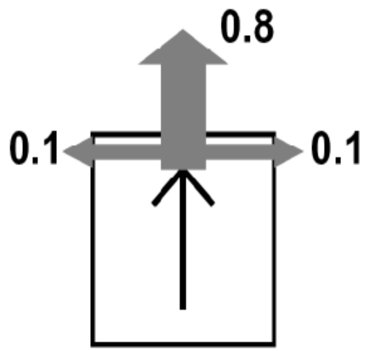
- Procesos de decisión de Márkov (MDP):
  - Definición
  - Ejemplo
- Algoritmos para resolver un MPD
  - Utilidad de una secuencia de acciones.
  - Iteración de valores

Ej. de probl. decisión secuencial

- El agente vive en una rejilla
- El bloque oscuro impide el paso
- Las acciones del agente no siempre tienen el mismo resultado. Si quiere ir hacia el norte:
  - 80% de las veces va al norte
  - 10% al este
  - 10% al oeste

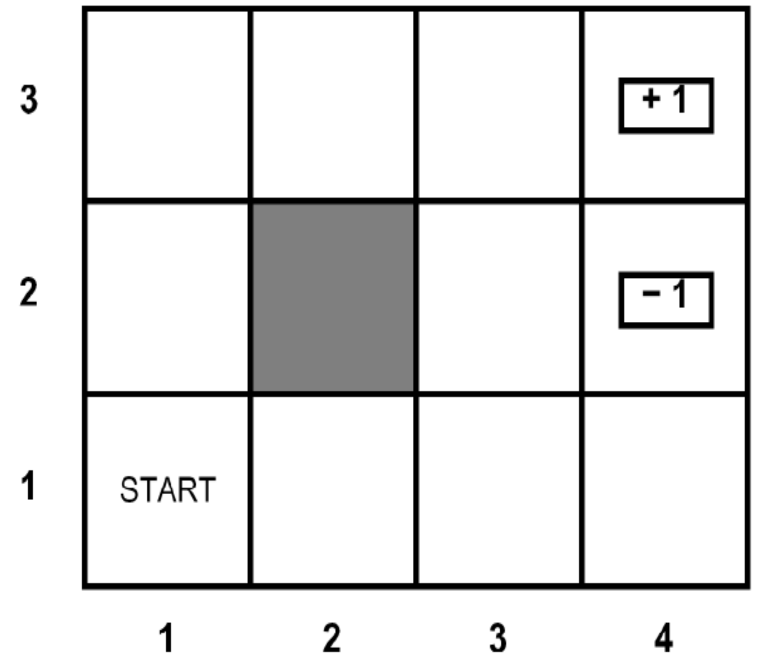
Problema no determinista



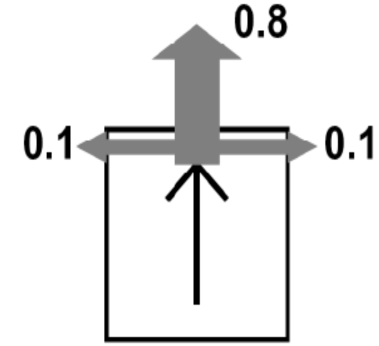


Ej. de probl. decisión secuencial

- El agente vive en una rejilla
- El bloque oscuro impide el paso
- Las acciones del agente no siempre tienen el mismo resultado. Si quiere ir hacia el norte:
  - 80% de las veces va al norte
  - 10% al este
  - 10% al oeste

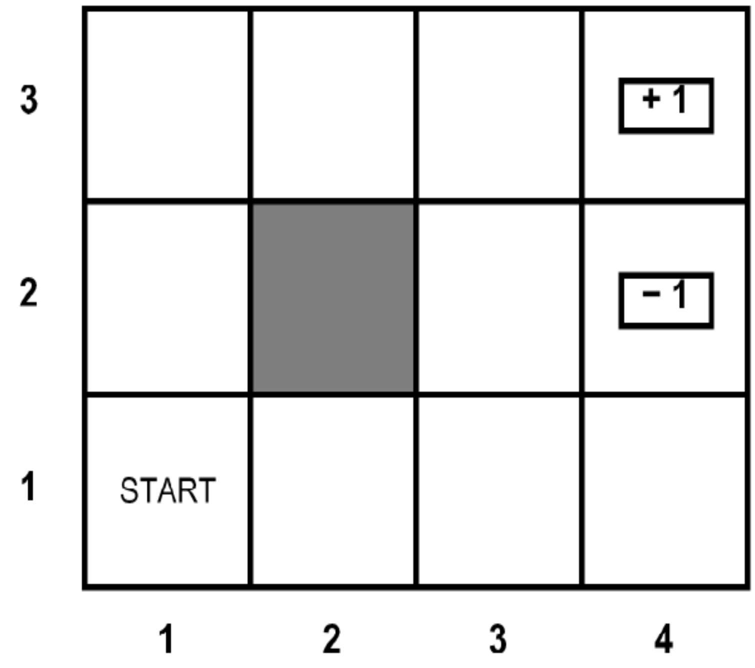


- Puede recibir una pequeña recompensa por ”sobrevivir”
- La mayor recompensa llega al final

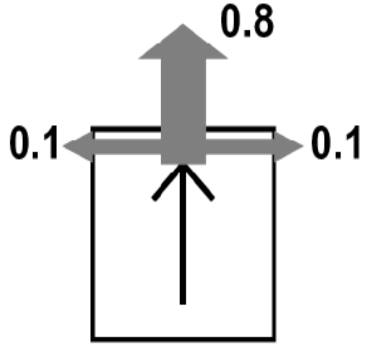


Ej.: problema decisión secuencial

- El agente vive en una rejilla
- El bloque oscuro impide el paso
- Las acciones del agente no siempre tienen el mismo resultado. Si quiere ir hacia el norte:
  - 80% de las veces va al norte
  - 10% al este
  - 10% al oeste



- Puede recibir una pequeña recompensa por ”sobrevivir”
- La mayor recompensa llega al final
- Objetivo: maximizar la suma de recompensas.



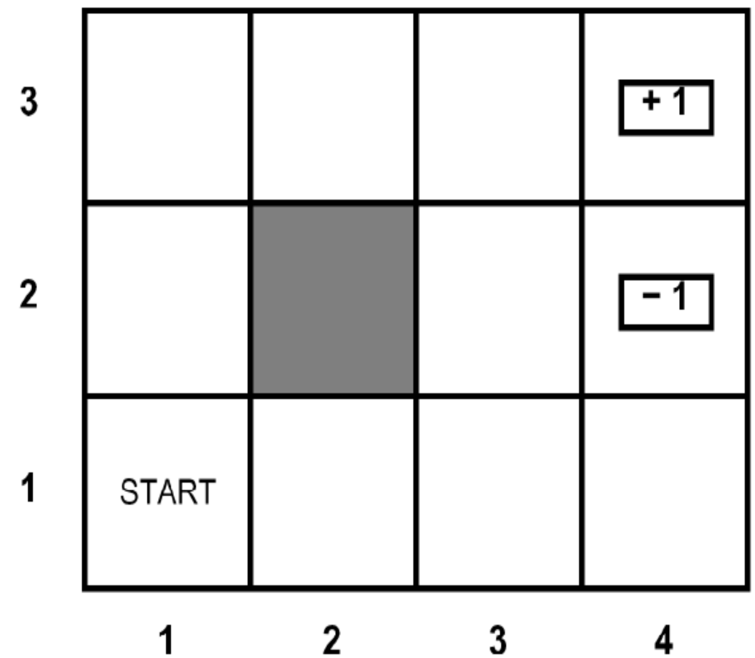
La utilidad del agente depende de una secuencia de decisiones

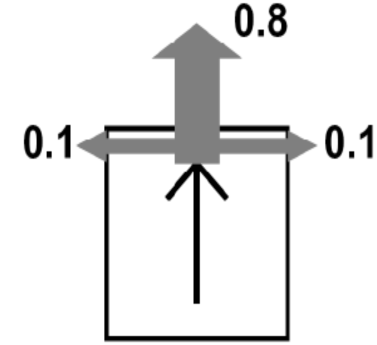
Procesos de decisión de Markov

Un MDP se define por:

- Un conjunto de estados S.
  - Un estado inicial
  - Posiblemente un estado terminal
- Un conjunto de acciones A
- Una función de transición

  $T: A \times S \times S \rightarrow R$
  - $T(a,s,s')$ = Probabilidad de ir al estado s' cuando estamos en el estado s y realizamos la acción a.
  - $T(a,s,s') = P(s'|s,a)$
- Función de recompensa $R(s,a,s')$
  - En ocasiones $R(s)$





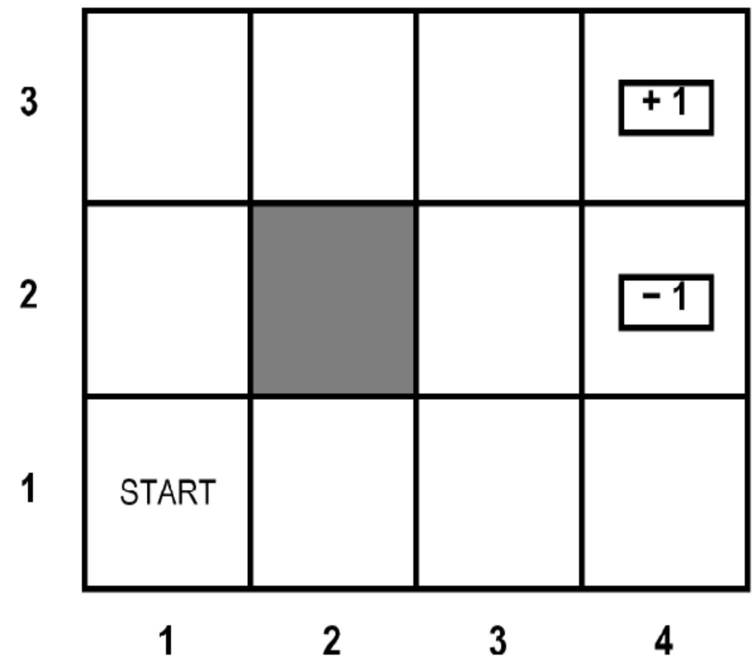
Procesos de decisión de Markov

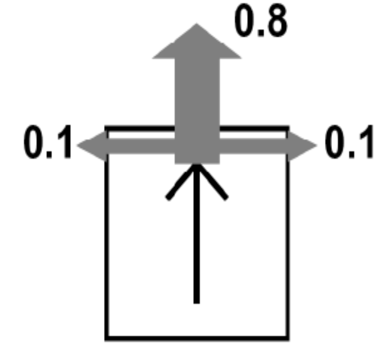
MDP del mundo rejilla?

- ¿Estados?:

  ¿inicial s_{ini}? ¿terminal s_{fin}?

- ¿Acciones?
- ¿Función de transición? T(s,a,s') = P(s'|s,a)





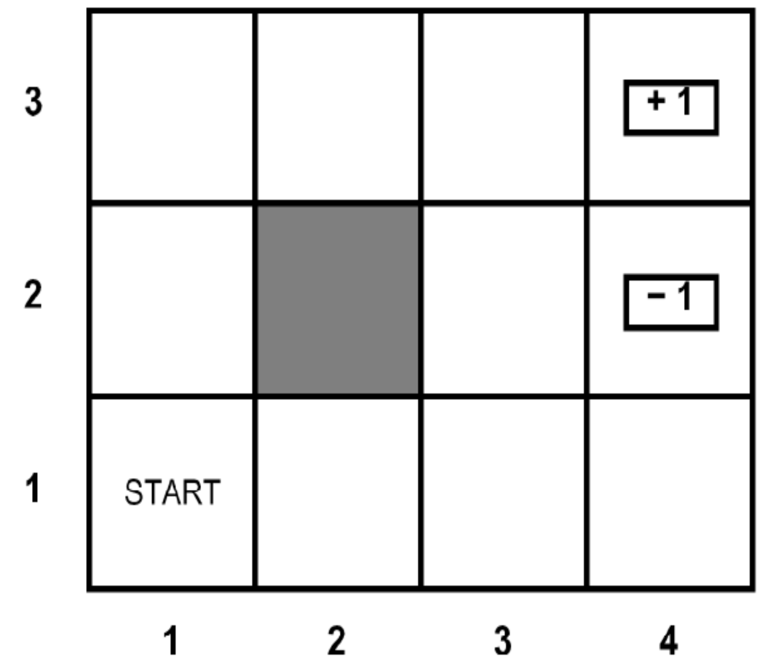
- ¿Función de recompensa? R(s,a,s'), R(s)

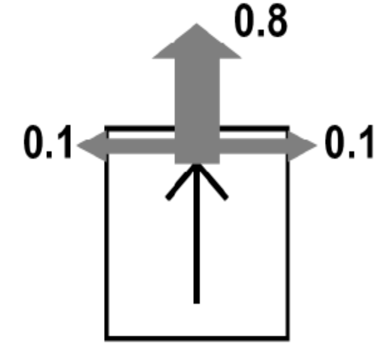
Procesos de decisión de Markov

MDP del mundo rejilla:

- Estados: S={(x,y)|x$\in${1,2,3,4}, y$\in${1,2,3}}

  inicial s_{ini}=(1,1) y terminales $s1_{fin}$=(4,3), $s2_{fin}$=(4,2)



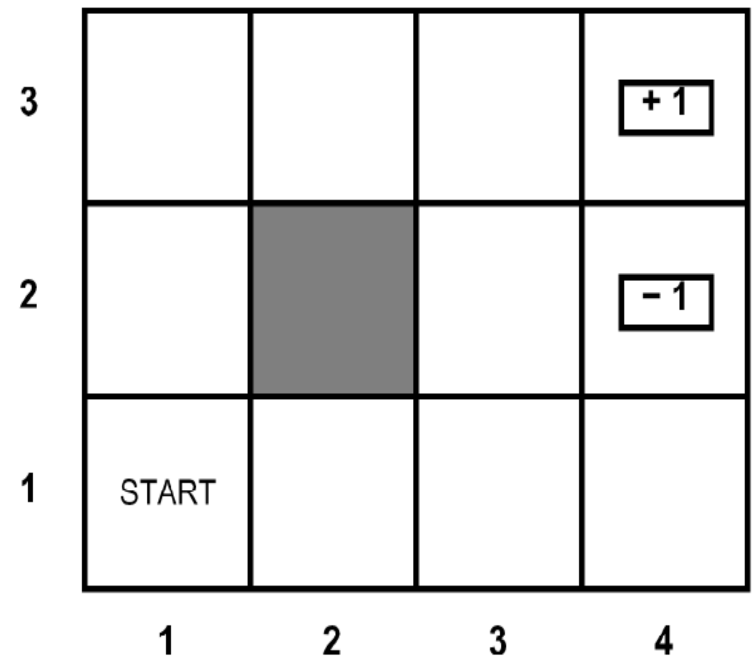


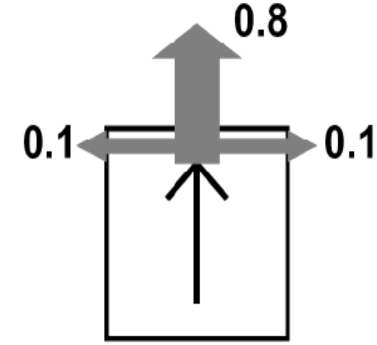
Procesos de decisión de Markov

MDP del mundo rejilla:

- Estados: $S=\{(x,y)|x\in\{1,2,3,4\}, y\in\{1,2,3\}\}$

  inicial $s_{ini}=(1,1)$ y terminales $s1_{fin}=(4,3)$, $s2_{fin}=(4,2)$
- Acciones A={up, down, right, left}.





Procesos de decisión de Markov

MDP del mundo rejilla:

- Estados: $S=\{(x,y)|x\in\{1,2,3,4\}, y\in\{1,2,3\}\}$

  inicial $s_{ini}=(1,1)$ y terminales $s1_{fin}=(4,3)$, $s2_{fin}=(4,2)$

- Acciones A={up, down, right, left}.
- Función de transición $T(s,a,s') = P(s'|s,a)$

$T((x,y),up,(x,y+1))=0.8$ si $y<3$ && $x \neq 2,y\neq 1$

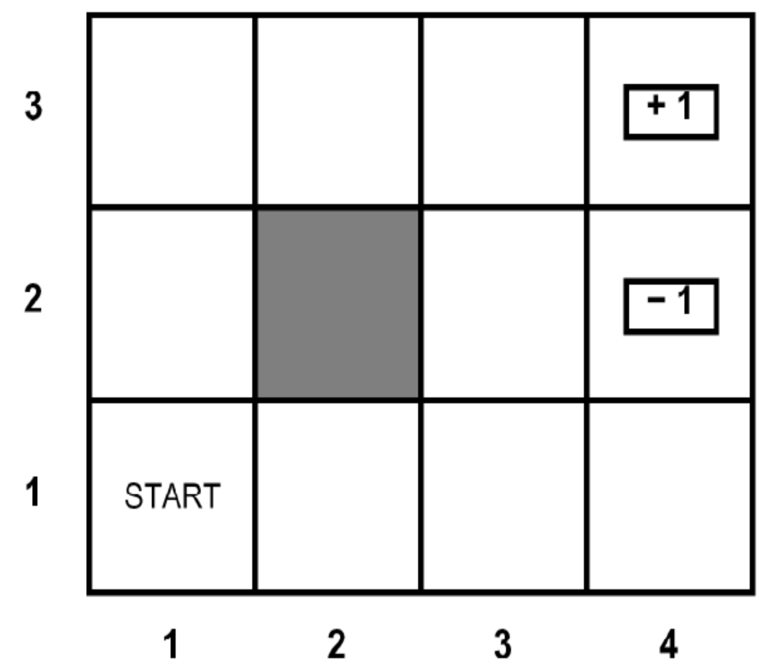
$T((x,y),up,(x-1,y))=0.1$ si $x>1$, && $x \neq 3,y\neq 2$

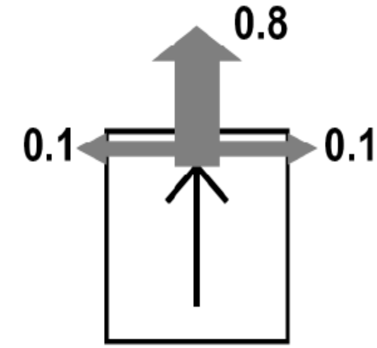
$T((x,y),up,(x+1,y))=0.1$ si $x<4$ && $x \neq 1,y\neq 2$

$T((x,y),right,(x+1,y))=0.8$ si $x<4$ && $x \neq 1,y\neq 2$

$T((x,y),right,(x,y+1))=0.1$ si $y<3$ && $x \neq 2,y\neq 1$

….





Procesos de decisión de Markov

MDP del mundo rejilla:

- Estados: $S=\{(x,y)|x\in\{1,2,3,4\}, y\in\{1,2,3\}\}$

  inicial $s_{ini}=(1,1)$ y terminales $s1_{fin}=(4,3)$, $s2_{fin}=(4,2)$
- Acciones A={up, down, right, left}.
- Función de transición T(s,a,s') = P(s'|s,a)

T((x,y),up,(x,y+1))=0.8 si y<3 && x ≠2,y≠1

T((x,y),up,(x-1,y))=0.1 si x>1, && x ≠3,y≠2

T((x,y),up,(x+1,y))=0.1 si x<4 && x ≠1,y≠2

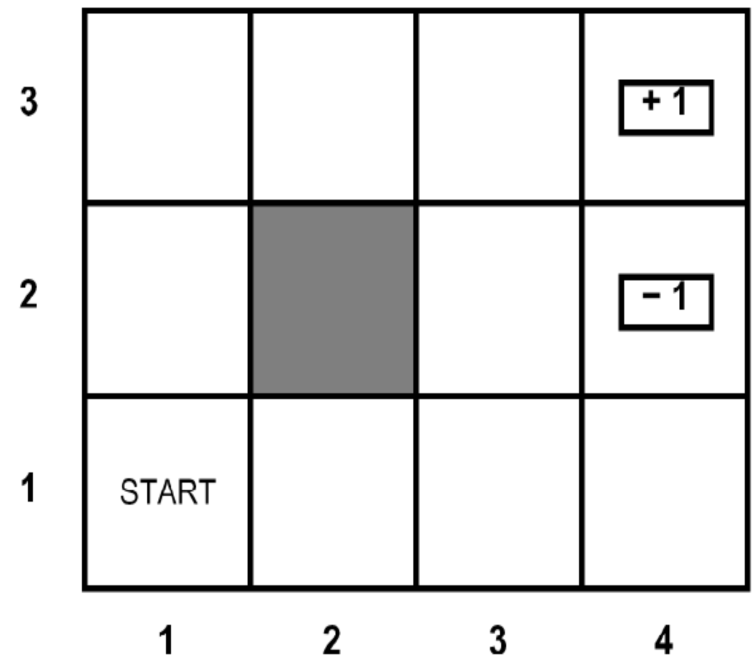
T((x,y),right,(x+1,y))=0.8 si x<4 && x ≠1,y≠2

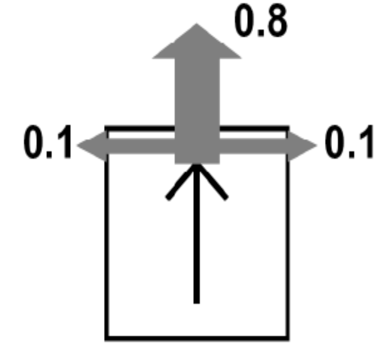
T((x,y),right,(x,y+1))=0.1 si y<3 && x ≠2,y≠1

….

- Función de recompensa R(s,a,s'), R(s)

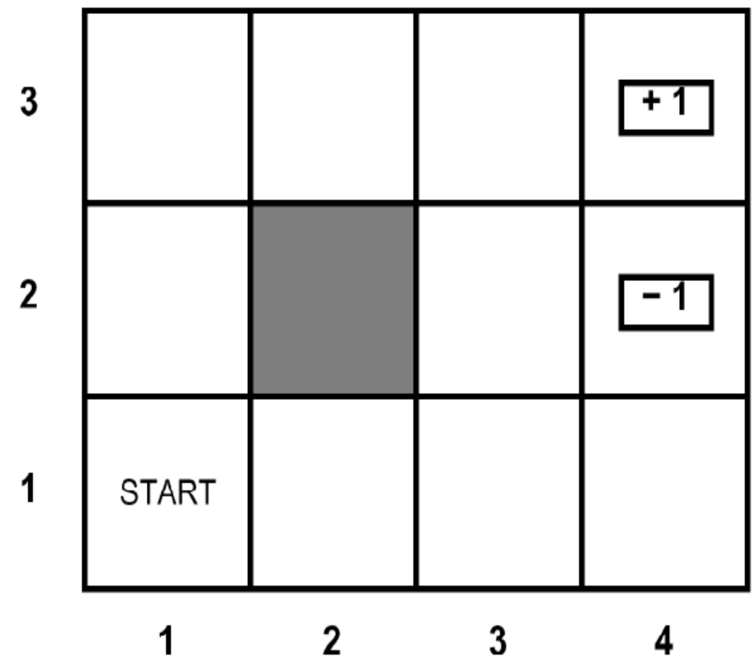
  $R(s1_{fin})=1$, $R(s2_{fin})=-1$, $R(s)=-0.04$, $s \neq s1_{fin}, s2_{fin}$

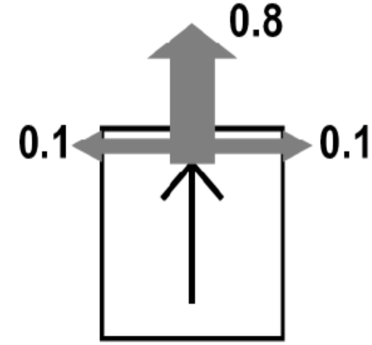




Procesos de decisión de Markov

- Los procesos de decisión de Markov son una familia de problemas de búsqueda no deterministas.
- El aprendizaje por refuerzo es un PDM del que desconocemos la función de recompensa o la función de transición.





Procesos de decisión de ¿Markov?

- Andréi Márkov (1856-1922)
- La hipótesis de Márkov:
  - El estado actual resume toda la información relevante del pasado.

Procesos de decisión de ¿Markov?

- Andréi Márkov (1856-1922)
- La hipótesis de Márkov:
  - El estado actual resume toda la información relevante del pasado.
- En el caso de los procesos de decisión de Markov, esto quiere decir que:

$$P(S_{t+1} | S_t, a_t, S_{t-1}, a_{t-1}, \ldots, S_0, a_0) = P(S_{t+1} | S_t, a_t)$$

¿Qué es resolver un PDM?

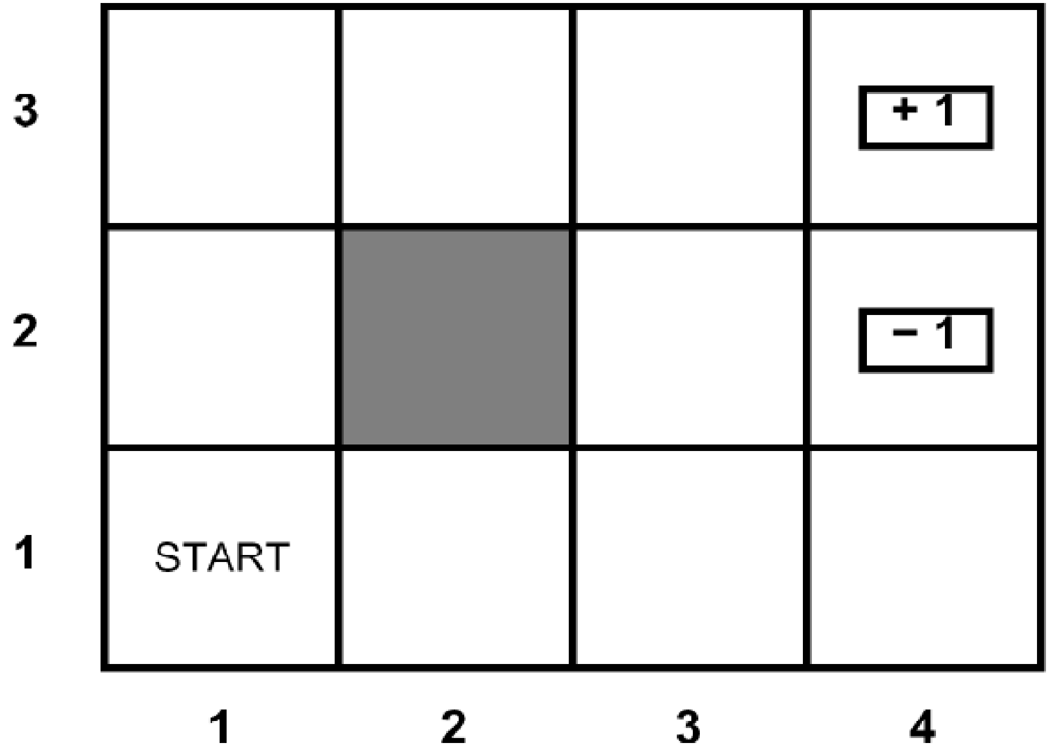
- En problemas de búsqueda deterministas en que interviene un único agente, queremos un plan óptimo, una secuencia de acciones desde el inicio hasta un objetivo.
- En un PDM queremos una política óptima $\pi^*: S \rightarrow A$

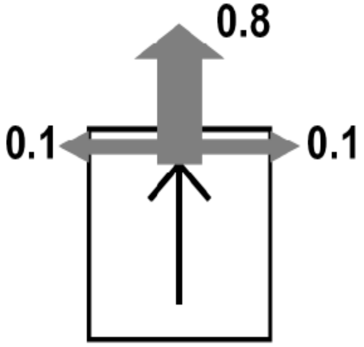
¿Qué es resolver un PDM?

- En problemas de búsqueda deterministas en que interviene un único agente, queremos un plan óptimo, una secuencia de acciones desde el inicio hasta un objetivo.
- En un PDM queremos una política óptima $\pi^*:S \rightarrow A$
  - Una política asigna una acción a cada estado $\pi(s)=a$.
  - Una política óptima maximiza la utilidad esperada si se sigue.
  - El resultado es un agente reflejo.

Mundo rejilla

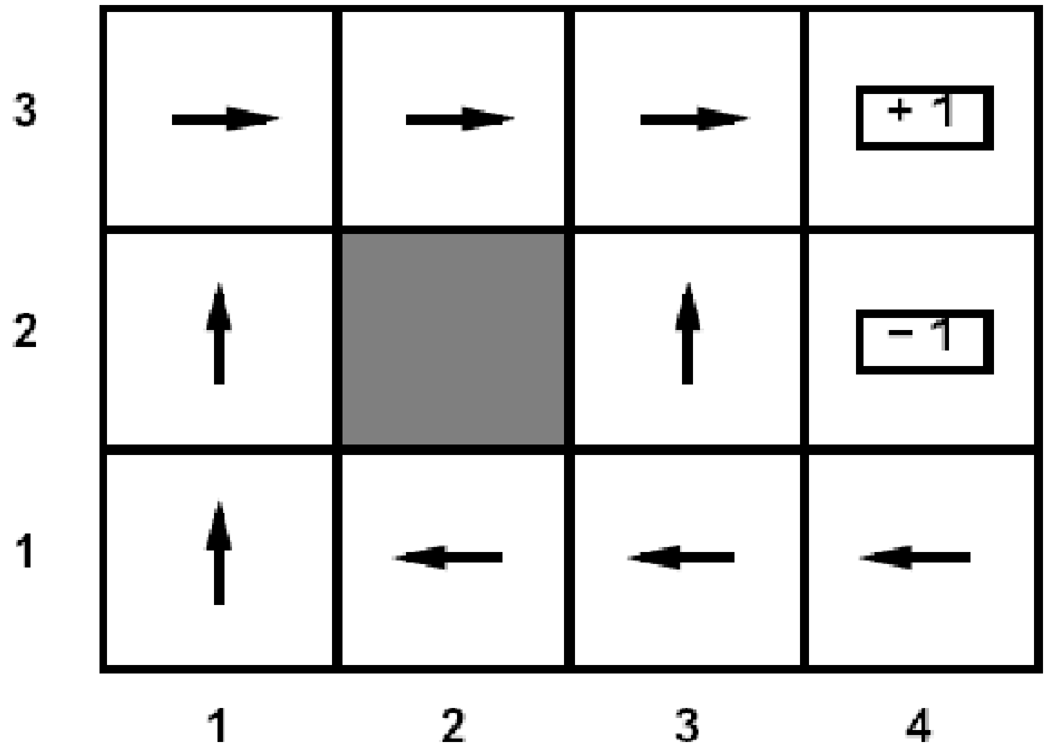
- ¿Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales?

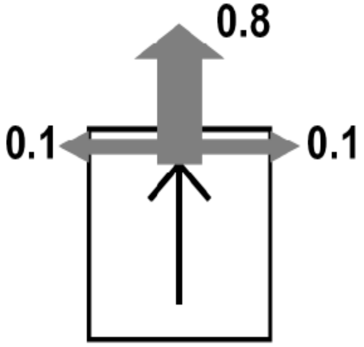




Mundo rejilla

- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales

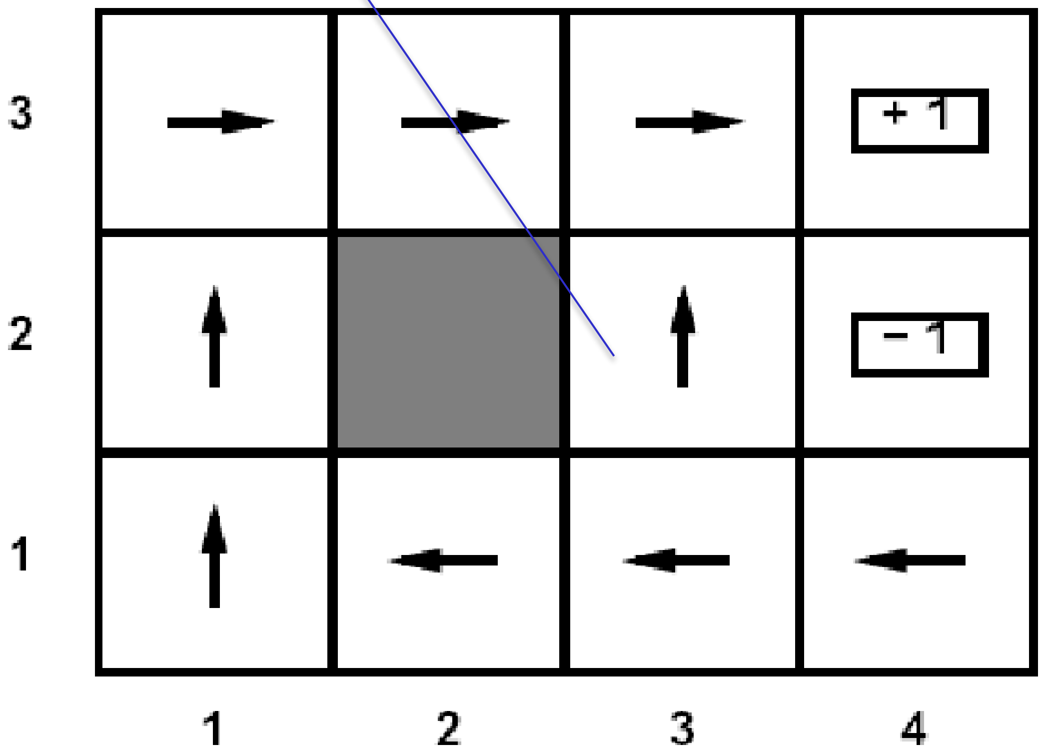


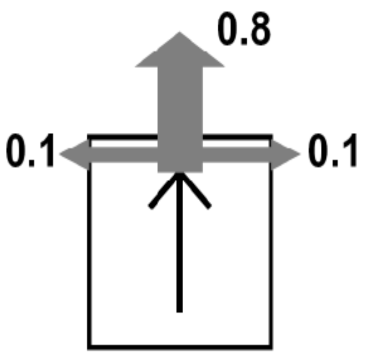


Mundo rejilla

- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales

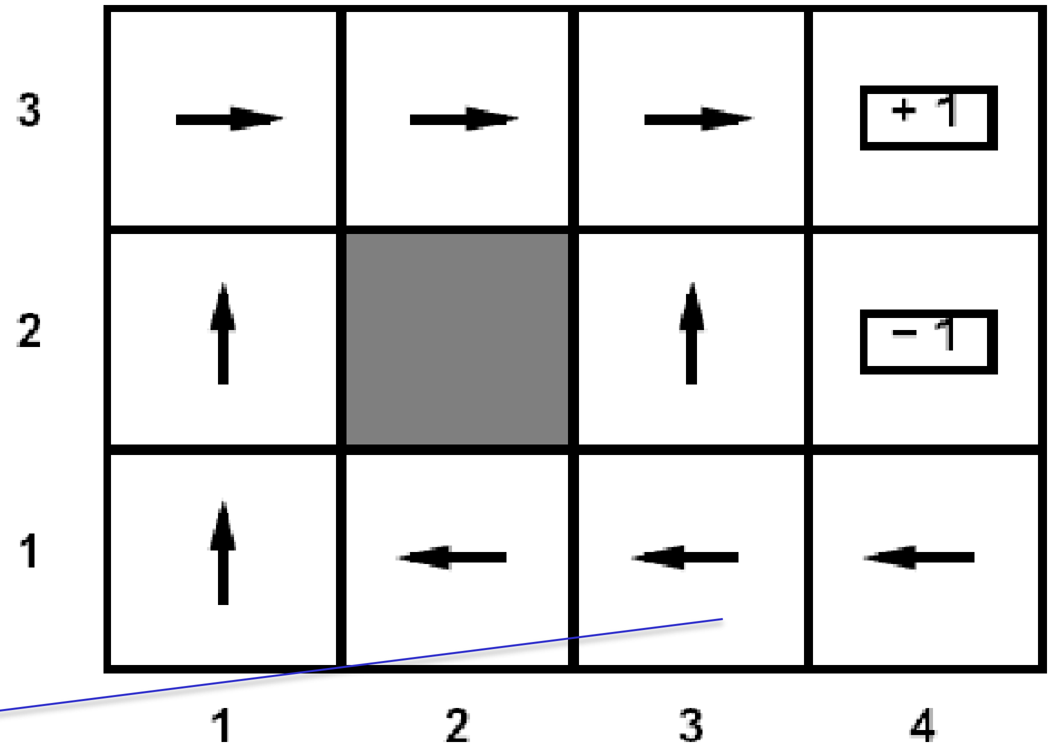
En el mejor de los casos -0.04+1=0,96
Tenemos posibilidad de que sea -1

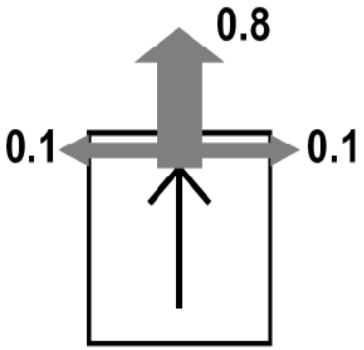




Mundo rejilla

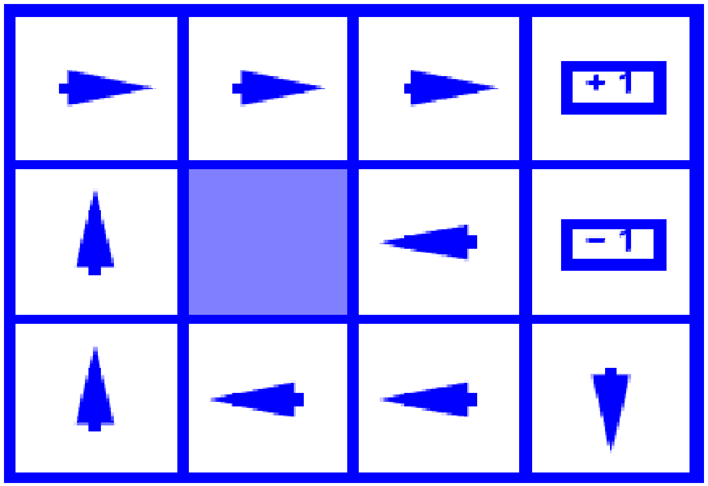
- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales





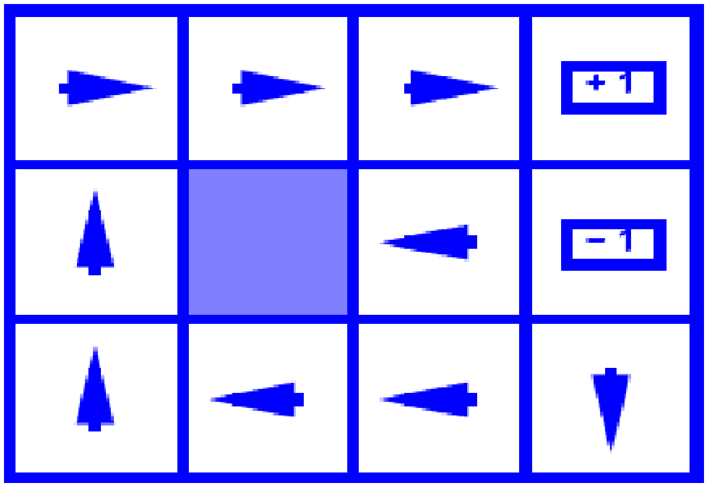
En el mejor de los casos -0.04*6+1=0,76
Si fuéramos hacia el N seria -0.04*2 + 1= 0.92 pero nos arriesgamos a tener -0.04-1= -1.04

¿Qué pasa con otros valores de R(s)? (para estados no terminales)

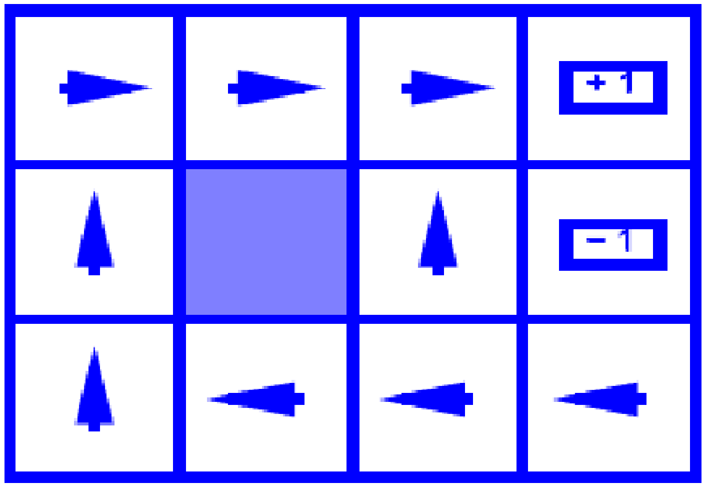


R(s) = -0.01

¿Qué pasa con otros valores de R(s)? (para estados no terminales)

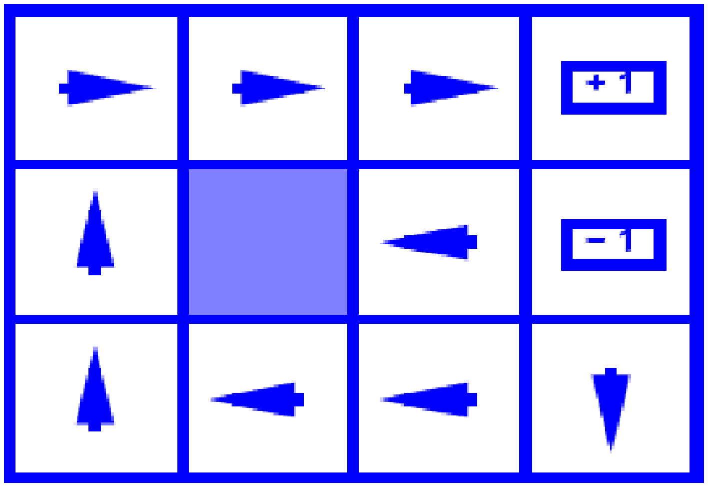


R(s) = -0.01

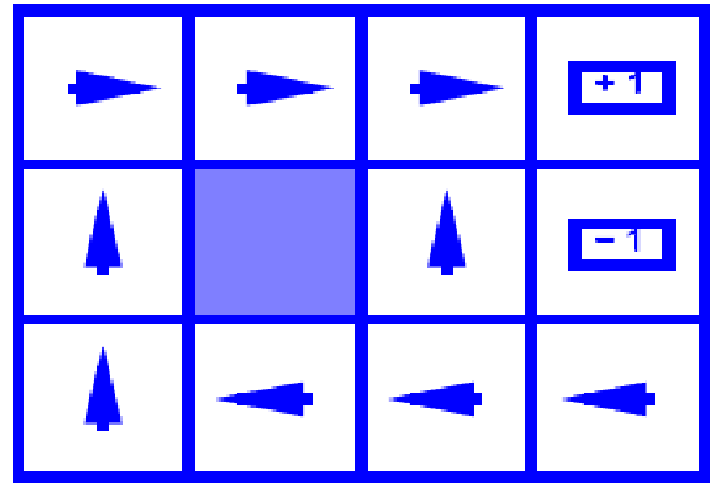


R(s) = -0.03

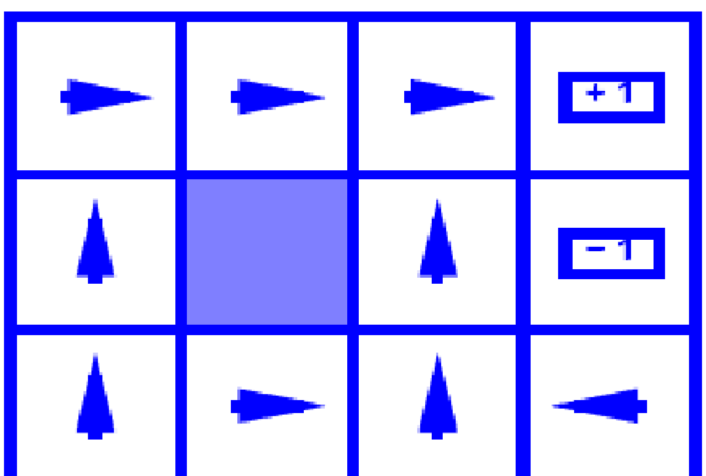
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



R(s) = -0.01

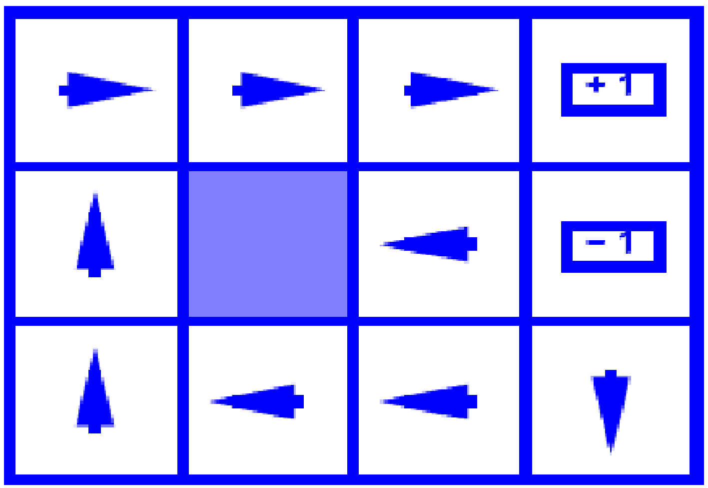


R(s) = -0.03

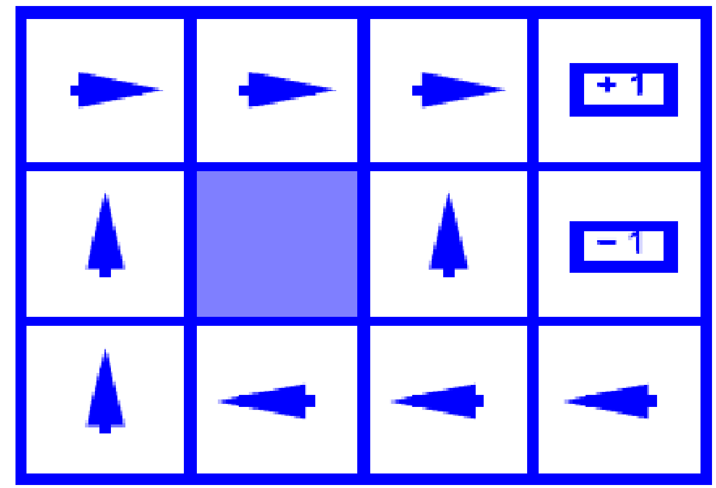


R(s) = -0.4

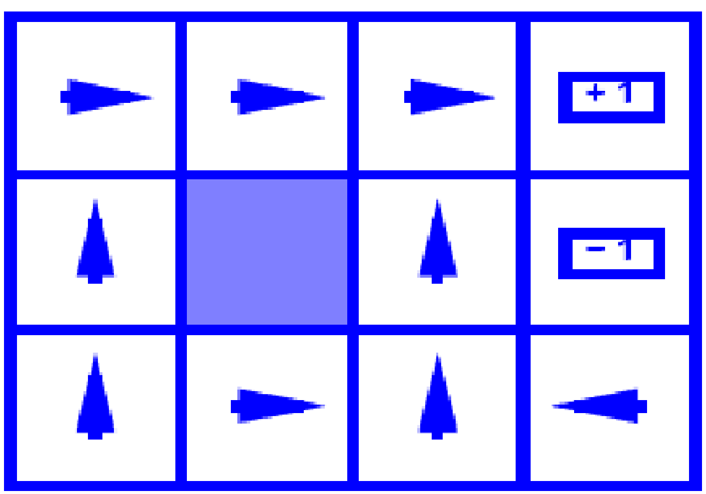
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



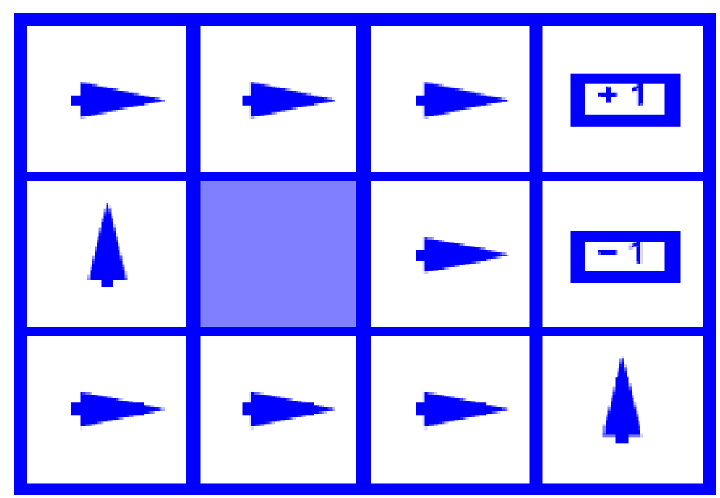
R(s) = -0.01



R(s) = -0.03

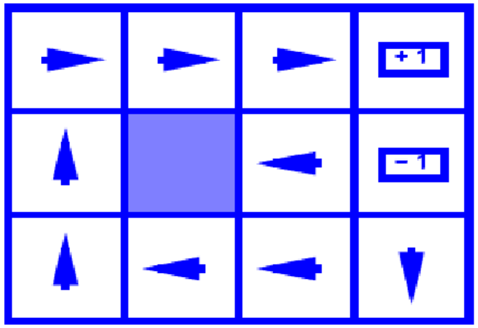


R(s) = -0.4

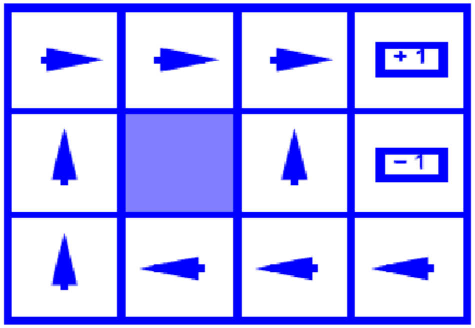


R(s) = -2.0

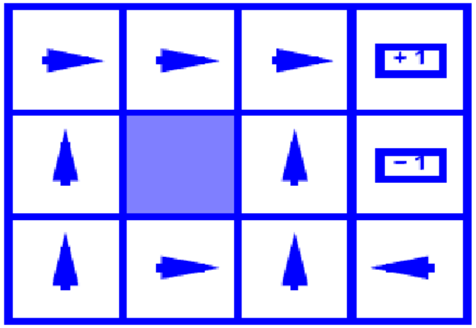
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



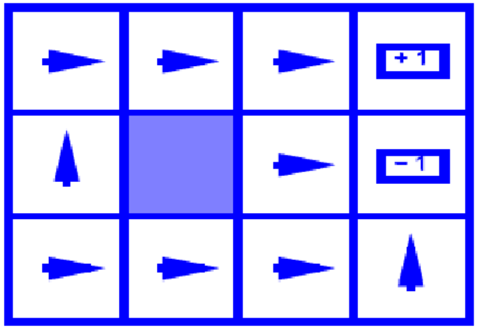
R(s) = -0.01



R(s) = -0.03



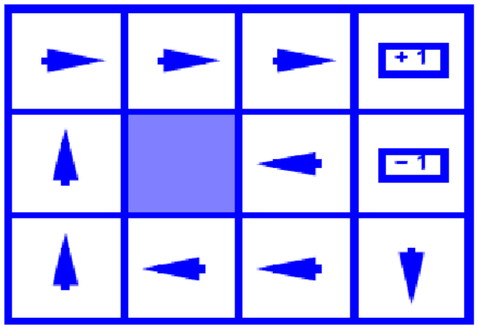
R(s) = -0.4



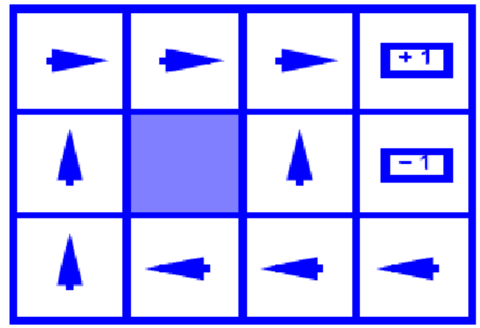
R(s) = -2.0

¿Y si R(s)>0 ?
(para estados no terminales)

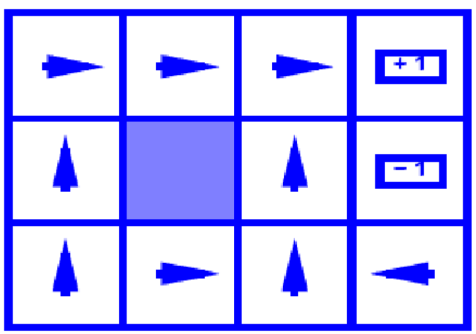
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



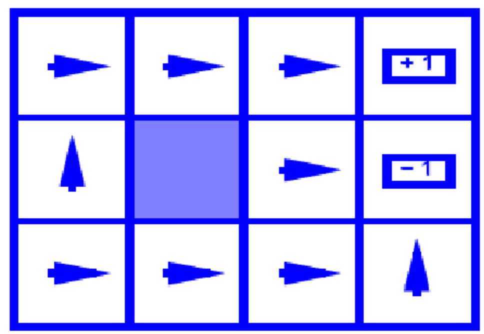
R(s) = -0.01



R(s) = -0.03

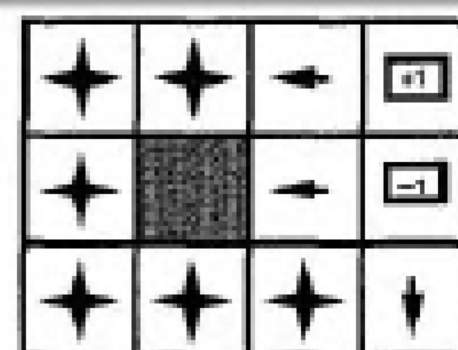


R(s) = -0.4



R(s) = -2.0

¿Y si R(s)>0 ?

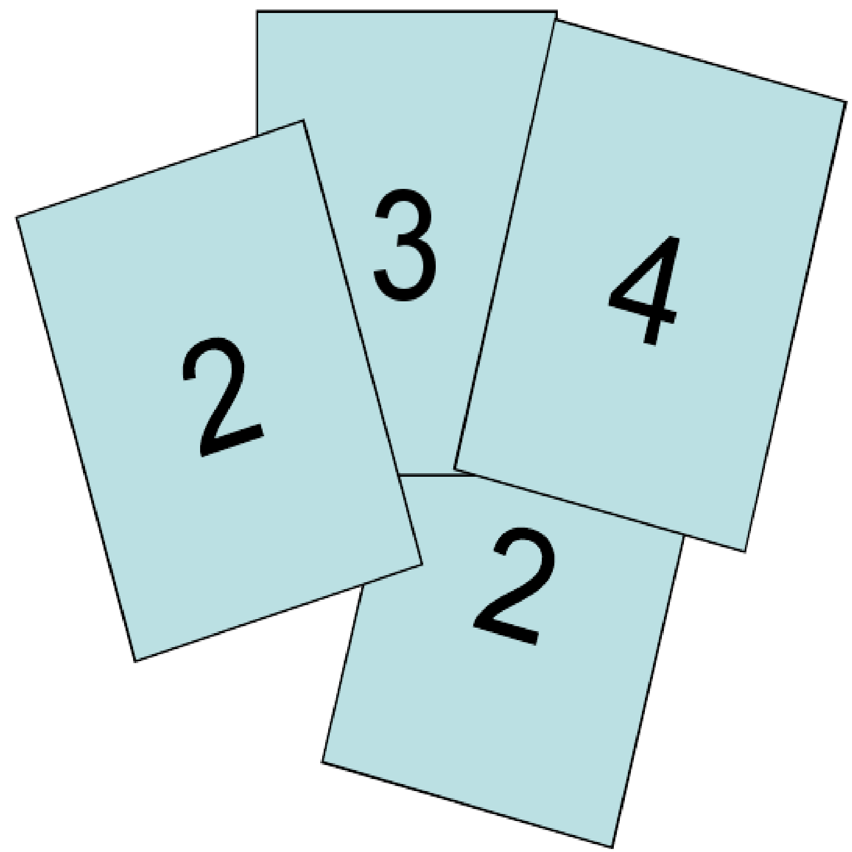


$R(s) > 0$

Superior/Inferior

- Tres tipos de cartas: 2,3,4
- Mazo infinito
- Comenzamos con un 3
- Después de cada carta tenemos que decir:

  Superior (≥) o Inferior (⋜)
- Se saca una nueva carta:
  - Si hemos acertado: ganamos tantos puntos como el valor de la carta.
  - Si no (hemos fallado): el juego termina.



Superior/Inferior

- ¿Podemos usar expectimax?
- No, porque:
  - En expectimax las recompensas llegan únicamente al final del juego
  - En expectimax el juego termina en algún momento.

- Formalicemos Superior/Inferior como un PDM...

Superior/Inferior como PDM

- Estados: 2,3,4,fin
  - Estado inicial: 3
- Acciones: Superior, Inferior

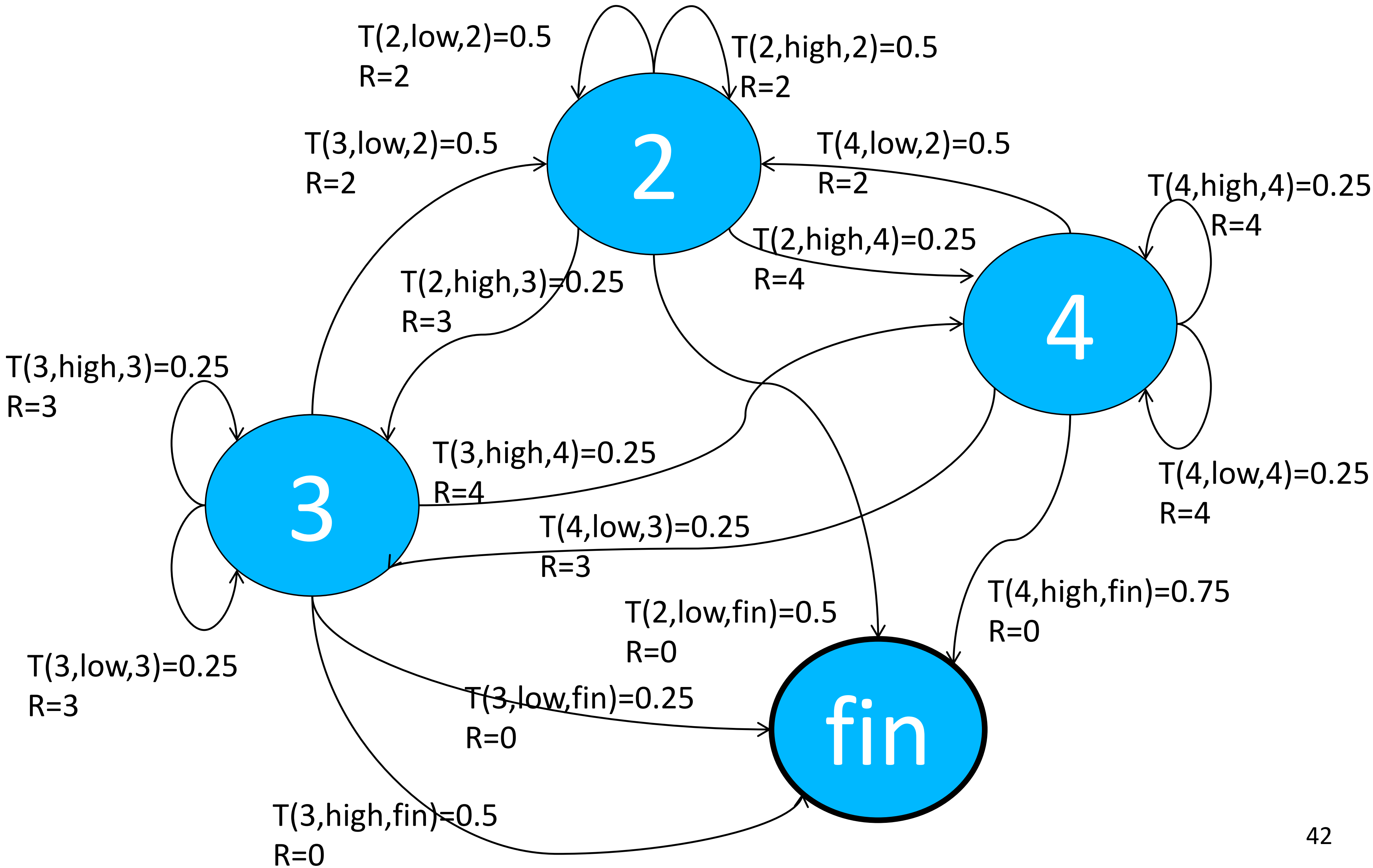
Superior/Inferior como PDM

- Estados: 2,3,4,fin
  - Estado inicial: 3
- Acciones: Superior, Inferior
- Modelo: T(s,a,s')
  - T(s=4,a=Inferior,s'=4)=0.25
  - T(s=4,a=Inferior,s'=3)=0.25
  - T(s=4,a=Inferior,s'=2)=0.5
  - T(s=4,a=Inferior,s'=fin)=0
  - ...
  - T(s=4,a=Superior,s'=4)=0.25
  - T(s=4,a=Superior,s'=3)=0
  - T(s=4,a=Superior,s'=2)=0
  - T(s=4,a=Superior,s'=fin)=0.75

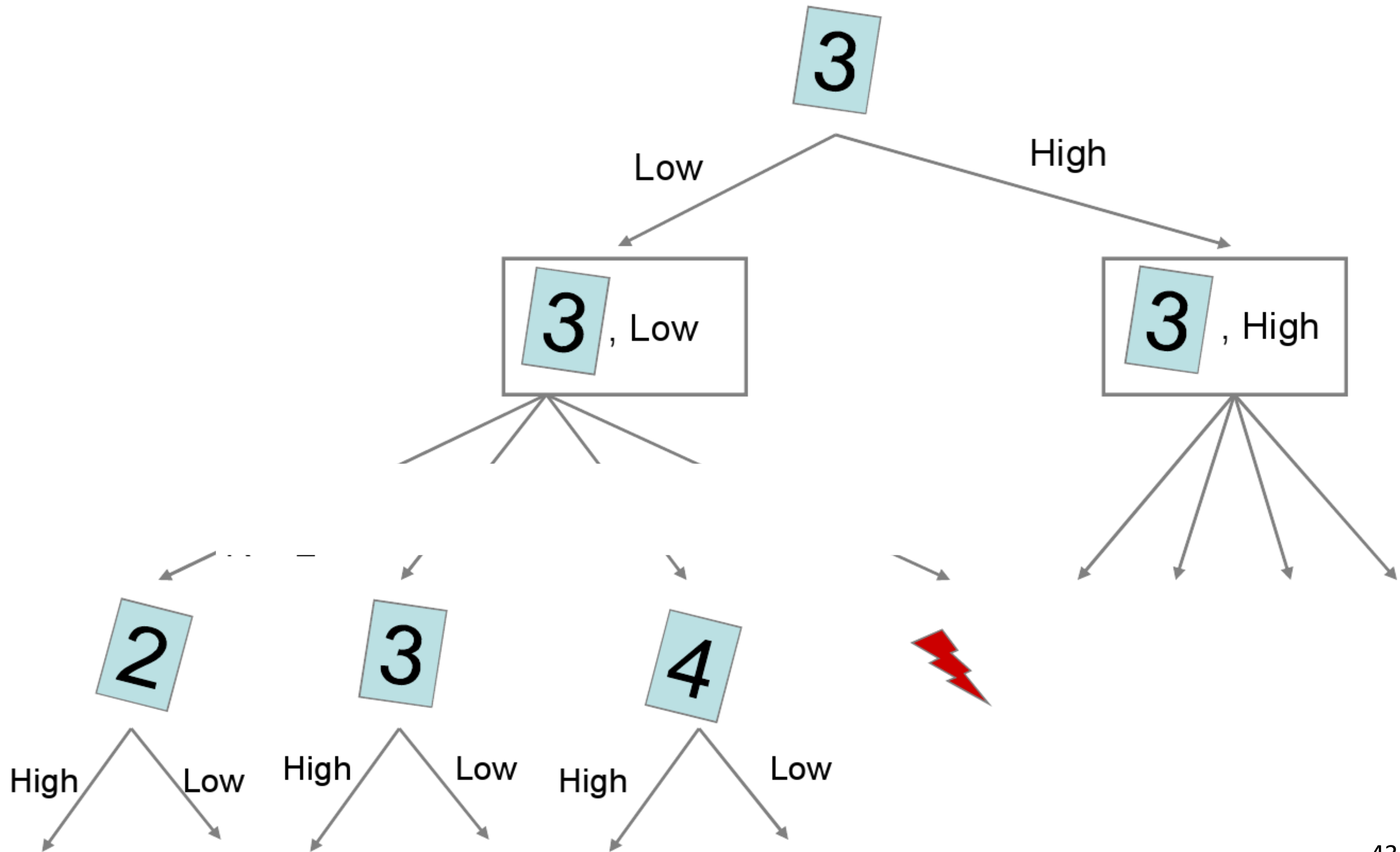
Superior/Inferior como PDM

- Estados: 2,3,4,fin
  - Estado inicial: 3
- Acciones: Superior, Inferior
- Modelo: T(s,a,s')
  - T(s=4,a=Inferior,s'=4)=0.25
  - T(s=4,a=Inferior,s'=3)=0.25
  - T(s=4,a=Inferior,s'=2)=0.5
  - T(s=4,a=Inferior,s'=fin)=0
  - ...
  - T(s=4,a=Superior,s'=4)=0.25
  - T(s=4,a=Superior,s'=3)=0
  - T(s=4,a=Superior,s'=2)=0
  - T(s=4,a=Superior,s'=fin)=0.75
- Retorno: R(s,a,s') = valor(s')   si s'≠fin,
  = 0   si s'=fin

Espacio estados Superior/Inferior

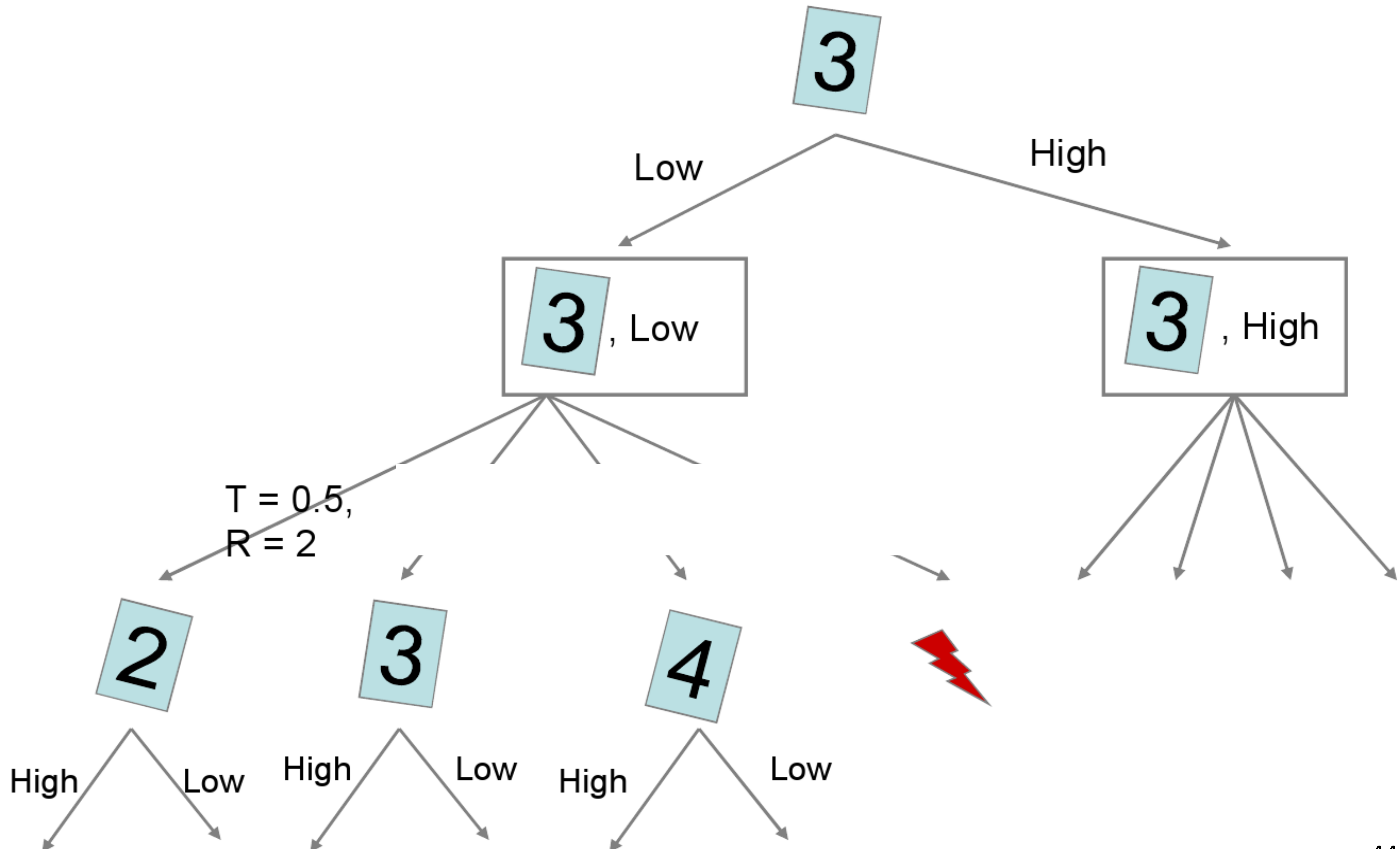


Superior/Inferior

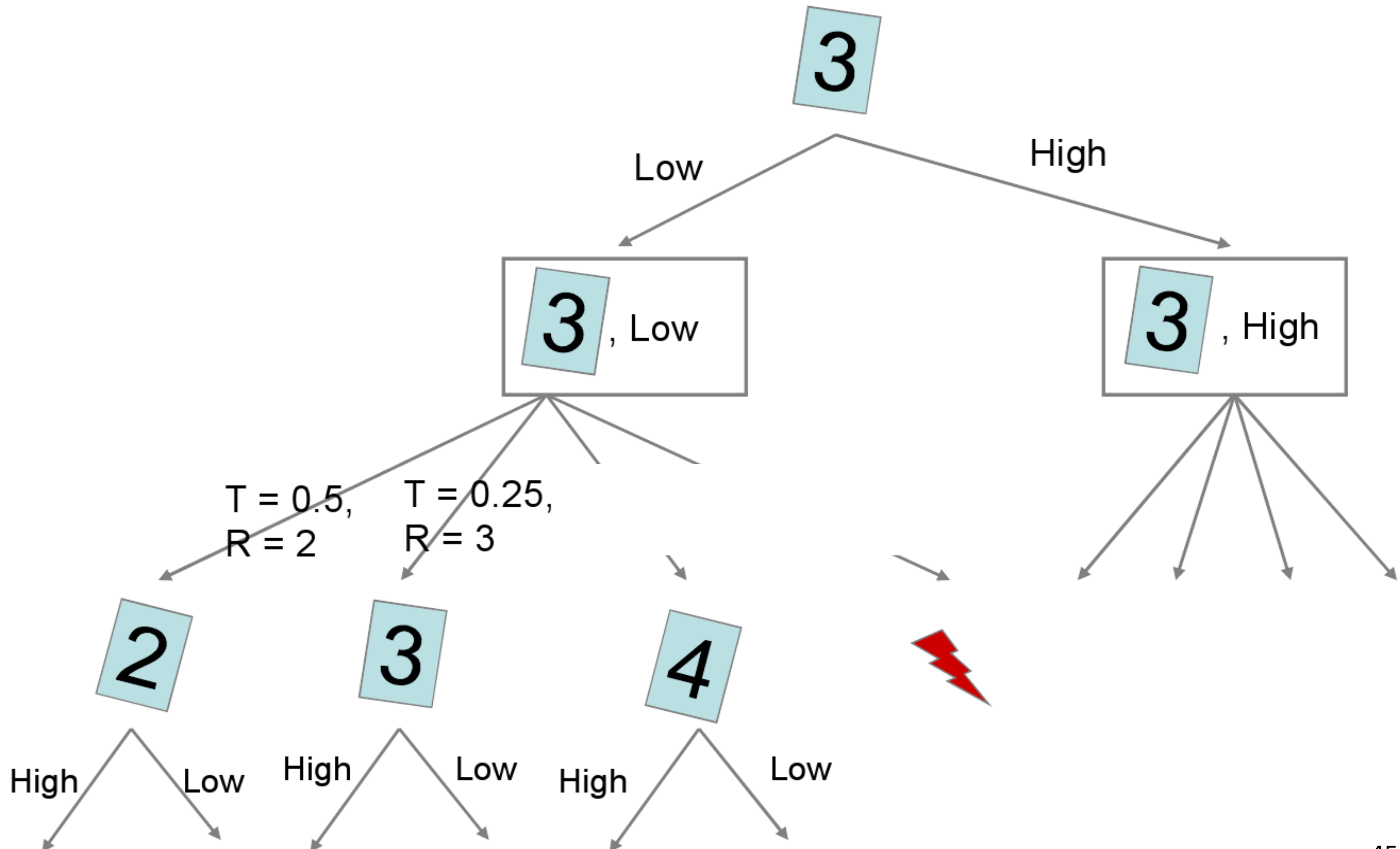


UNIVERSITAT DE BARCELONA

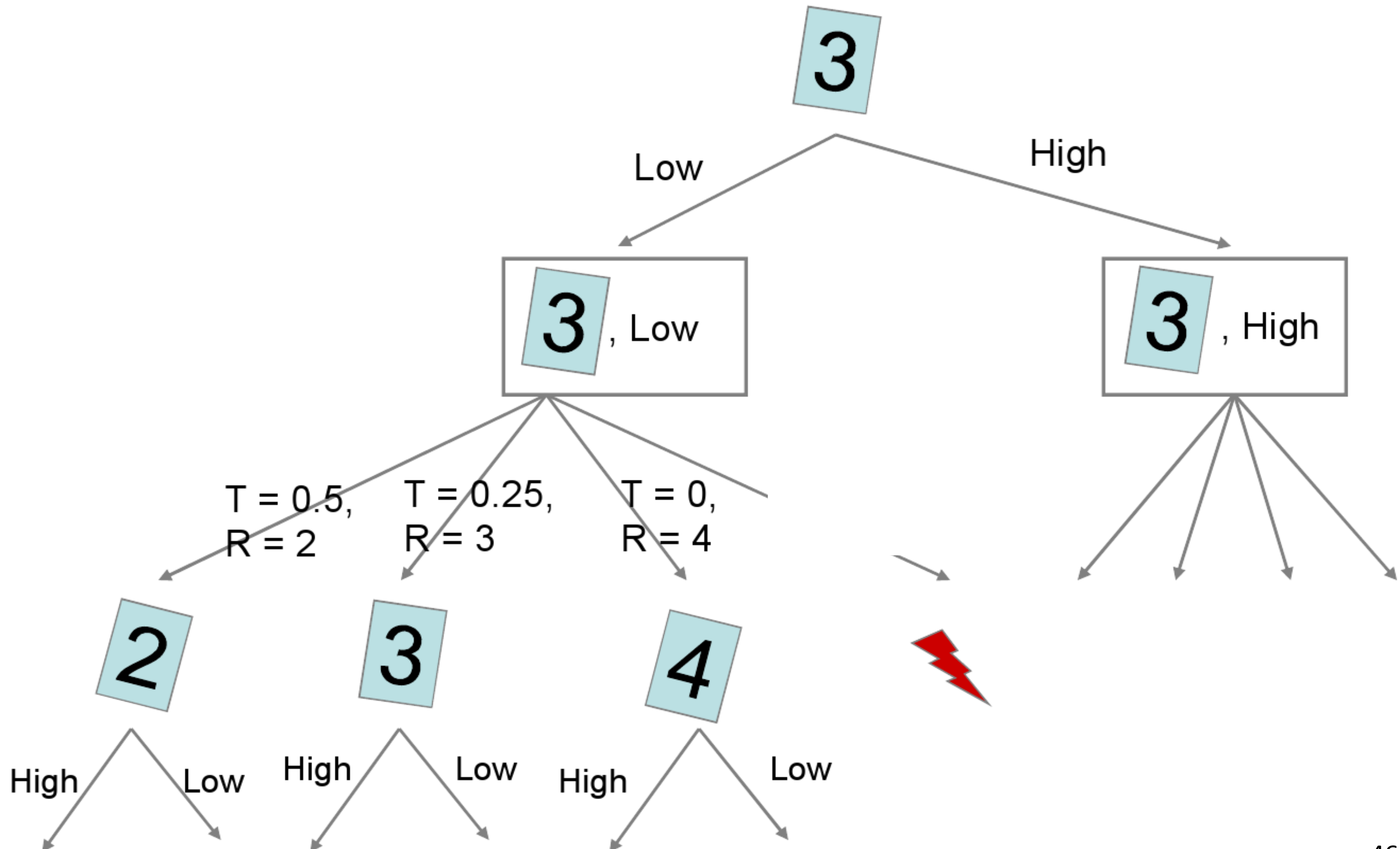
Superior/Inferior



Superior/Inferior

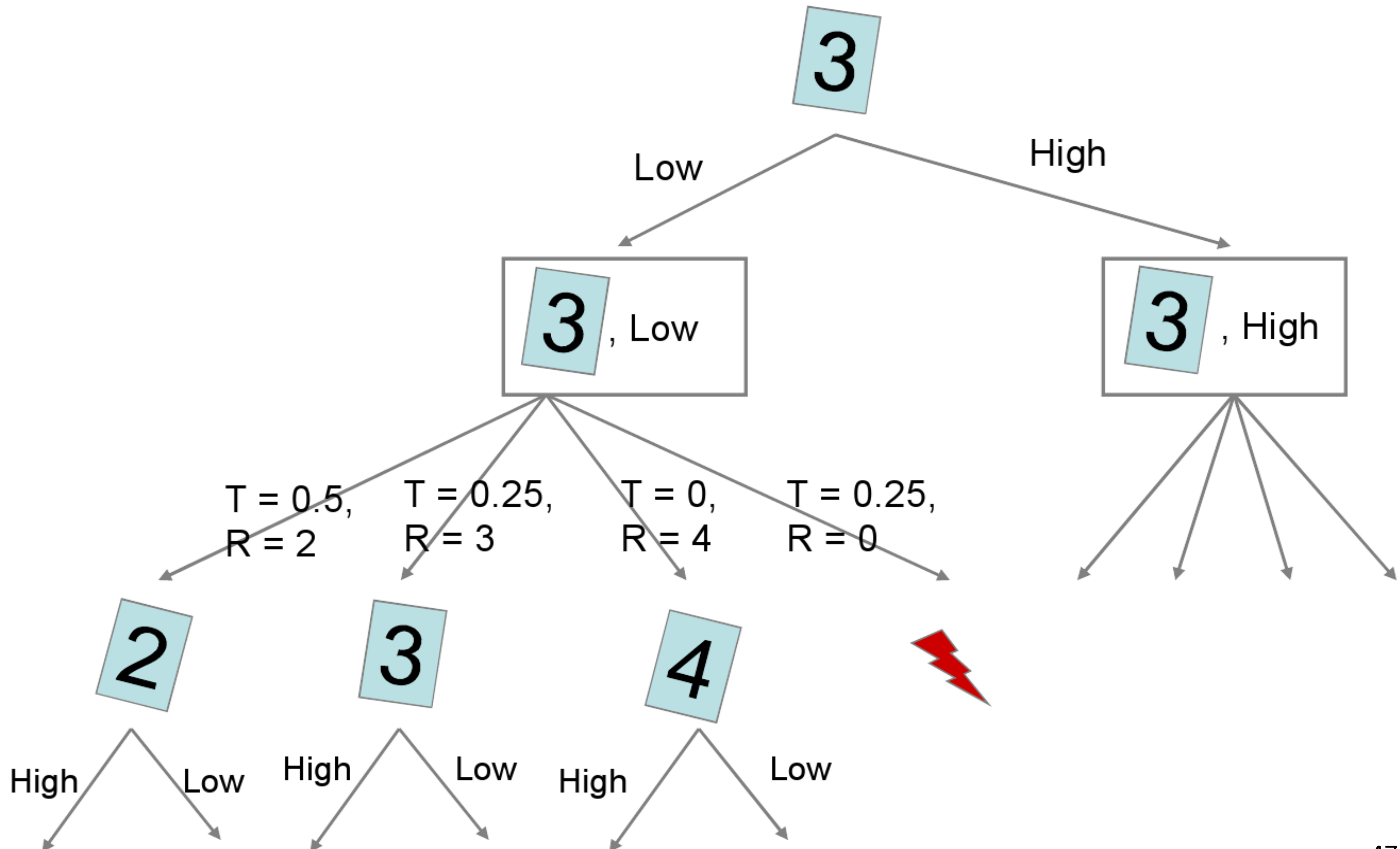


Superior/Inferior



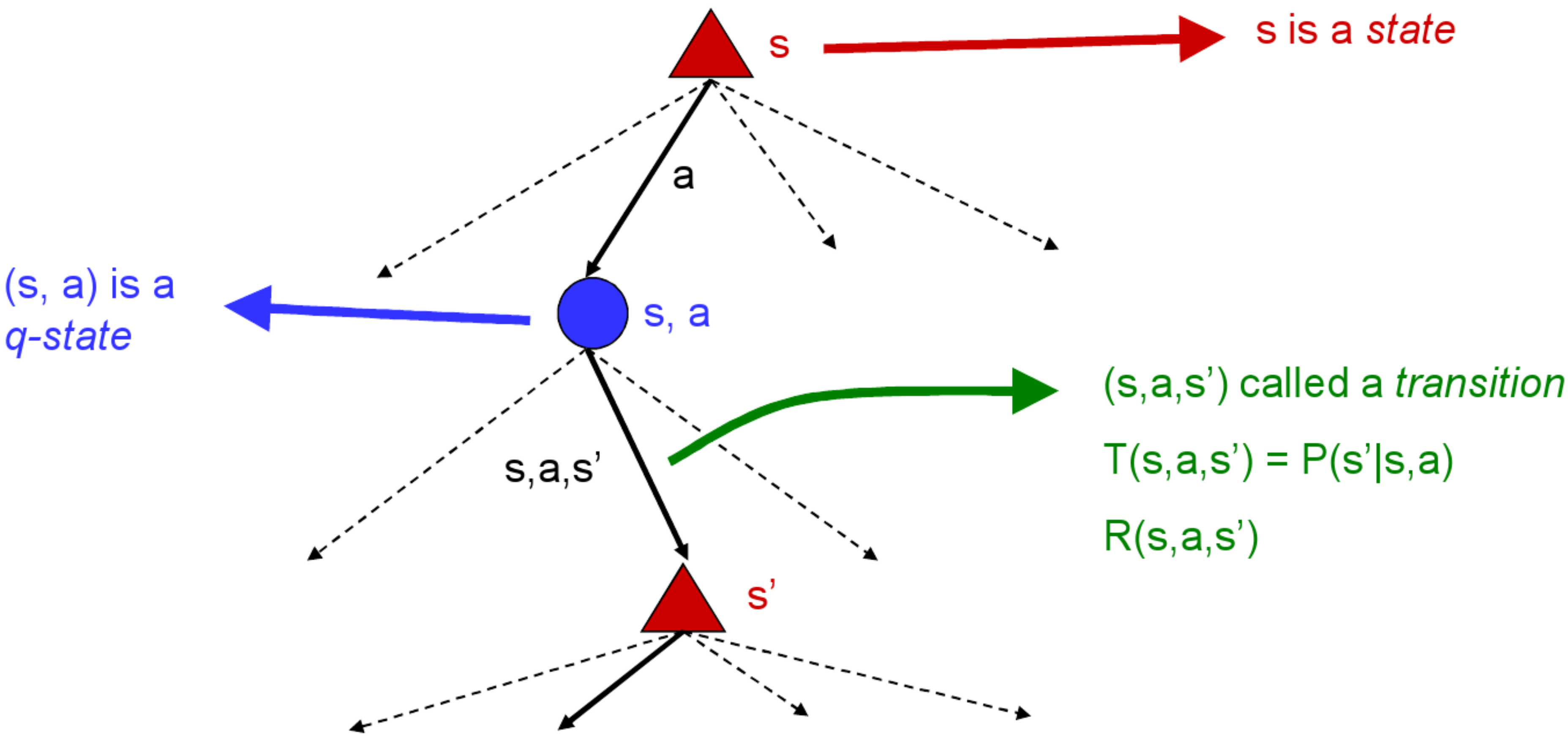
UNIVERSITAT DE BARCELONA

Superior/Inferior



Árboles de búsqueda PDM

- Cada estado del PDM genera un árbol de búsqueda expectimax.



Nos interesa la utilidad de hacer una acción a en un estado s

Utilidad de una secuencia de acciones

- Para formalizar la optimalidad de una política, necesitamos entender la optimalidad de una secuencia de acciones.
- Típicamente consideramos preferencias estacionarias sobre secuencias de acciones.

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots]$$
$$\Leftrightarrow$$
$$[r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

Utilidad de una secuencia de acciones

- Teorema: Si las preferencias son estacionarias únicamente existen dos maneras de definir la utilidad de una secuencia de acciones:
  – Utilidad aditiva:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$$

  – Utilidad aditiva con descuento:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

¿Utilidad infinita?

- Problema: Las secuencias de acciones infinitas pueden tener una utilidad infinita.
- Soluciones:
  - Horizonte finito. Establecemos que sólo se jugará al juego durante T unidades de tiempo. Da lugar a políticas no estacionarias (π* depende del tiempo que queda)
  - Estado absorbente: Garantizamos que para cualquier política se alcance siempre un nodo terminal.
  - Descontar: Dado un $0 < \gamma < 1$

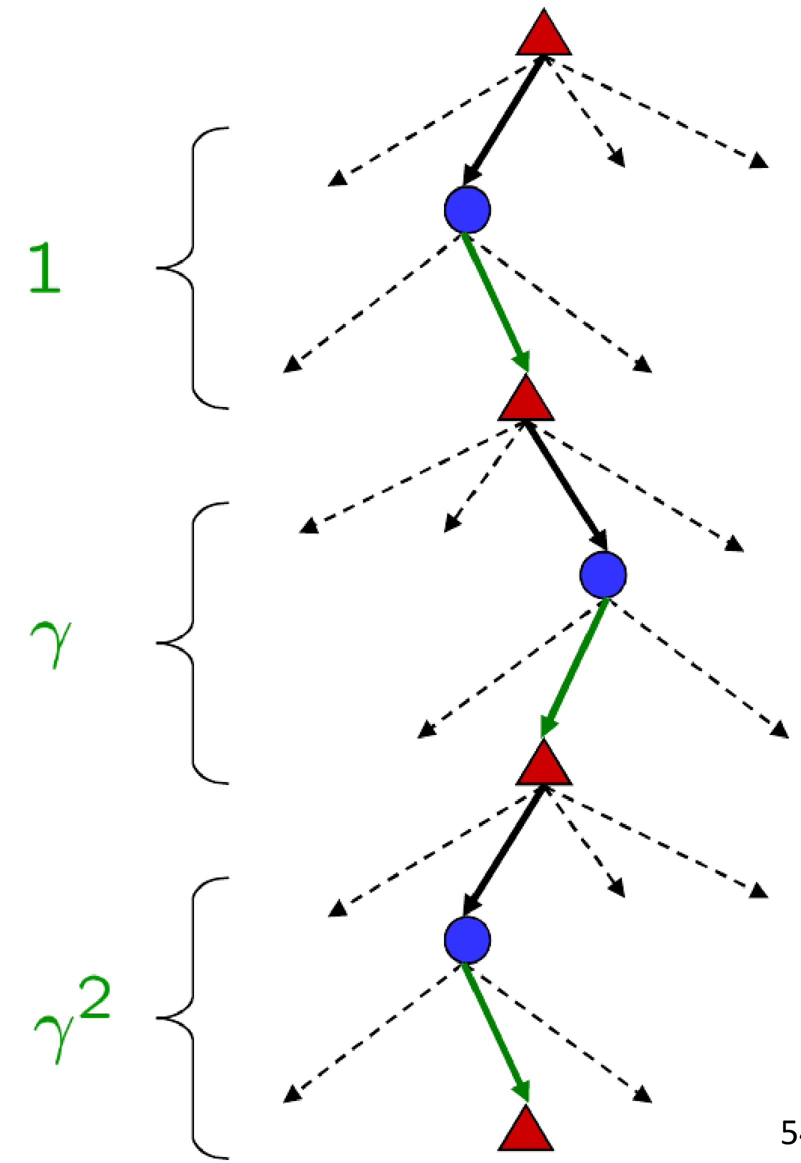
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

    - Un γ más pequeño indica que estamos mucho más interesados en los estados más inmediatos que en los que llegarán más tarde.

UNIVERSITAT DE BARCELONA

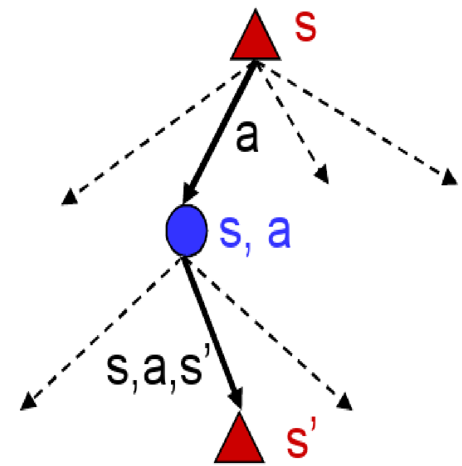
Descontar

- Normalmente se descuenta por $\gamma < 1$ cada paso de tiempo
- Las recompensas más cercanas en el tiempo tienen una utilidad mayor que las que tardarán más en llegar.
  - "Más vale pájaro en mano..."
- Ayuda a los algoritmos a converger.



Valor óptimo de un estado

- Operación fundamental: Calcular el valor óptimo de cada estado.
- V*(s) = Valor que obtendríamos si empezáramos en s y aplicáramos la política óptima



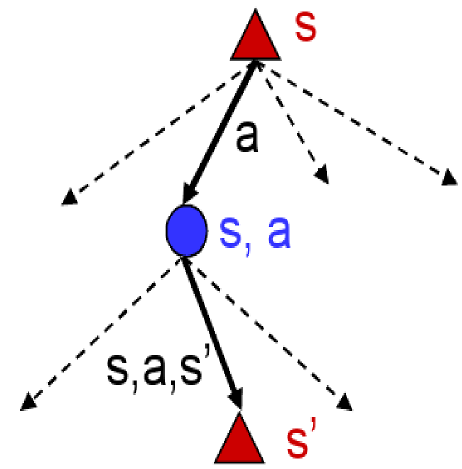
V*(s)

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

$\gamma = 1$ y $R(s) = -0.04$ para estados no terminales

Valor óptimo de un estado

- Operación fundamental: Calcular el valor óptimo de cada estado.
- V*(s) = Valor que obtendríamos si empezáramos en s y aplicáramos la política óptima
- Q*(s,a) = Valor que obtendríamos si empezáramos en s, hiciéramos la acción a y después aplicáramos la política óptima



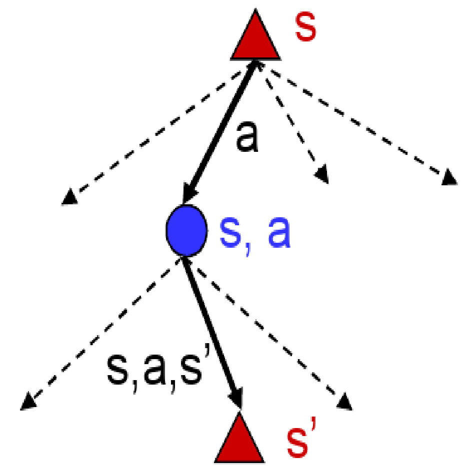
V*(s)

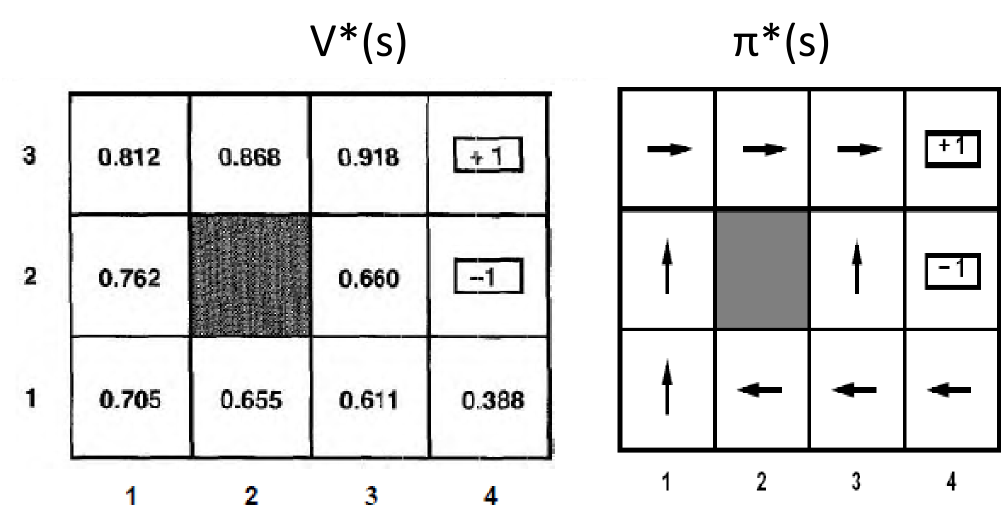
	1	2	3	4
3	0.812	0.868	0.918	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388

$\gamma = 1$ y $R(s) = -0.04$ para estados no terminales

Valor óptimo de un estado

- Operación fundamental: Calcular el valor óptimo de cada estado.
- V*(s) = Valor que obtendríamos si empezáramos en s y aplicáramos la política óptima
- Q*(s,a) = Valor que obtendríamos si empezáramos en s, hiciéramos la acción a y después aplicáramos la política óptima
- π*(s) = Acción óptima cuando nos encontramos en el estado s.

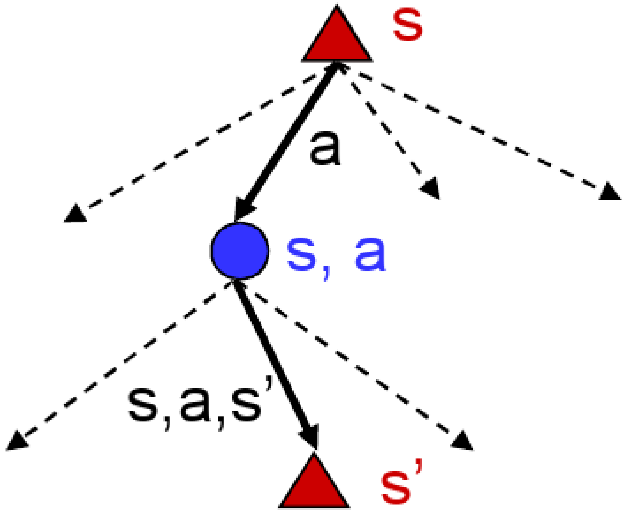




$\gamma = 1$ y $R(s) = -0.04$ para estados no terminales

Ecuaciones de Bellman

- La definición de ”utilidad óptima” nos lleva a pensar que obtenemos la recompensa óptima maximizando sobre la primera acción y siguiendo la política óptima a partir de ahí.



- Formalmente:

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

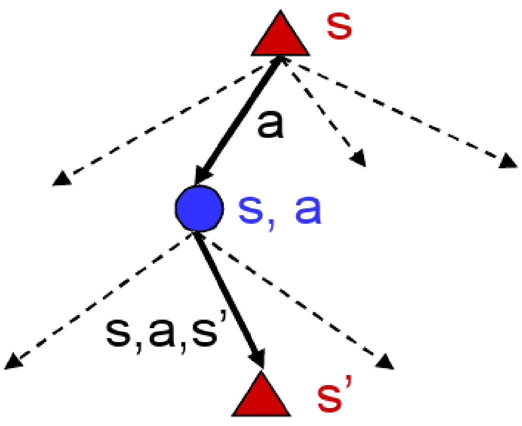
Resolviendo PDM

- Queremos encontrar una política óptima π*
- Propuesta 1: Podemos hacer una búsqueda expectimax modificada empezando en el estado s.

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

$$V^*(s) = \max_a Q^*(s,a)$$



UNIVERSITAT DE BARCELONA

Iteración de valores

- Computar los valores óptimos para todos los estados al mismo tiempo utilizando aproximaciones sucesivas.
- $V^*_0, V^*_1, \ldots, V^*_k, \ldots$
- Una vez terminamos, no necesitamos replanificar (toda la planificación se realiza offline).

Estimación de valor

- Para cada estado s calculamos estimaciones $V^*_k(s)$:
  - ¡No son el valor óptimo de s!!
  - Son el valor óptimo considerando k recompensas.
  - Cuando $k \rightarrow \infty$, se aproxima al valor óptimo

Estimación de valor

- Para cada estado s calculamos estimaciones $V^*_k(s)$:
  - ¡No son el valor óptimo de s!!
  - Son el valor óptimo considerando k recompensas.
  - Cuando $k \rightarrow \infty$, se aproxima al valor óptimo
- ¿Por qué?
  - Cuando descontamos, las recompensas que están lejos se vuelven negligibles.
  - Si desde cualquier parte se puede alcanzar un estado terminal, la fracción de episodios que no terminan se convierte en negligible.
  - En otro caso, podríamos tener utilidad infinita y la aproximación no funcionará.

Iteración de valores

- Idea (algoritmo):
  - Empezar con $V^*_0(s)=0$
  - Dado V^*_i, calcular el valor de todos los estados a profundidad i+1

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s')\right]$$

  - Ésta es la **actualización de Bellman**.
  - Repetir hasta que converja

UNIVERSITAT DE BARCELONA

Iteración de valores

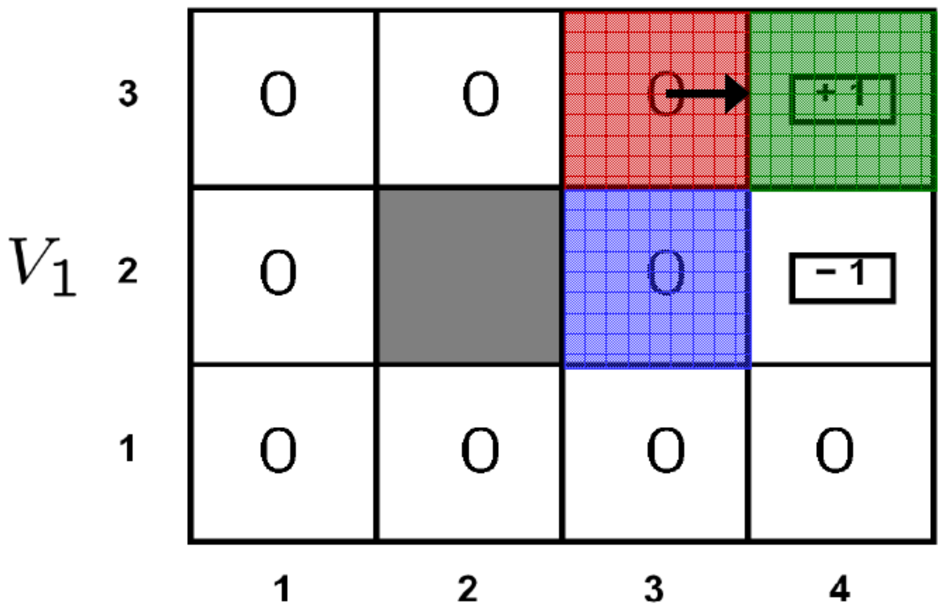
- Idea (algoritmo):
  - Empezar con $V^*_0(s)=0$
  - Dado V^*_i, calcular el valor de todos los estados a profundidad i+1

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s')\right]$$

  - Esta es la **actualización de Bellman.**
  - Repetir hasta que converja
- Teorema: El algoritmo converge a valores óptimos únicos.
  - Nota: Las políticas pueden converger mucho antes de que lo hagan los valores

Ecuaciones de Bellman: Ejemplo

(extraído del curso de Dan Klein – UC Berkeley)



$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Cálculo de V_2 para $s=\langle 3,3 \rangle$:

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9\, V_1(s') \right]$$

max happens for a=right, other actions not shown

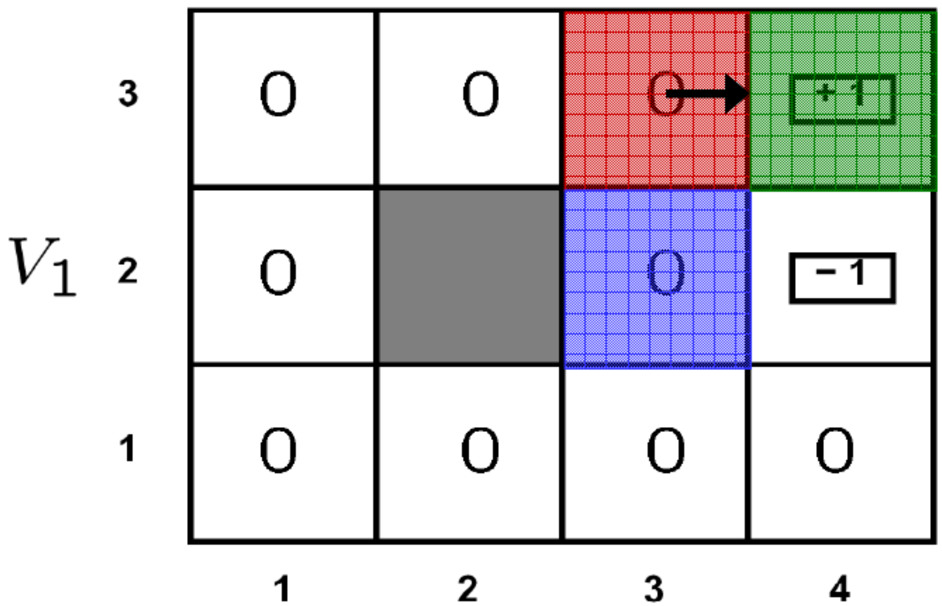
$$= 0.9\, [0.8 \cdot 1$$

$s'=\langle 4,3 \rangle$

Example: γ=0.9, living reward=0, noise=0.2

Ecuaciones de Bellman: Ejemplo

(extraído del curso de Dan Klein – UC Berkeley)



$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Cálculo de V_2 para $s=\langle 3,3\rangle$:

$$V_2(\langle 3,3\rangle) = \sum_{s'} T(\langle 3,3\rangle, \text{right}, s') \left[R(\langle 3,3\rangle) + 0.9\, V_1(s') \right]$$

max happens for a=right, other actions not shown

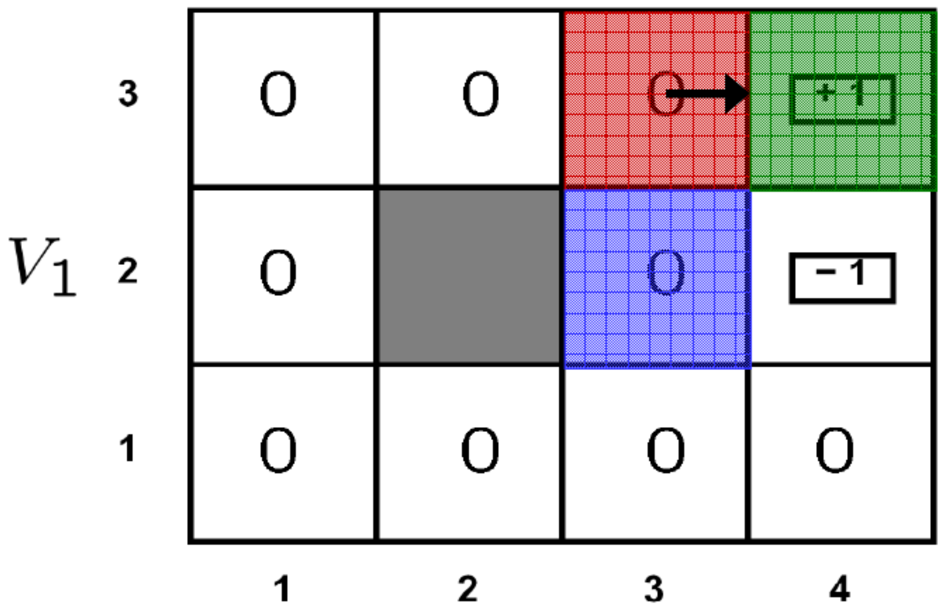
$$= 0.9\, [0.8 \cdot 1 + 0.1 \cdot 0$$

s'=⟨3,2⟩

Example: γ=0.9, living reward=0, noise=0.2

Ecuaciones de Bellman: Ejemplo

(extraído del curso de Dan Klein – UC Berkeley)



$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Cálculo de V_2 para $s=\langle 3,3 \rangle$:

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9\, V_1(s') \right]$$

max happens for a=right, other actions not shown

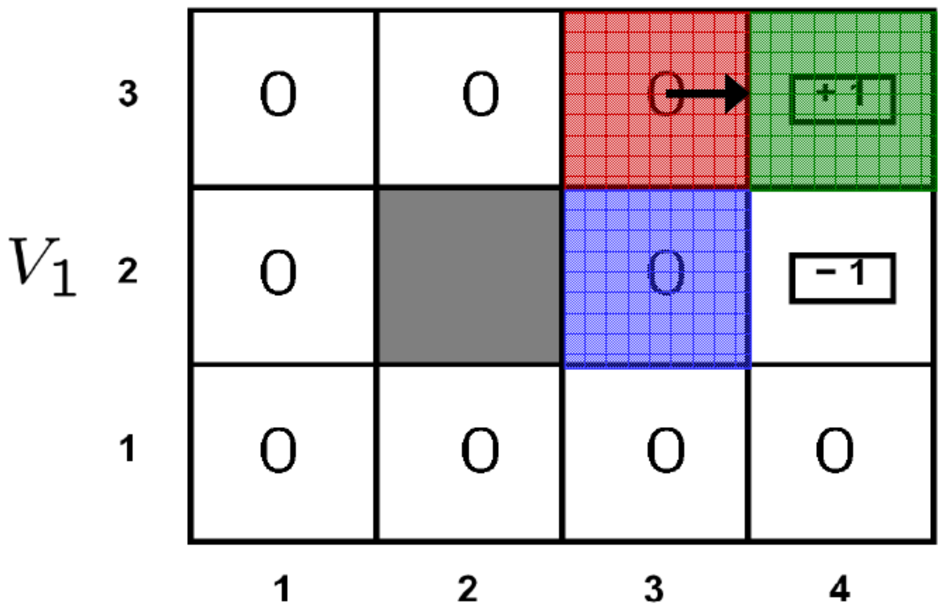
$$= 0.9\, [0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

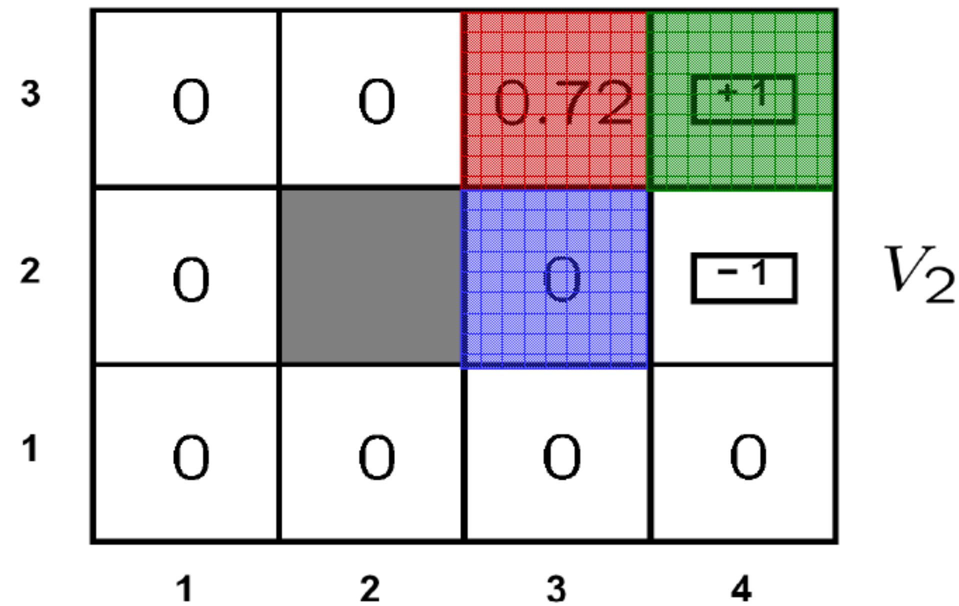
$s'=\langle 3,3 \rangle$

Example: γ=0.9, living reward=0, noise=0.2

Ecuaciones de Bellman: Ejemplo

(extraído del curso de Dan Klein – UC Berkeley)





$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Cálculo de V_2 para $s=\langle 3,3 \rangle$:

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9\, V_1(s') \right]$$

max happens for a=right, other actions not shown

$$= 0.9\,[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0] = 0.72$$

Example: γ=0.9, living reward=0, noise=0.2

Cada iteración i se calcula el $V_{i+1}(s)$ de todos los estados s

Ejemplo: Iteración de valores

V_2

	1	2	3	4
3	0	0	0.72	+1
2	0		0	−1
1	0	0	0	0

V_3

	1	2	3	4
3	0	0.52	0.78	+1
2	0		0.43	−1
1	0	0	0	0

- La información se propaga hacia atrás desde los estados terminales. Al final todos los estados tienen estimaciones de valor correctas

Iteración de valores

(libro Russell and Norvig: capítulo 16)

Section 16.2 Algorithms for MDPs

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
      rewards $R(s, a, s')$, discount γ
      ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
      δ, the maximum relative change in the utility of any state

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U)
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta \leq \epsilon(1-\gamma)/\gamma$
  return U

function Q-VALUE(mdp, s, a, U) **returns** a utility value
  return $\sum_{s'} \boldsymbol{P(s'|s,a)[R(s,a,s') + \gamma\, U[s']]}$

Figure 16.6 The value iteration algorithm for calculating utilities of states. The termination condition is from Equation (16.12).

Iteración de valores
(e-libro Sutton and Barto, 2nd ed.)

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

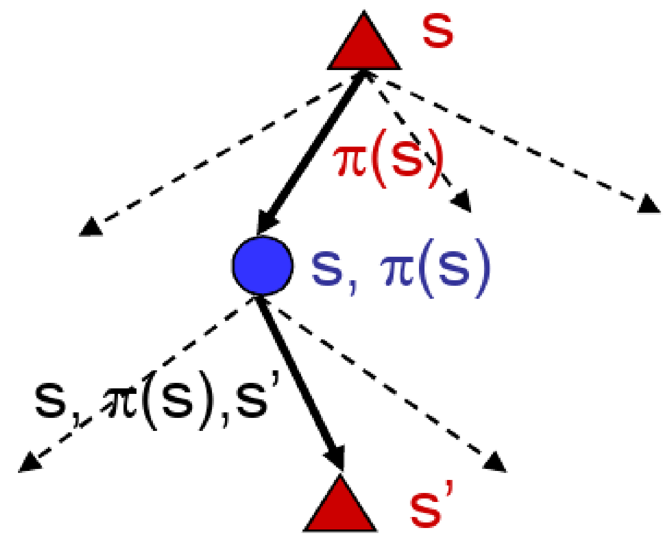
Repeat
$\quad \Delta \leftarrow 0$
$\quad$ For each $s \in \mathcal{S}$:
$\quad\quad v \leftarrow V(s)$
$\quad\quad V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
$\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

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

Figure 4.5: Value iteration.

Evaluación de políticas

- Otra operación básica: calcular el valor de un estado en una política (en general, no necesariamente óptima) dada.
- $V^{\pi}(s)$ = Suma total de recompensas esperadas a partir de s si seguimos la política π.



$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

Evaluación de políticas

- ¿Cómo calculamos las V's para una política π determinada?
- Idea 1: Modifiquemos las ecuaciones de Bellman

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^\pi(s')]$$

- Es simplemente un sistema lineal, se puede resolver con Matlab (por ejemplo).

Iteración de políticas

- Problemas de la iteración de valores:
  - Considerar todas las acciones en cada iteración es lento. Tarda |A| veces más tiempo que la evaluación de una política
  - En ese tiempo la política no cambia... tiempo perdido.
- Alternativa a la iteración de valores:
  - Iteración de políticas:
    - Repetir hasta que converja:
      - Paso 1: Evaluar una política
      - Paso 2: Mejorar la política

Iteración de políticas

- Repetir hasta que converja la política
  - Evaluación de política: Dada la política actual π encontrar los valores asociados aplicando las ecuaciones de Bellman simplificadas de forma iterativa

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s') \left[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

  - Mejora de política: Determinar una nueva política a partir de los valores encontrados en el paso anterior
  - $$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

Comparación

- Iteración de valores:
  - Cada paso modifica tanto los valores de cada estado (explícitamente) como la política.
- Iteración de políticas:
  - Varias iteraciones para calcular las utilidades de una política determinada.
  - Cada iteración se modifica la política.

$$\pi_0 \xrightarrow{\mathrm{E}} V^{\pi_0} \xrightarrow{\mathrm{I}} \pi_1 \xrightarrow{\mathrm{E}} V^{\pi_1} \xrightarrow{\mathrm{I}} \pi_2 \xrightarrow{\mathrm{E}} \cdots \xrightarrow{\mathrm{I}} \pi^* \xrightarrow{\mathrm{E}} V^*,$$

Iteración de políticas
(libro Russell and Norvig)

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, transition model T
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ *POLICY-EVALUATION*(π, U, mdp)
    unchanged? $\leftarrow$ true
    for each state s **in** S **do**
      if $\max_a \sum_{s'} T(s,a,s')\, U[s'] > \sum_{s'} T(s,\pi[s],s)U[s']$ **then**
        $\pi[s] \leftarrow \text{argmax}_a \sum_{s'} T(s,a,s)\, U[s']$
        unchanged? $\leftarrow$ false
  until *unchanged?*
  return n

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

Iteración de políticas
(e-libro Sutton and Barto)

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

Figure 4.3: Policy iteration (using iterative policy evaluation) for V^*. In the "$\arg\max$" step in 3, it is assumed that ties are broken in a consistent order.

Algunas consideraciones

- Notación:
  - en las transparencias casi siempre se considera R(s,a,s'):
    - R: S x A x S $\rightarrow \Re$
  - en la 2ª ed. del libro de AI (Russell and Norvig) se dice R(s)
    - R: S $\rightarrow \Re$
    - "*does not change the problem in any fundamental way*"
  - U(s) para denotar V(s)
    - Utilidades de los estados (es decir, Valores)