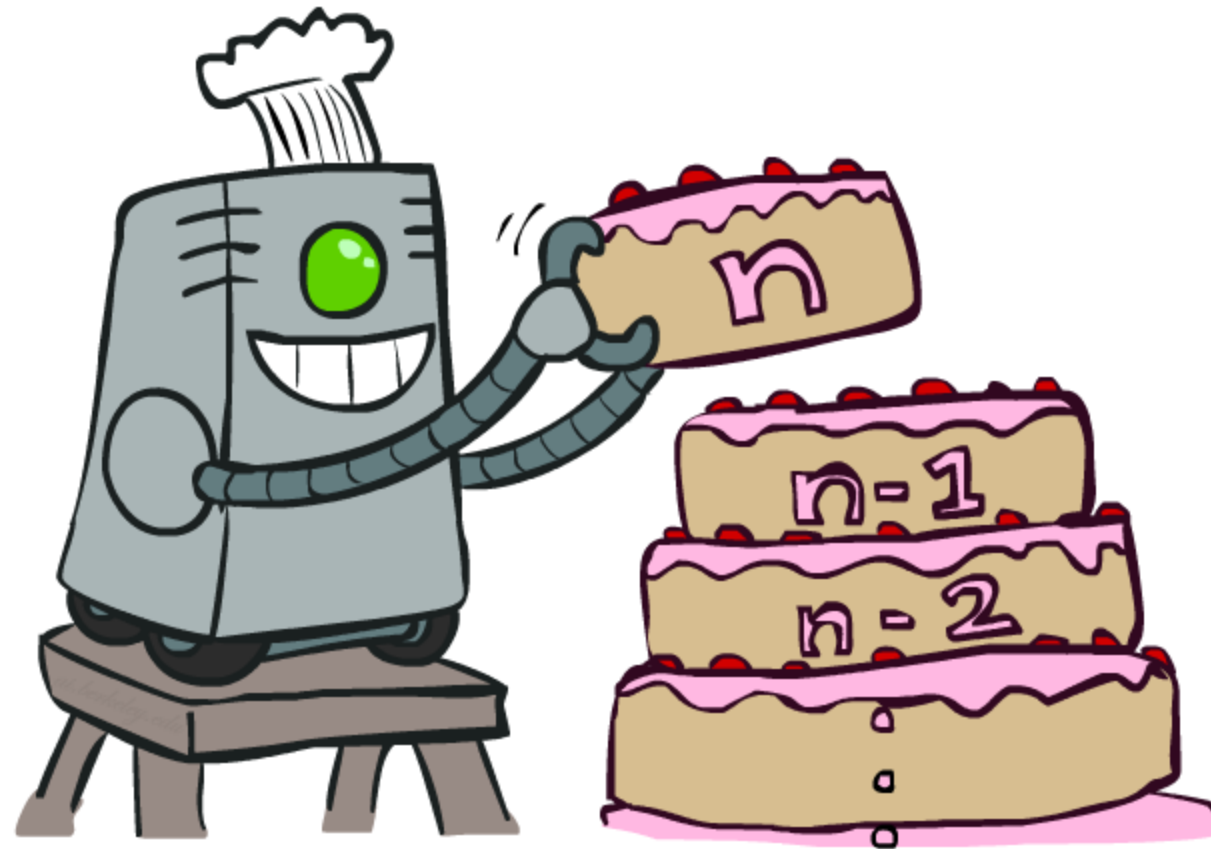


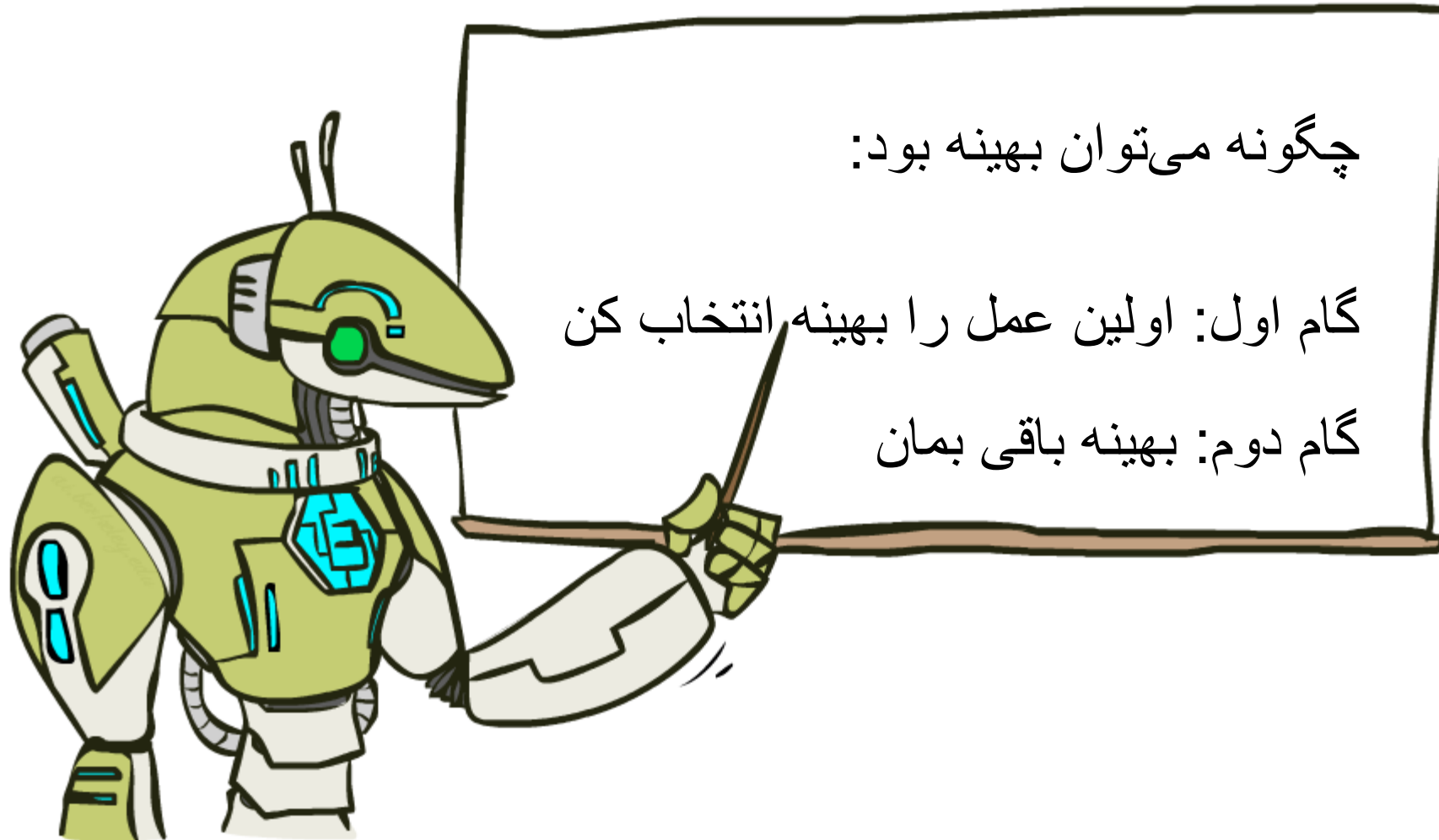
یادگیری تقویتی (Reinforcement Learning)

1. فرایند تصمیم مارکوف (Markov Decision Processes)
2. الگوریتم تکرار مقدار (Value Iteration)
3. الگوریتم تکرار سیاست (Policy Iteration)
4. یادگیری تقویتی (Reinforcement Learning)

الڱورٽم تڪرار مقدار (Value Iteration)



معادلات بلمن (The Bellman Equations)



الگوریتم Value Iteration

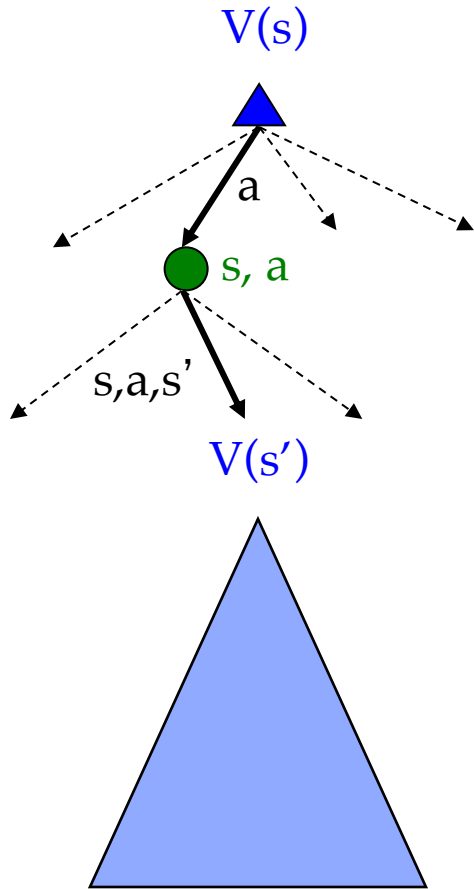
□ معادلات بلمن (Bellman) مقادیر بهینه را مشخص می کنند:

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

□ تکرار مقدار (Value Iteration) این مقادیر را محاسبه می کند.

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

“Bellman Update”



الگوریتم Value Iteration

□ با بردار $V_0(s) = 0$ شروع کن. یعنی وقتی هیچ گامی (time step) باقی نمانده، مجموع پاداش مورد انتظار صفر است.

□ با داشتن بردار $V_k(s)$ ، برای هر حالت $V_{k+1}(s)$ را به دست بیاور:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

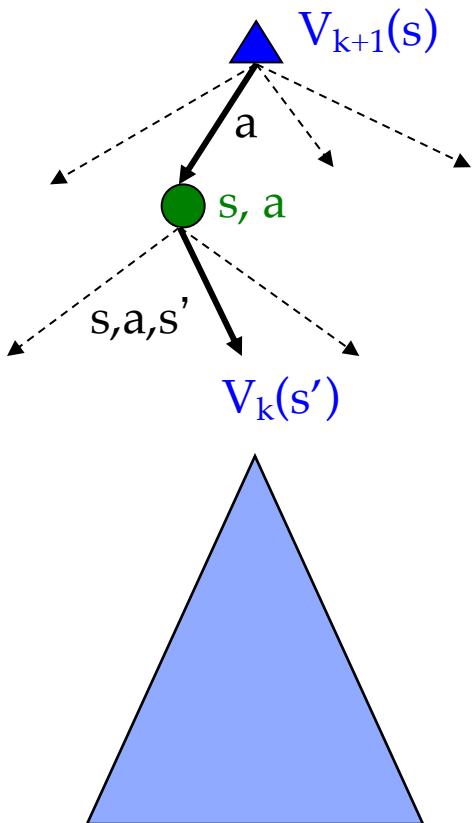
▪ $V = B(V)$ Where B is the Bellman update operator

□ تکرار کن تا همگرایی حاصل شود، که در نهایت منجر به V^* می شود.

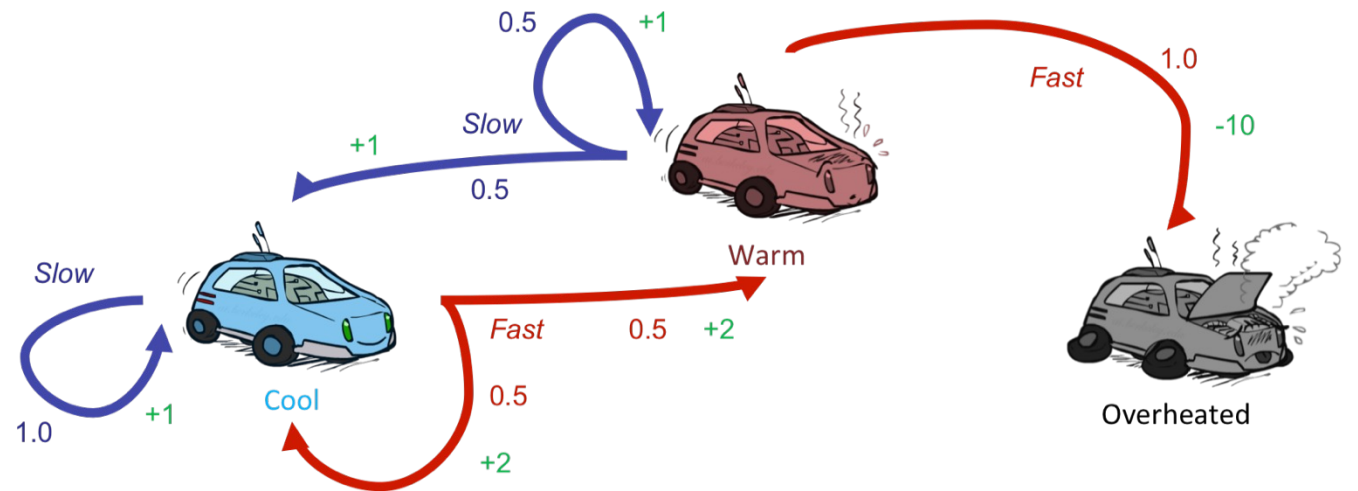
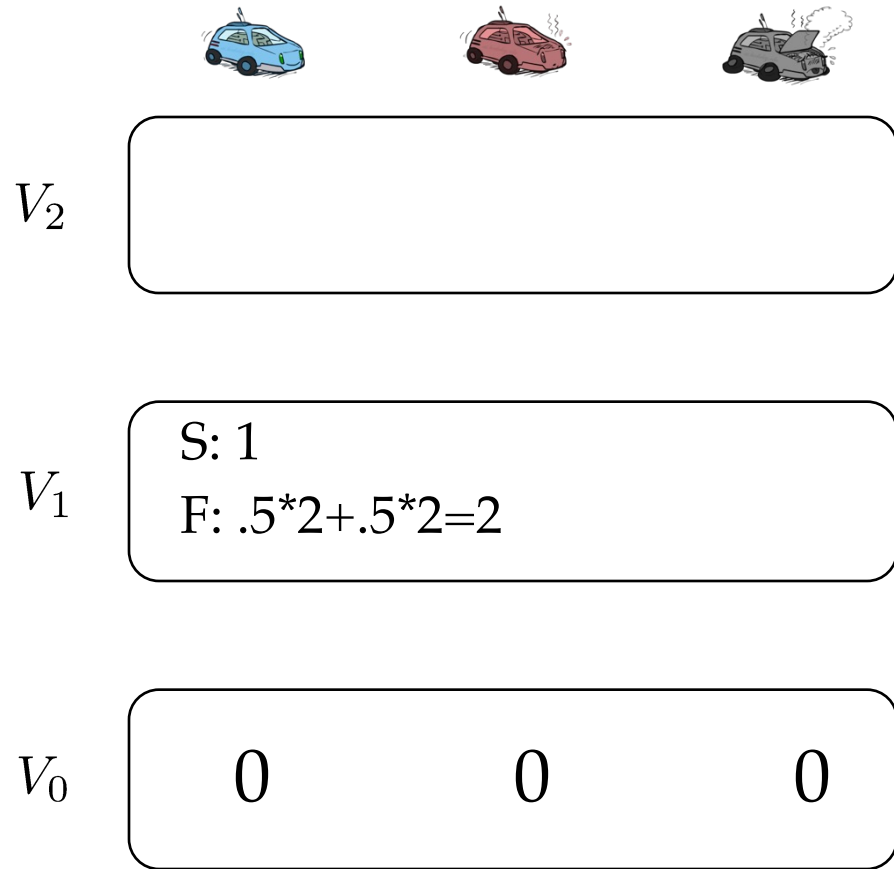
□ پیچیدگی زمانی هر تکرار: $O(S^2A)$

□ قضیه: این الگوریتم به یک مجموعه ی یکتا از مقادیر بهینه همگرا می شود:

- ایده اصلی: تقریب ها به تدریج به سمت مقادیر بهینه اصلاح می شوند.
- سیاست (Policy) ممکن است خیلی زودتر از مقادیر (Values) همگرا شود






مثال : Value Iteration

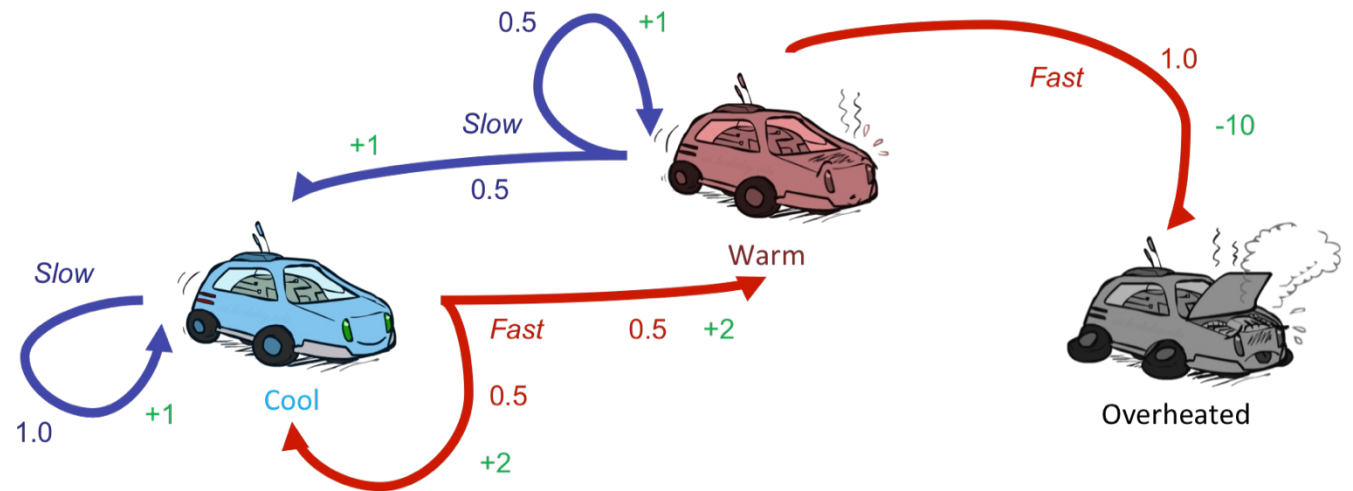


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

مثال : Value Iteration




			
V_2			
V_1	2	S: $.5*1+.5*1=1$ F: -10	
V_0	0	0	0

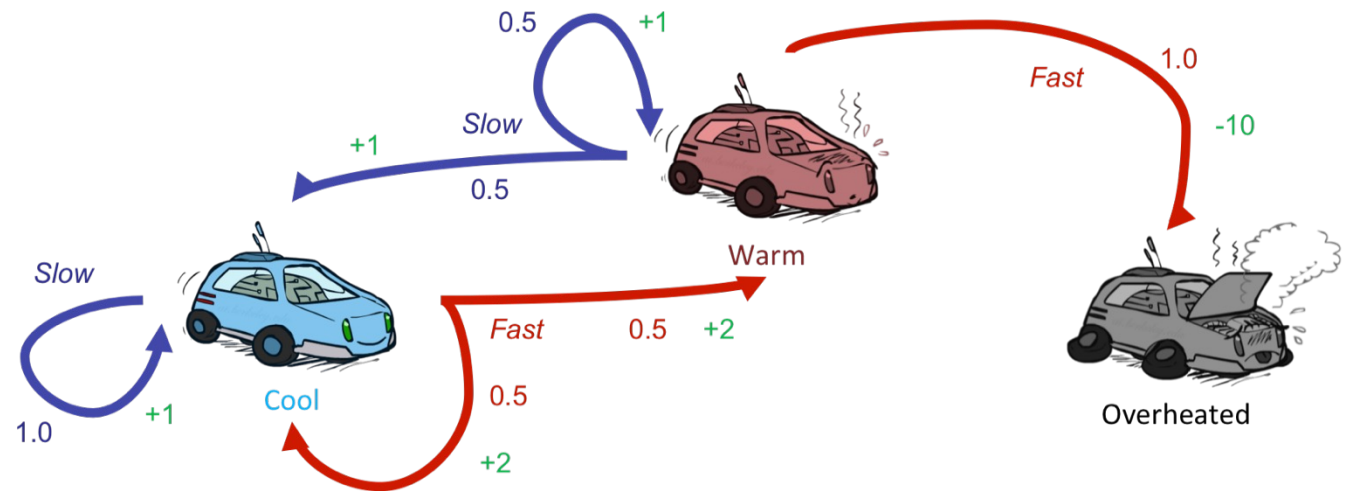


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

مثال : Value Iteration

			
V_2			
V_1	2	1	0
V_0	0	0	0



Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

مثال : Value Iteration



V_2

S: $1+2=3$

F: $.5*(2+2)+.5*(2+1)=3.5$

V_1

2

1

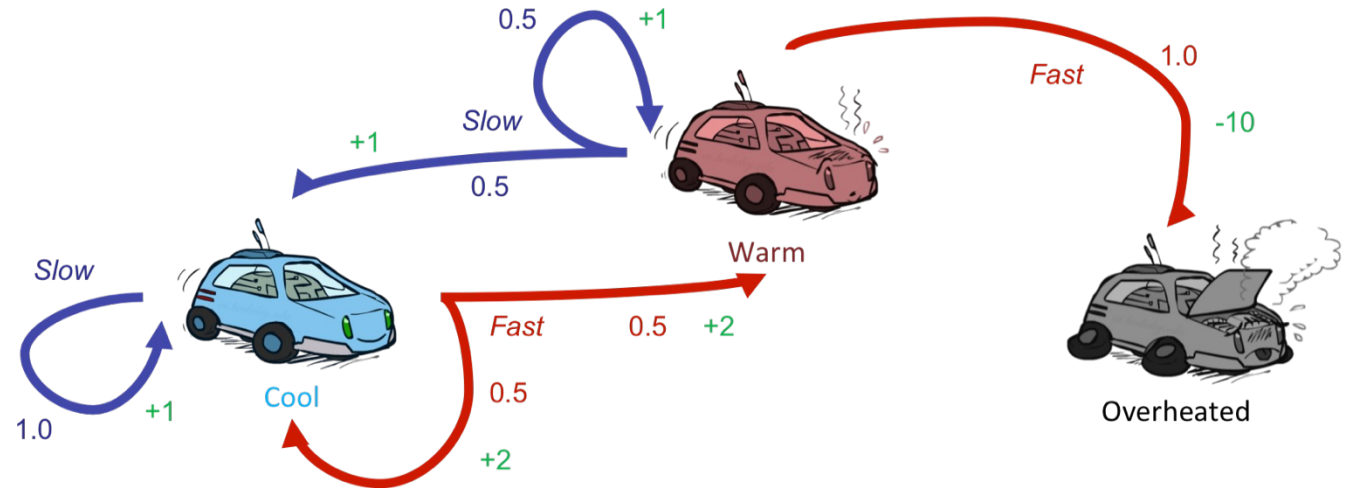
0

V_0

0

0




0

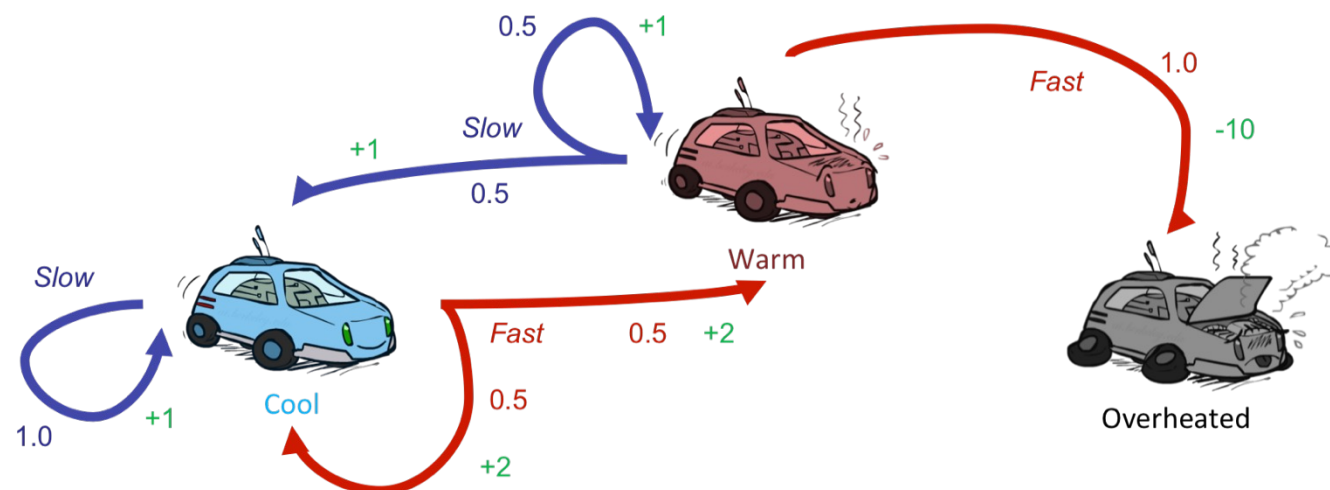


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

مثال : Value Iteration

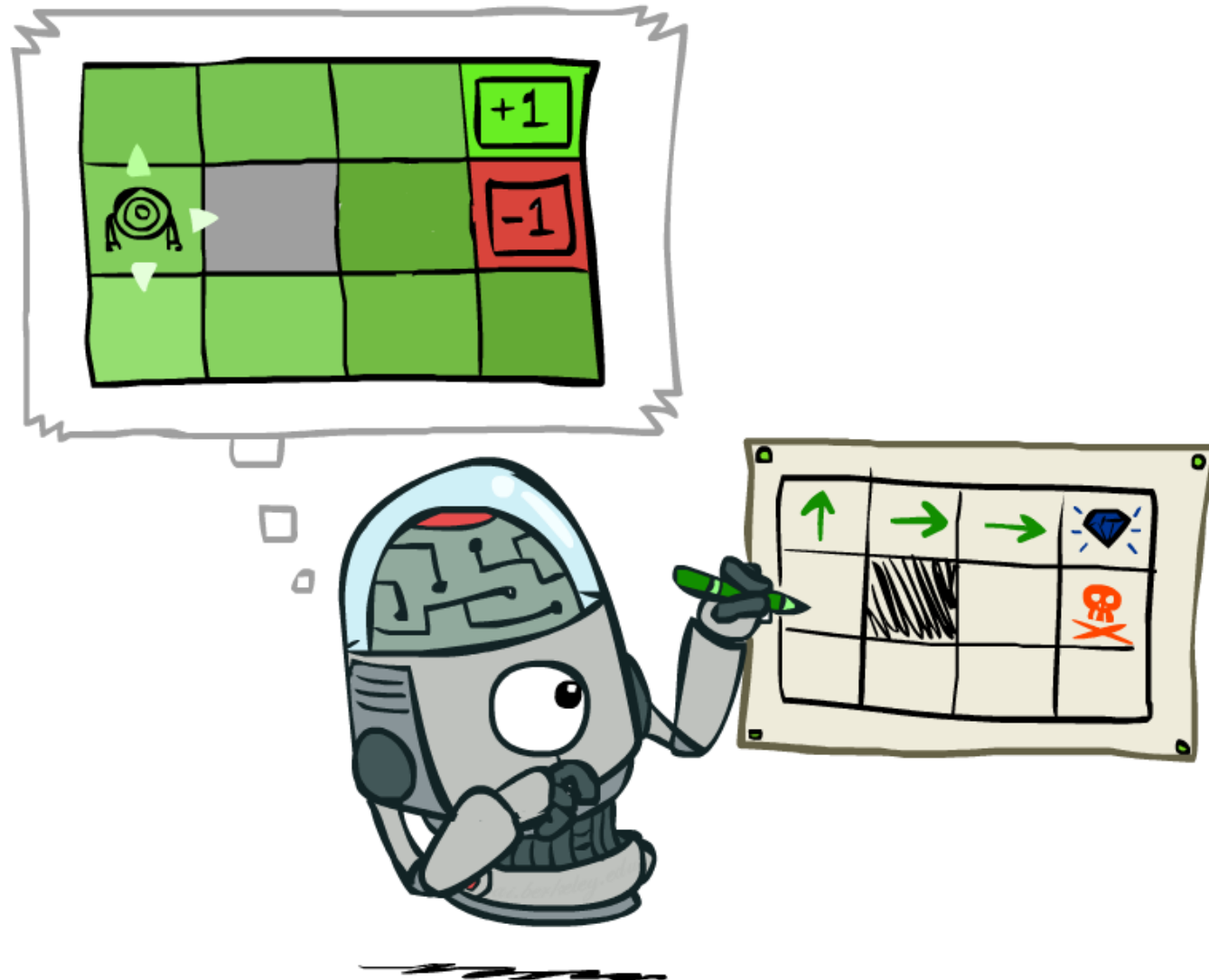
			
V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0



Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

استخراج سیاست



محاسبه‌ی عمل‌ها از روی ارزش حالت‌ها

0.95 ▶	0.96 ▶	0.98 ▶	1.00
▲ 0.94		◀ 0.89	-1.00
▲ 0.92	◀ 0.91	◀ 0.90	0.80 ▼

❑ فرض کنید ارزش بهینه‌ی حالت‌ها را در اختیار داریم $V^*(s)$

❑ در هر حالت باید چه عملی را انجام دهیم؟
▪ خیلی واضح نیست!

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

❑ این مرحله را استخراج سیاست (policy extraction) می‌نامند، زیرا در این مرحله سیاست عامل از روی ارزش حالت‌ها (values) به دست می‌آید.

محاسبه‌ی عمل از روی ارزش حالت‌های Q

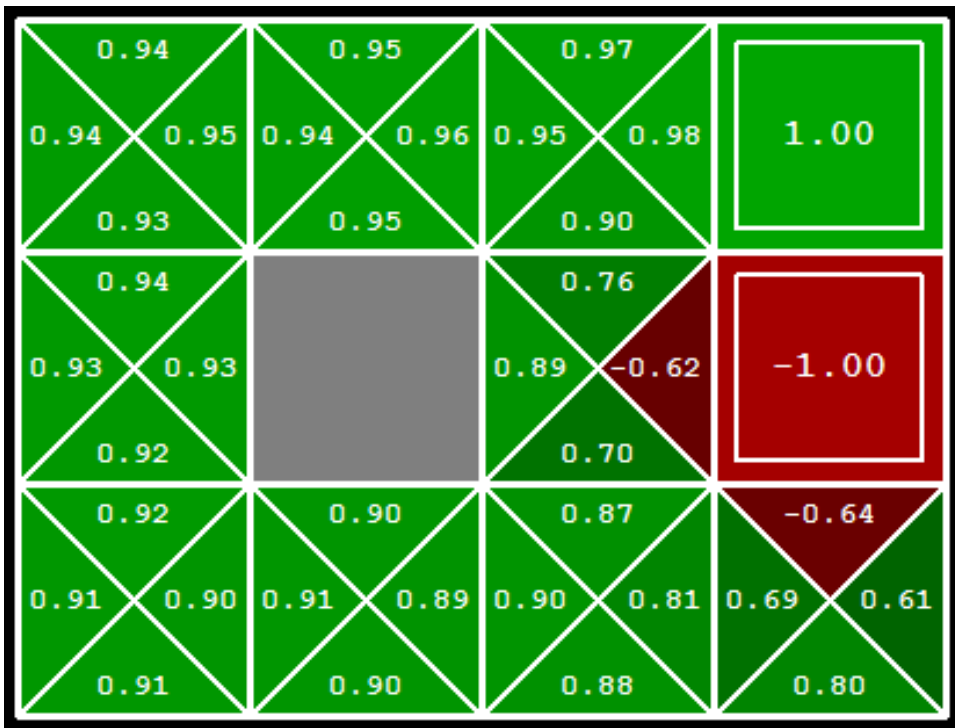
❑ فرض کنید ارزش بهینه‌ی حالت‌های Q را در اختیار داریم

❑ در هر حالت باید چه عملی را انجام دهیم؟

▪ بسیار ساده است!

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

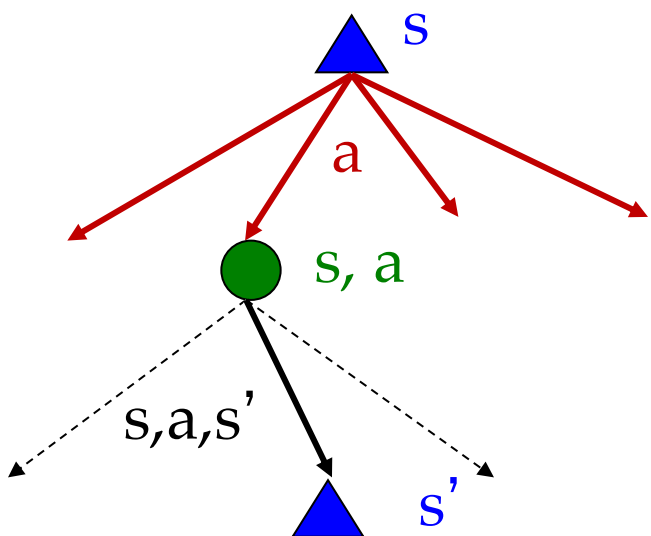
❑ یک درس مهم: انتخاب عمل از روی مقادیر q ساده‌تر است.



مشکلات الگوریتم تکرار مقدار

□ الگوریتم تکرار مقدار، معادله ی بلمن را تکرار می کند:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$



■ مشکل ۱. این الگوریتم کند است - هر تکرار $O(S^2A)$

■ مشکل ۲. مقدار «ماکزیمم» در هر حالت به ندرت تغییر می کند.

■ مشکل ۳. در اغلب موارد، سیاست بسیار سریعتر از ارزش حالت ها همگرا می شود.

k=12



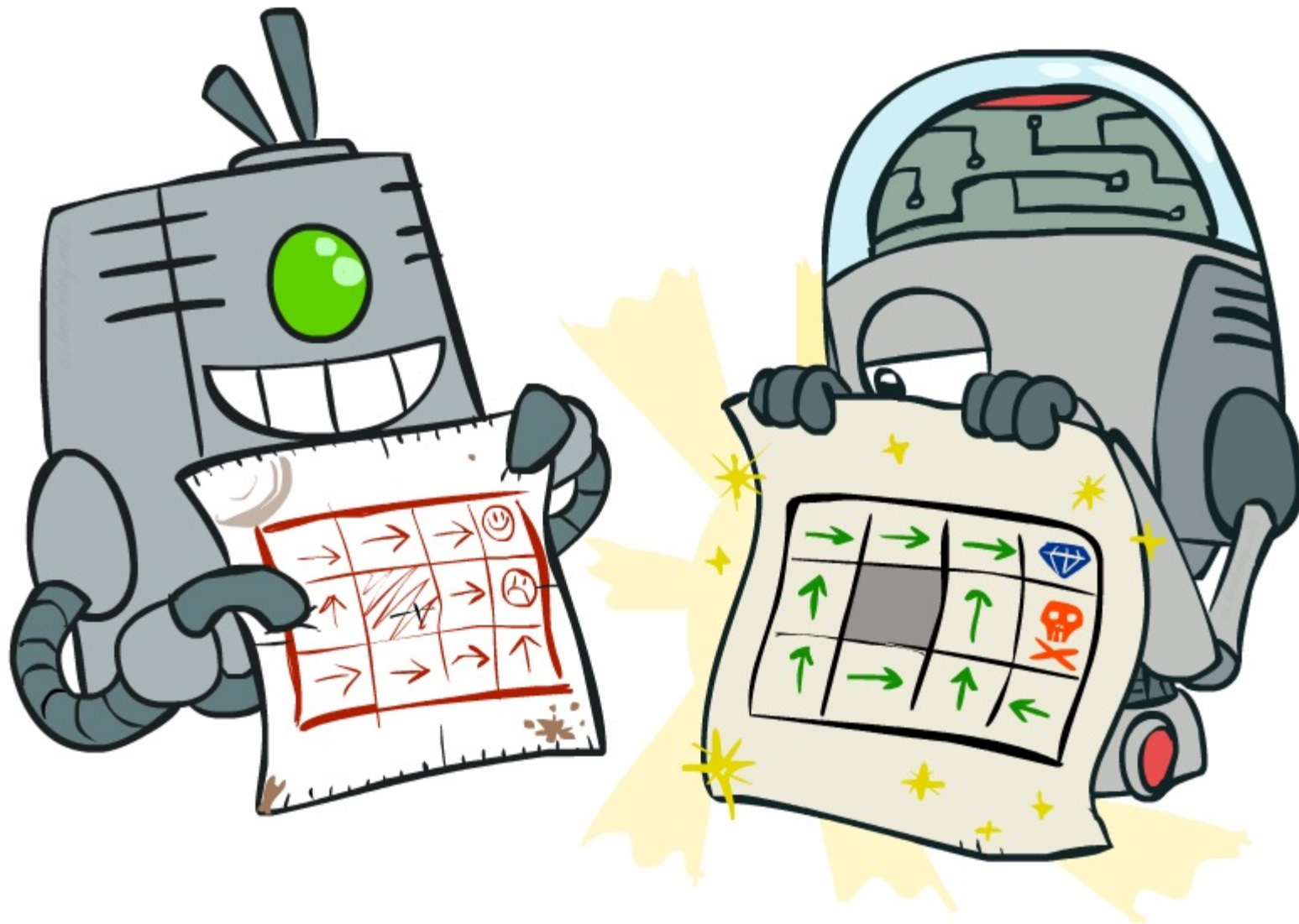
•/۲ = نويز
•/۹ = ضريب کاهش
• = پاداش مرحله‌ای

$k=100$



•/۲ = نویز
•/۹ = ضریب کاهش
• = پاداش مرحله‌ای

روش‌های مبتنی بر سیاست (Policy)



الگوریتم تکرار سیاست (Policy Iteration)

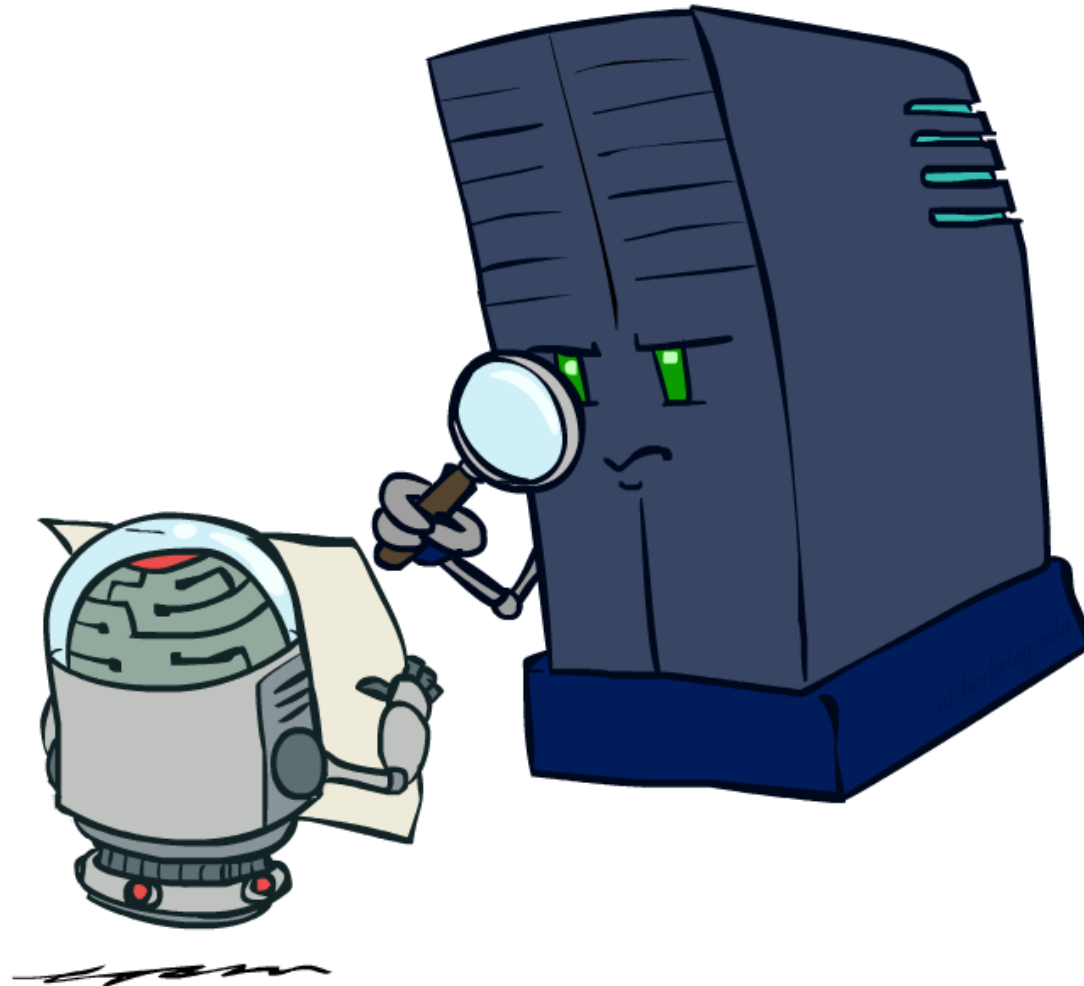
□ رویکرد جایگزین برای محاسبه مقادیر بهینه:

- **گام ۱: ارزیابی سیاست (Policy Evaluation):** محاسبه سودمندی‌ها (utilities) برای یک سیاست ثابت (نه سودمندی‌های بهینه!) تا زمان همگرایی.
- **گام ۲: بهبود سیاست (Policy Improvement):** به‌روزرسانی سیاست با استفاده از نگاه یک‌گام به جلو (one-step look-ahead) و استفاده از سودمندی‌های همگراشده (اما هنوز نه بهینه!) به عنوان مقادیر آینده.
- این مراحل تکرار می‌شود تا سیاست همگرا شود.

□ این فرآیند **تکرار سیاست (Policy Iteration)** نام دارد.

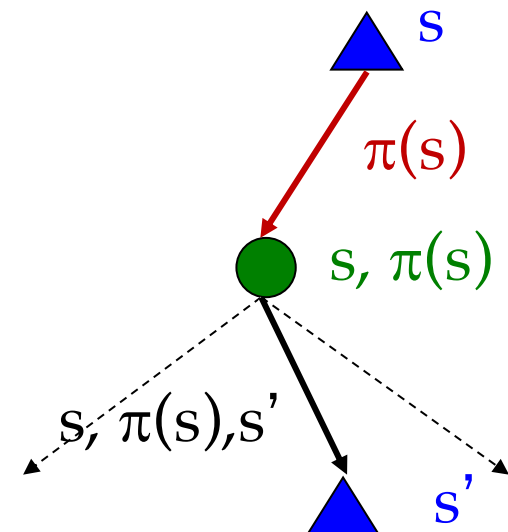
- همچنان به جواب بهینه می‌رسد!
- در برخی شرایط می‌تواند (خیلی) سریع‌تر همگرا شود.

ارزیابی سیاست (Policy Evaluation)

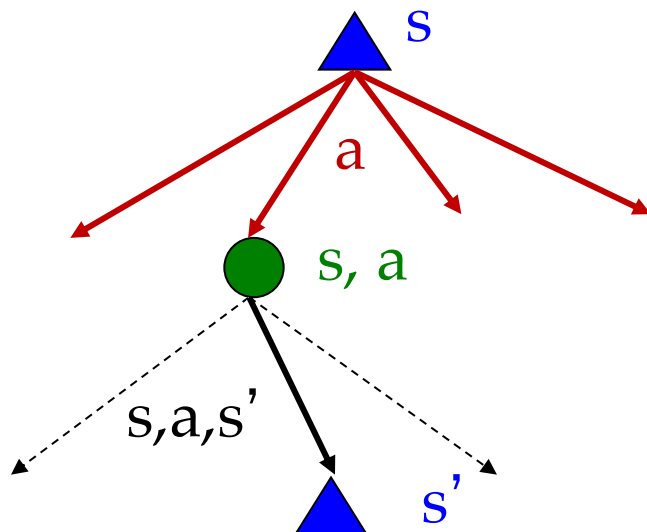


سیاست‌های ثابت

بر طبق سیاست π عمل کن



عمل بهینه را انجام بده



□ اگر از سیاست ثابت $\pi(s)$ استفاده کنیم، درخت ساده‌تر می‌شود. - یک عمل به ازای هر حالت.
▪ ... بنابراین ارزش حالت‌ها در درخت بستگی به سیاست انتخاب شده دارد.

محاسبه‌ی سودمندی برای سیاست های ثابت

□ یک عمل پایه‌ای دیگر :

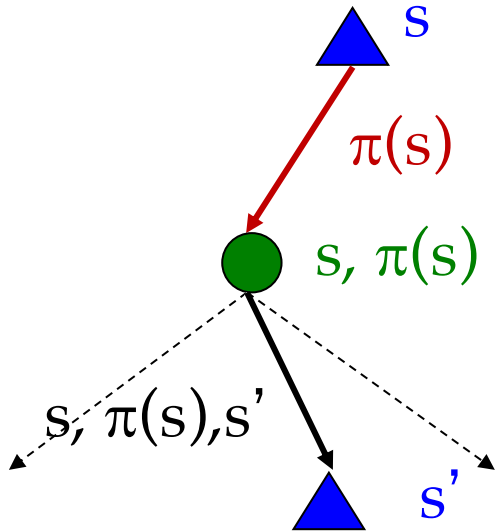
▪ محاسبه‌ی ارزش حالت s تحت یک سیاست ثابت (معمولا غیر بهینه)

□ تعریف ارزش حالت s تحت سیاست ثابت π :

▪ $V^\pi(s)$ = سودمندی موردانتظار با شروع از s و دنبال کردن π

□ رابطه بازگشتی (نگاه یک گام به جلو / معادله بلمن):

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



ارزیابی سیاست

□ چگونه مقادیر V را برای یک سیاست ثابت π محاسبه کنیم؟

□ ایده ۱: معادلات بازگشتی بلمن را به صورت فرمول‌های به‌روزرسانی تبدیل کن (مثل value iteration)

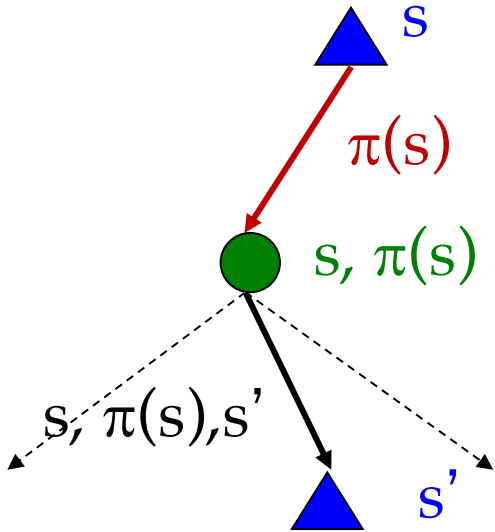
$$V_0^\pi(s) = 0$$

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

▪ پیچیدگی زمانی هر تکرار: $O(S^2)$

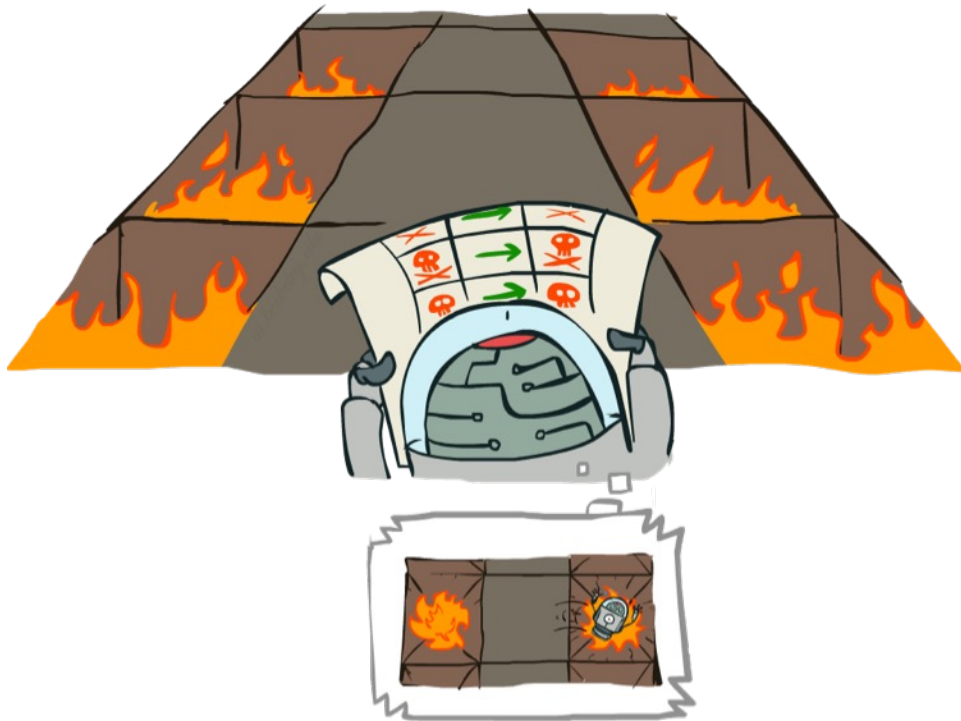
□ ایده ۲: وقتی ماکزیمم‌گیری وجود ندارد، معادلات بلمن فقط یک دستگاه معادلات خطی هستند.

▪ این دستگاه مستقیم قابل حل است!

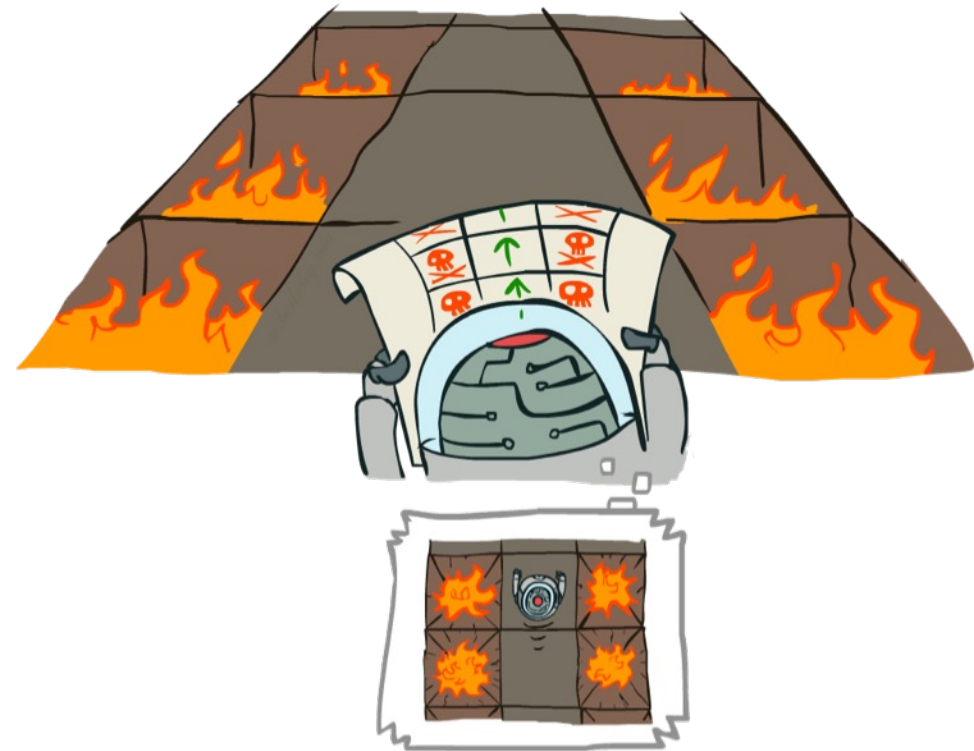


مثال: ارزیابی سیاست

همیشه به راست برو



همیشه مستقیم برو



مثال: ارزیابی سیاست

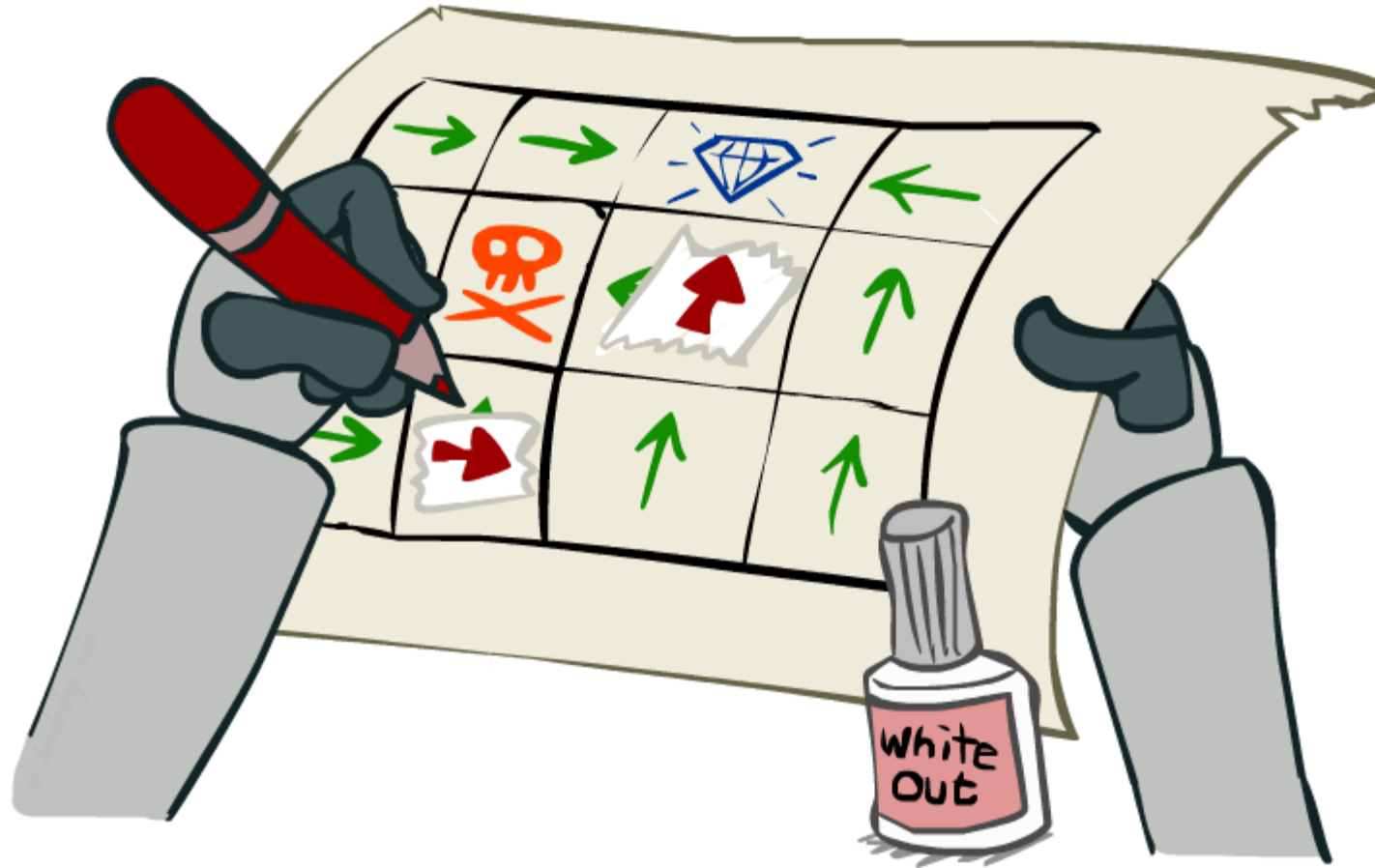
همیشه مستقیم برو

-10.00	100.00	-10.00
-10.00	70.20 ▲	-10.00
-10.00	48.74 ▲	-10.00
-10.00	33.30 ▲	-10.00

همیشه به راست برو

-10.00	100.00	-10.00
-10.00	1.09 ►	-10.00
-10.00	-7.88 ►	-10.00
-10.00	-8.69 ►	-10.00

الگوریتم تکرار سیاست (Policy Iteration)



الگوریتم تکرار سیاست (Policy Iteration)

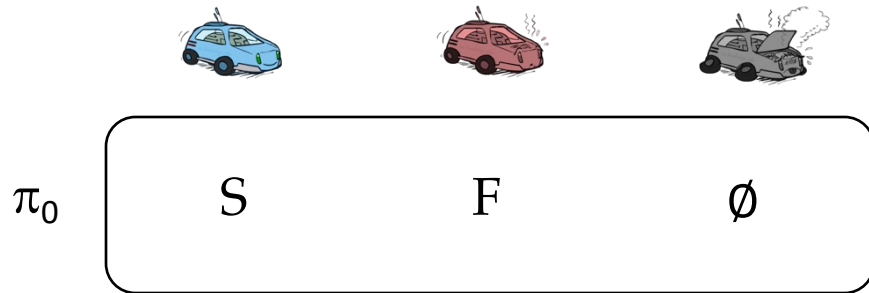
□ **ارزیابی:** برای سیاست ثابت فعلی π مقادیر را با استفاده از ارزیابی سیاست به دست آورید:
▪ تکرار کن تا ارزش ها همگرا شوند.

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') [R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')]$$

□ **بهبود:** برای مقادیر ثابت، با استفاده از استخراج سیاست، سیاست بهتری به دست آور:
▪ نگاه یک گام به جلو.

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



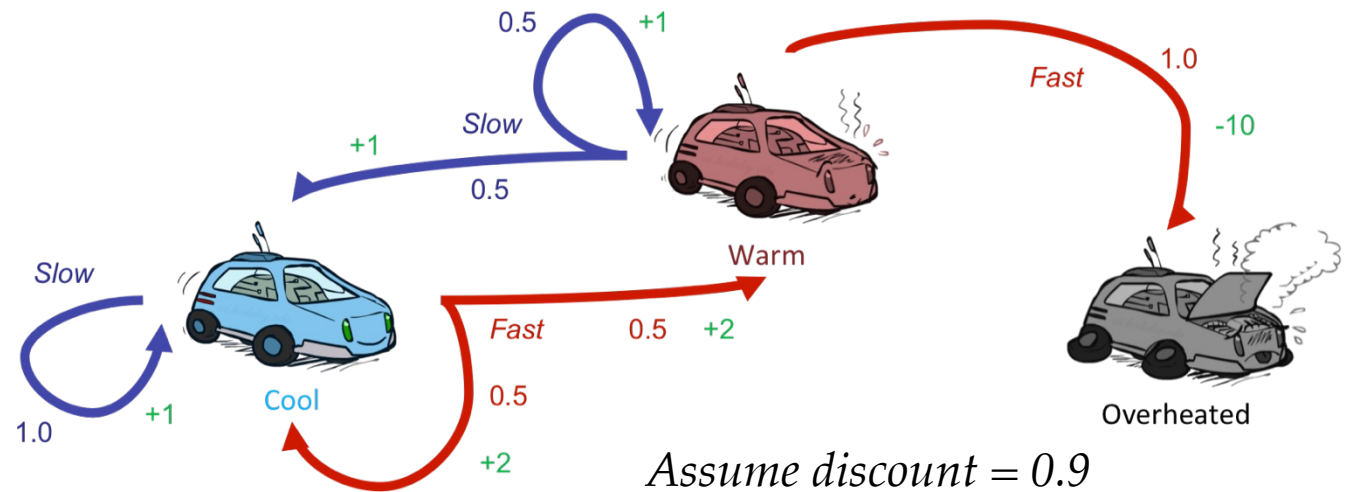
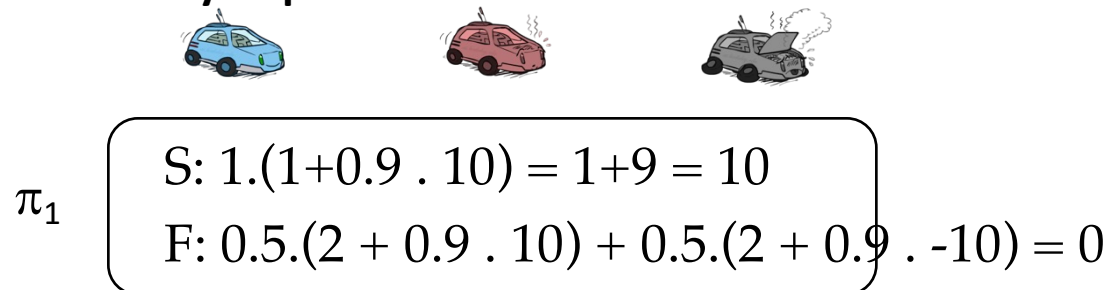
Policy Evaluation:

$$V^{\pi_0}(C) = 1 + 0.9 \cdot V^{\pi_0}(C) \Rightarrow V^{\pi_0}(C) = 10$$

$$V^{\pi_0}(W) = -10 + 0.9 \cdot V^{\pi_0}(O) \Rightarrow V^{\pi_0}(W) = -10$$

$$V^{\pi_0}(O) = 0$$

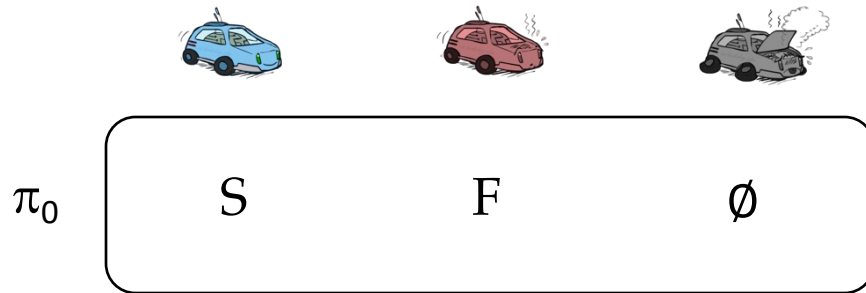
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



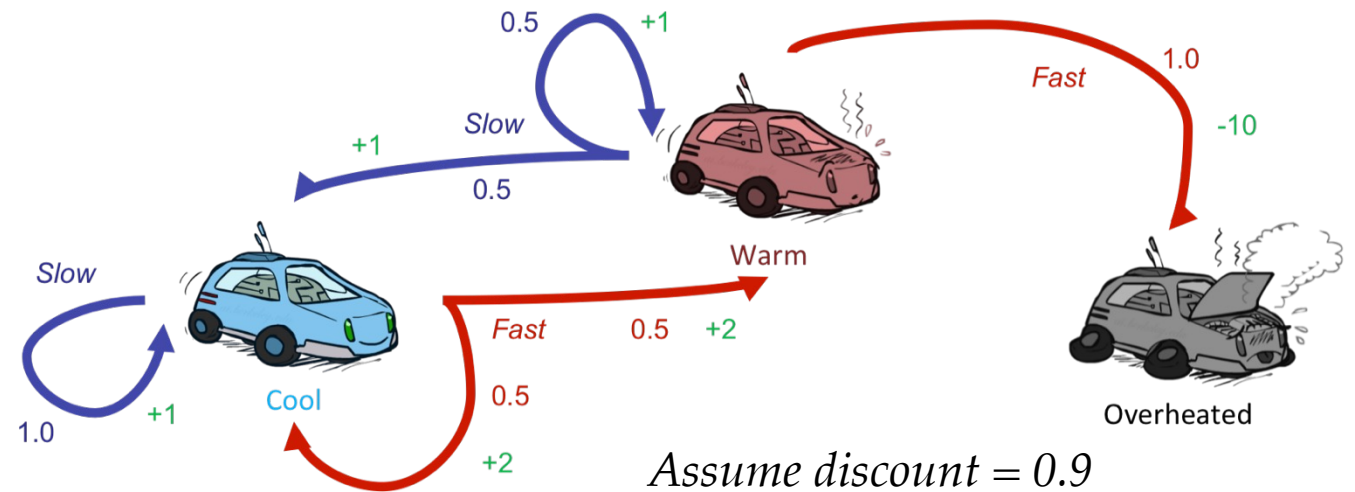
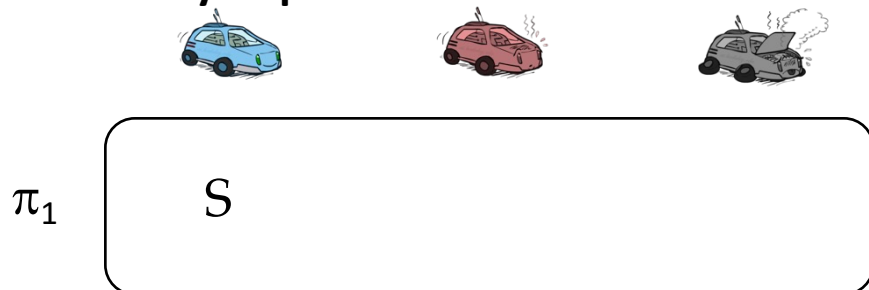
Policy Evaluation:

$$V^{\pi_0}(C) = 1 + 0.9 \cdot V^{\pi_0}(C) \Rightarrow V^{\pi_0}(C) = 10$$

$$V^{\pi_0}(W) = -10 + 0.9 \cdot V^{\pi_0}(O) \Rightarrow V^{\pi_0}(W) = -10$$

$$V^{\pi_0}(O) = 0$$

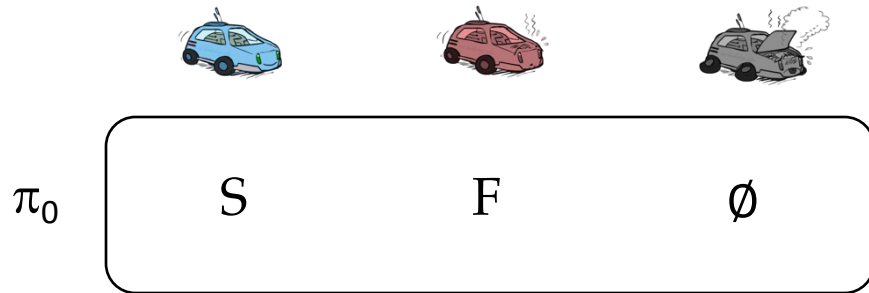
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



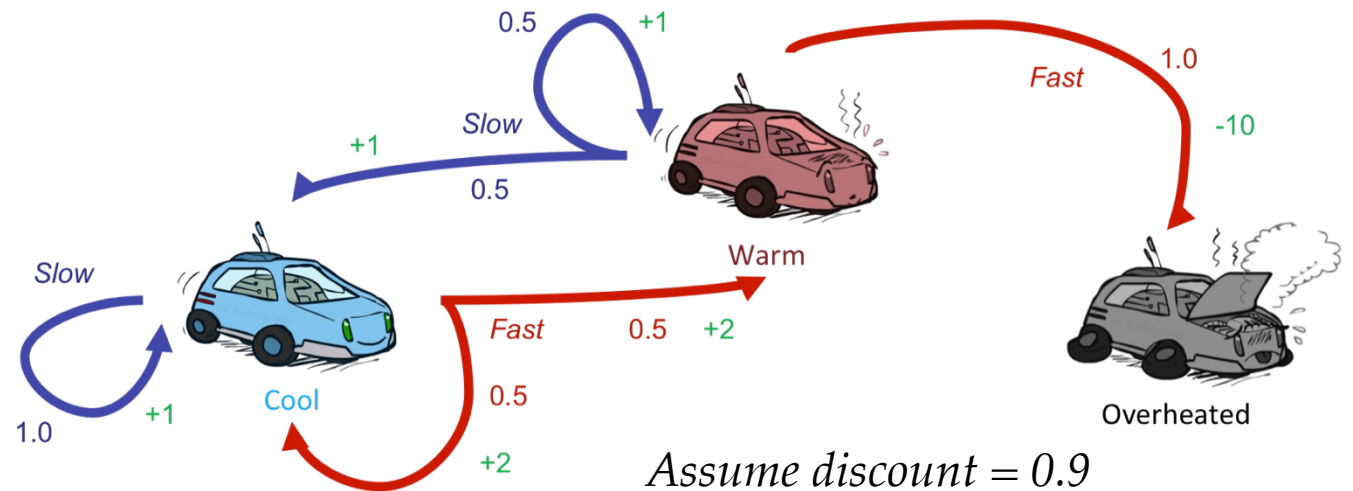
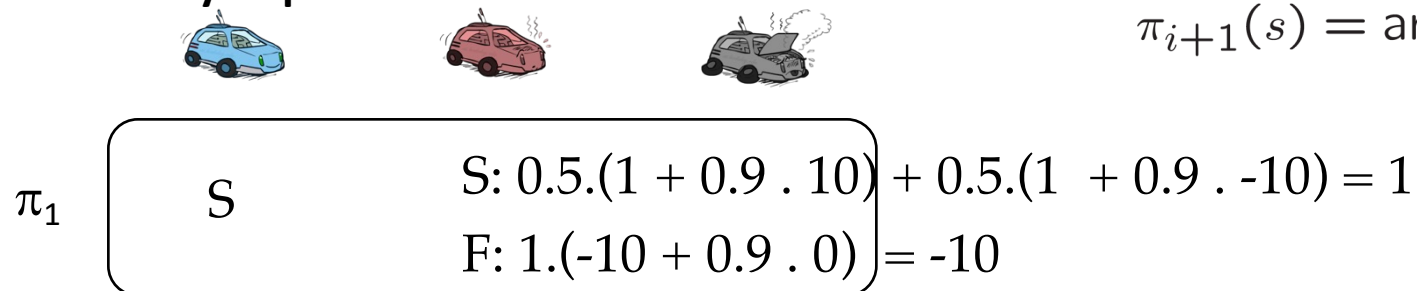
Policy Evaluation:

$$V^{\pi_0}(C) = 1 + 0.9 \cdot V^{\pi_0}(C) \Rightarrow V^{\pi_0}(C) = 10$$

$$V^{\pi_0}(W) = -10 + 0.9 \cdot V^{\pi_0}(O) \Rightarrow V^{\pi_0}(W) = -10$$

$$V^{\pi_0}(O) = 0$$

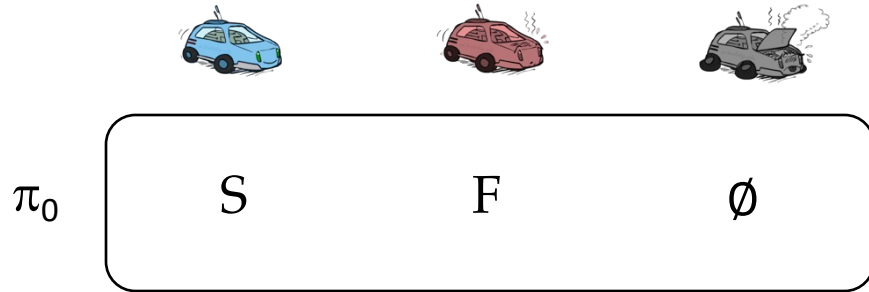
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



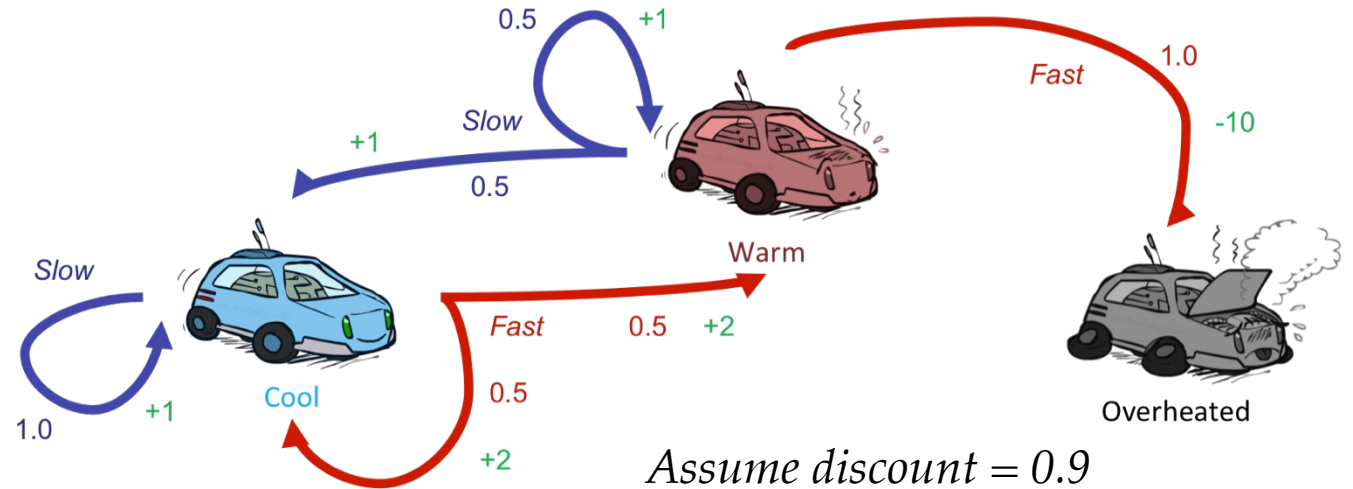
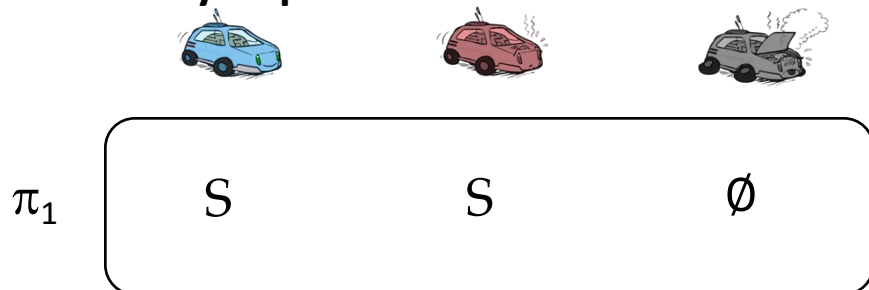
Policy Evaluation:

$$V^{\pi_0}(C) = 1 + 0.9 \cdot V^{\pi_0}(C) \Rightarrow V^{\pi_0}(C) = 10$$

$$V^{\pi_0}(W) = -10 + 0.9 \cdot V^{\pi_0}(O) \Rightarrow V^{\pi_0}(W) = -10$$

$$V^{\pi_0}(O) = 0$$

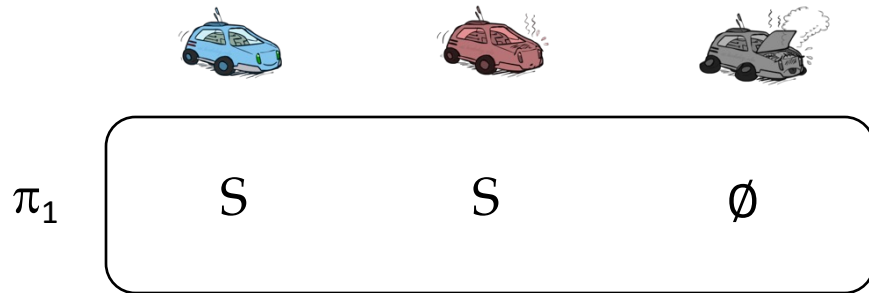
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



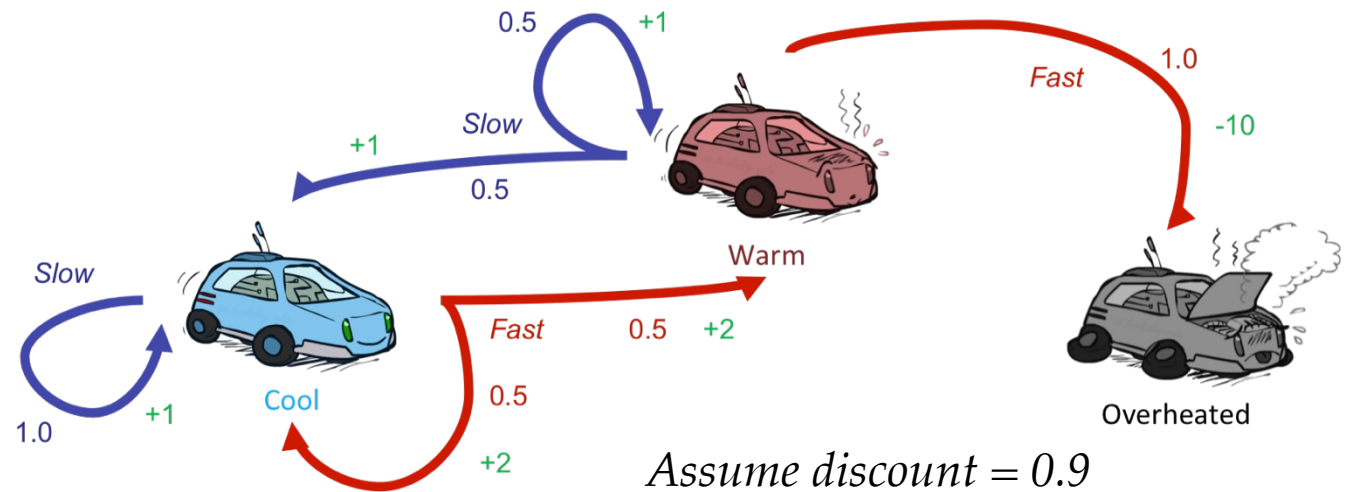
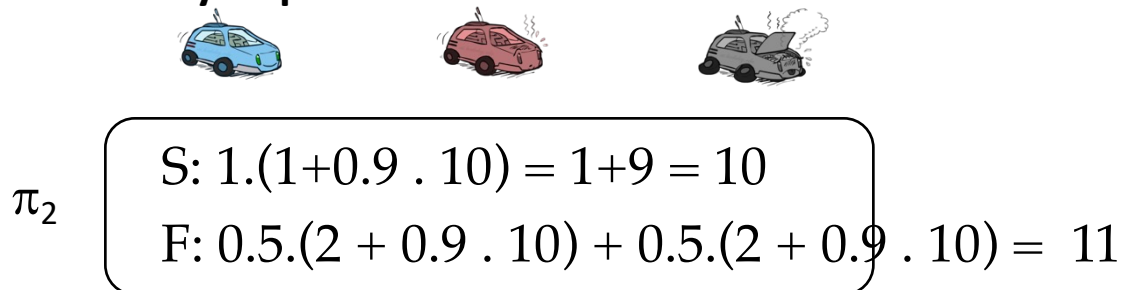
Policy Evaluation:

$$V^{\pi_1}(C) = 1 + 0.9 \cdot V^{\pi_1}(C) \Rightarrow V^{\pi_1}(C) = 10$$

$$V^{\pi_1}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_1}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_1}(W))$$

$$V^{\pi_1}(O) = 0 \Rightarrow V^{\pi_1}(W) = 10$$

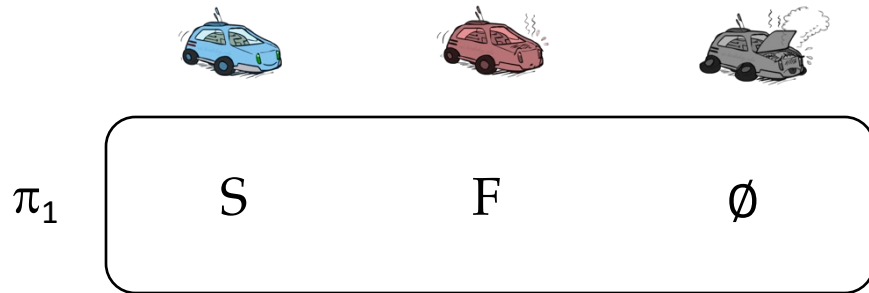
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



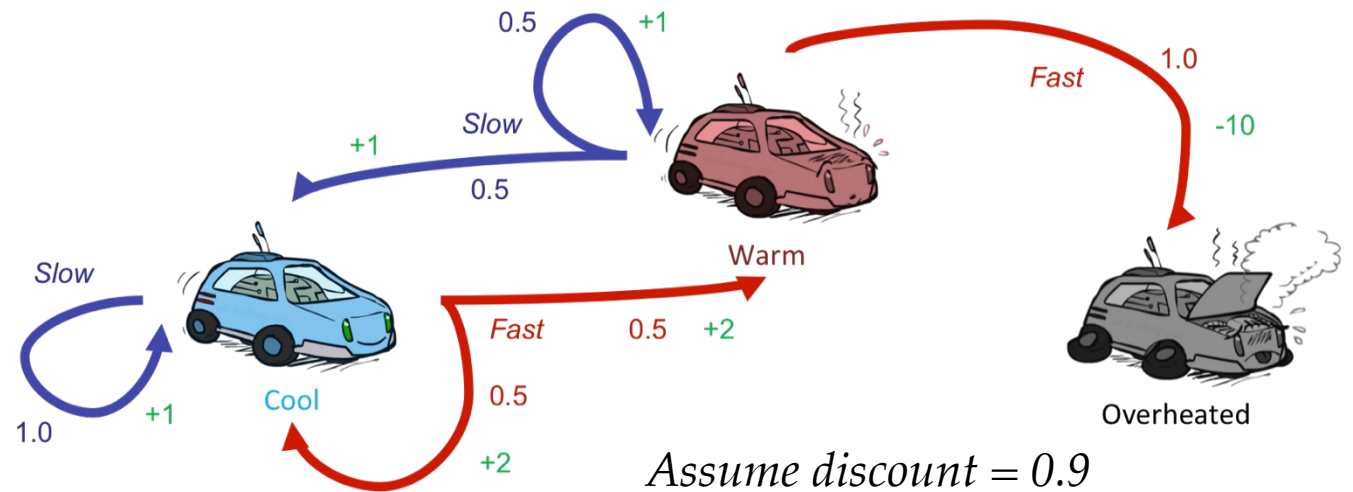
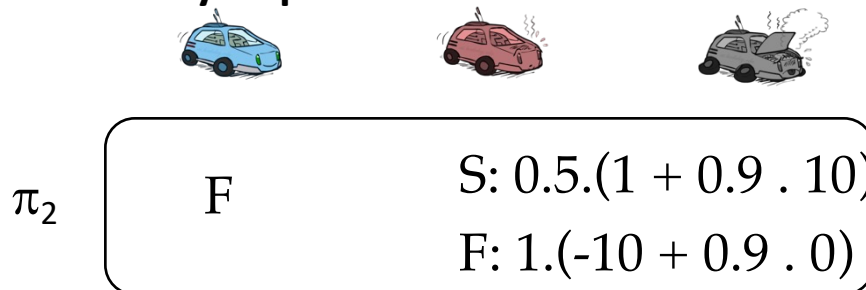
Policy Evaluation:

$$V^{\pi_1}(C) = 1 + 0.9 \cdot V^{\pi_1}(C) \Rightarrow V^{\pi_1}(C) = 10$$

$$V^{\pi_1}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_1}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_1}(W))$$

$$V^{\pi_1}(O) = 0 \quad \Rightarrow V^{\pi_1}(W) = 10$$

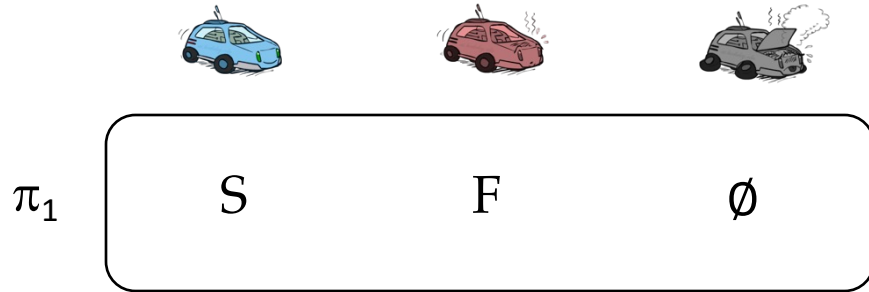
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



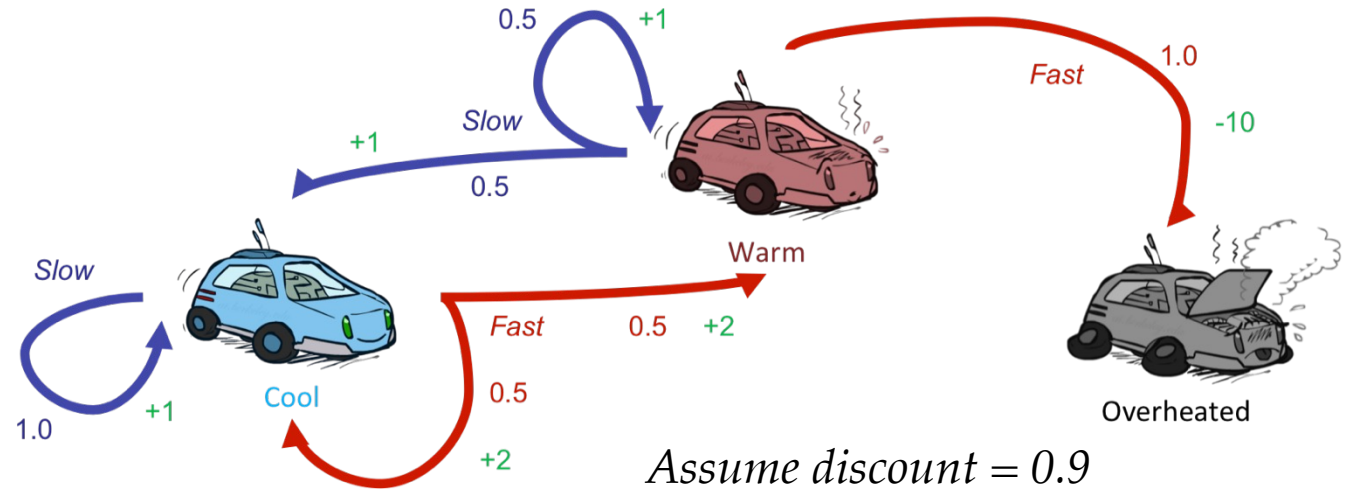
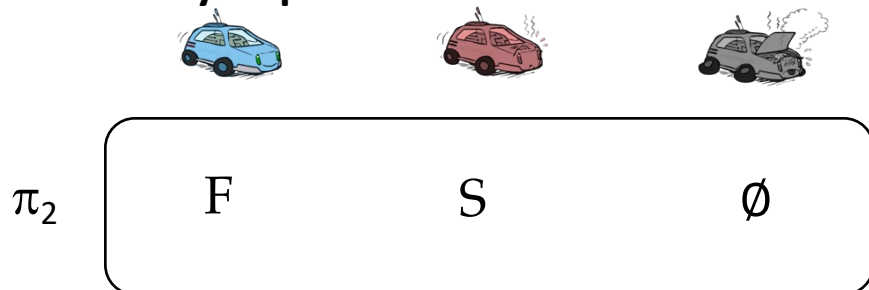
Policy Evaluation:

$$V^{\pi_1}(C) = 1 + 0.9 \cdot V^{\pi_1}(C) \Rightarrow V^{\pi_1}(C) = 10$$

$$V^{\pi_1}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_1}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_1}(W))$$

$$V^{\pi_1}(O) = 0 \Rightarrow V^{\pi_1}(W) = 10$$

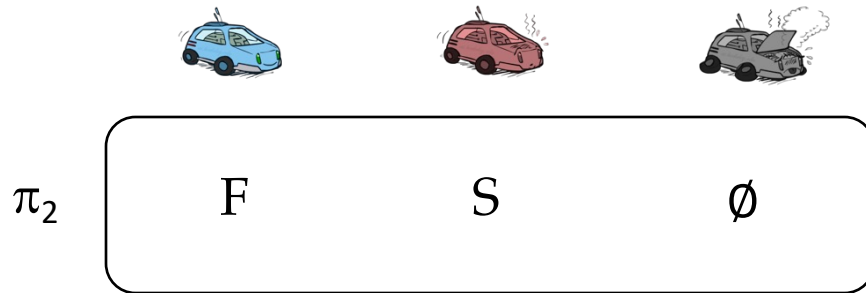
Policy Improvement:



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



Policy Evaluation:

$$\Rightarrow V^{\pi_2}(C) = 15.5$$

$$V^{\pi_2}(C) = 0.5 (2 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (2 + 0.9 \cdot V^{\pi_2}(W))$$

$$V^{\pi_2}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_2}(W))$$

$$\Rightarrow V^{\pi_2}(W) = 10$$

$$V^{\pi_2}(O) = 0$$

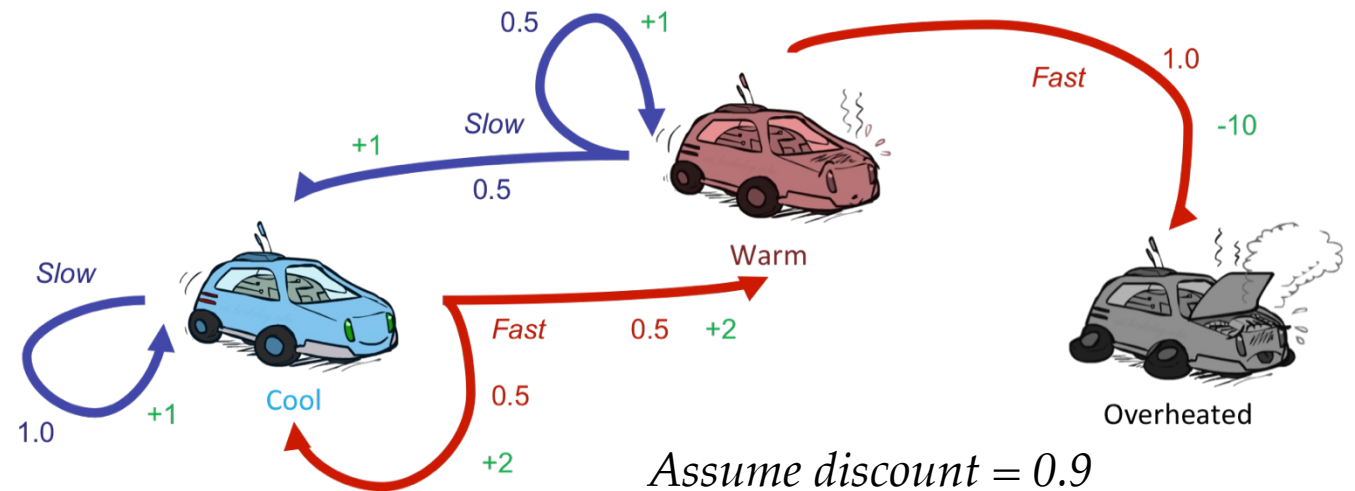
Policy Improvement:



π_3

$$S: 1 \cdot (1 + 0.9 \cdot 15.5) = 14.95$$

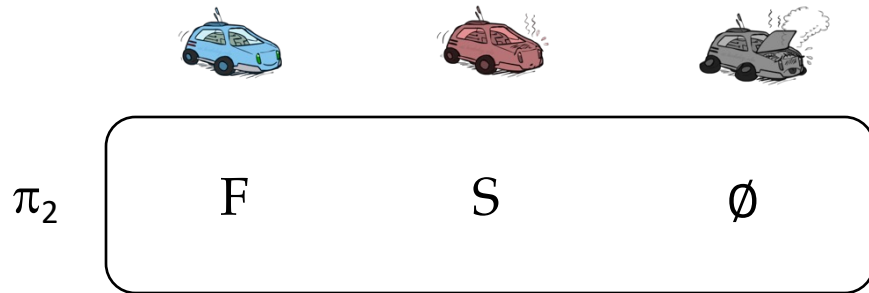
$$F: 0.5 \cdot (2 + 0.9 \cdot 15.5) + 0.5 \cdot (2 + 0.9 \cdot 14.5) = 15.5$$



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



Policy Evaluation:

$$\Rightarrow V^{\pi_2}(C) = 15.5$$

$$V^{\pi_2}(C) = 0.5 (2 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (2 + 0.9 \cdot V^{\pi_2}(W))$$

$$V^{\pi_2}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_2}(W))$$

$$\Rightarrow V^{\pi_2}(W) = 10$$

$$V^{\pi_2}(O) = 0$$

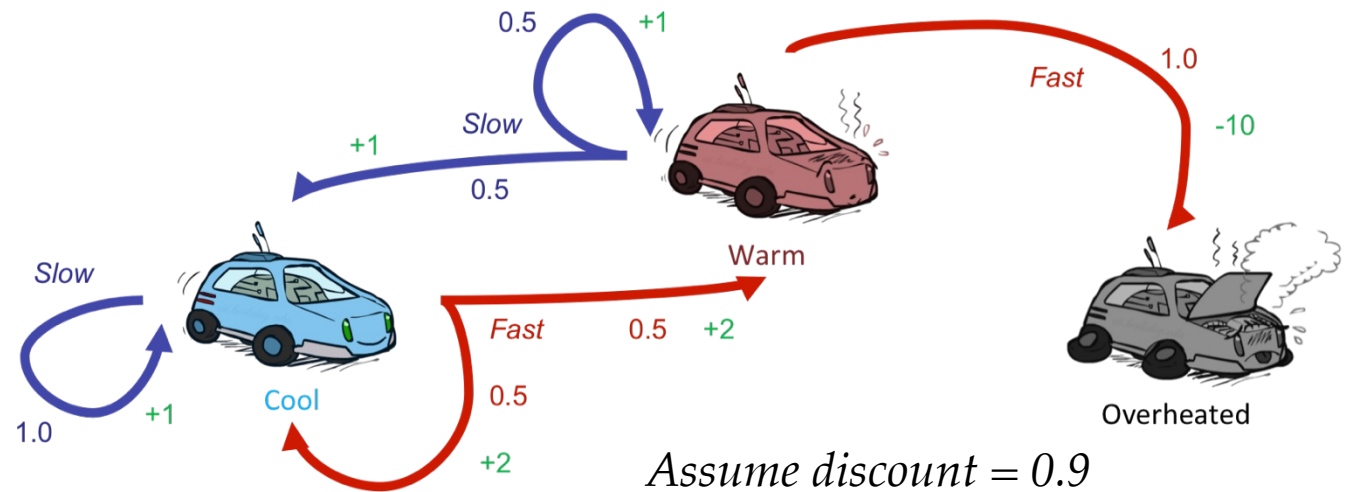
Policy Improvement:



π_3

F

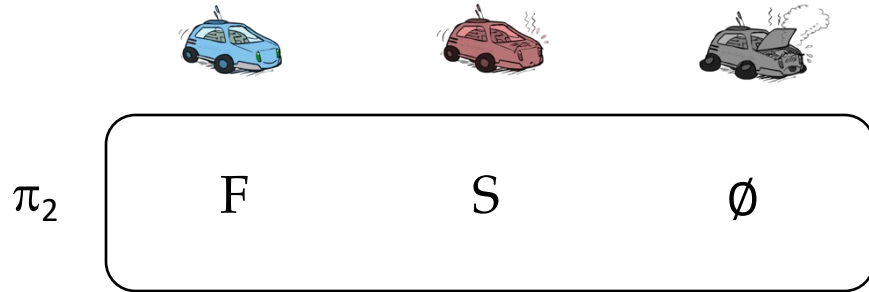
$$\begin{aligned} S: & 0.5.(1 + 0.9 \cdot 15.5) + 0.5.(1 + 0.9 \cdot 10) = 14.5 \\ F: & 1.(-10 + 0.9 \cdot 0) = -10 \end{aligned}$$



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

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

مثال : Policy Iteration



Policy Evaluation:

$$\Rightarrow V^{\pi_2}(C) = 15.5$$

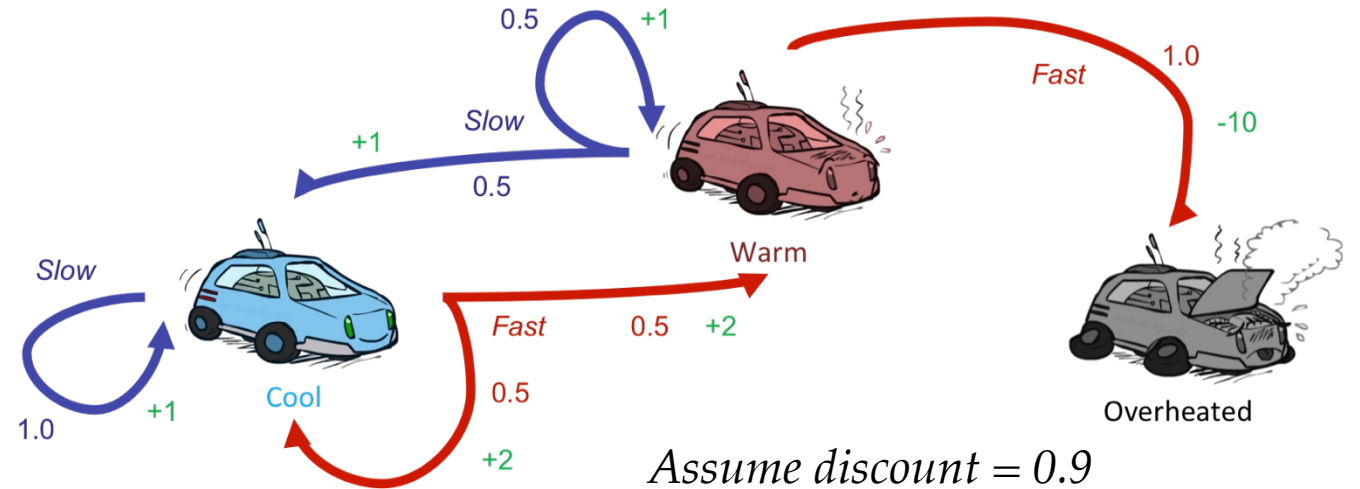
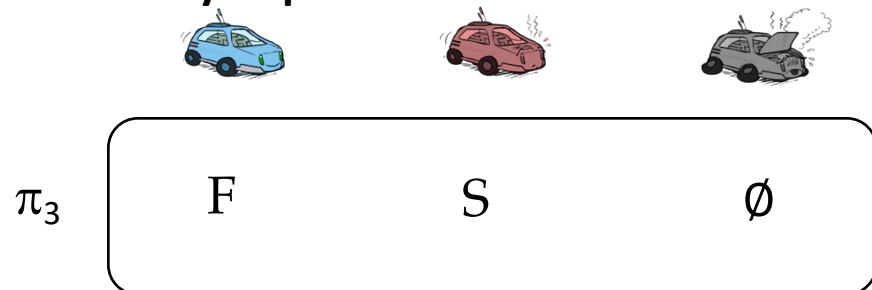
$$V^{\pi_2}(C) = 0.5 (2 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (2 + 0.9 \cdot V^{\pi_2}(W))$$

$$V^{\pi_2}(W) = 0.5 (1 + 0.9 \cdot V^{\pi_2}(C)) + 0.5 (1 + 0.9 \cdot V^{\pi_2}(W))$$

$$\Rightarrow V^{\pi_2}(W) = 10$$




$$V^{\pi_2}(O) = 0$$

Policy Improvement:

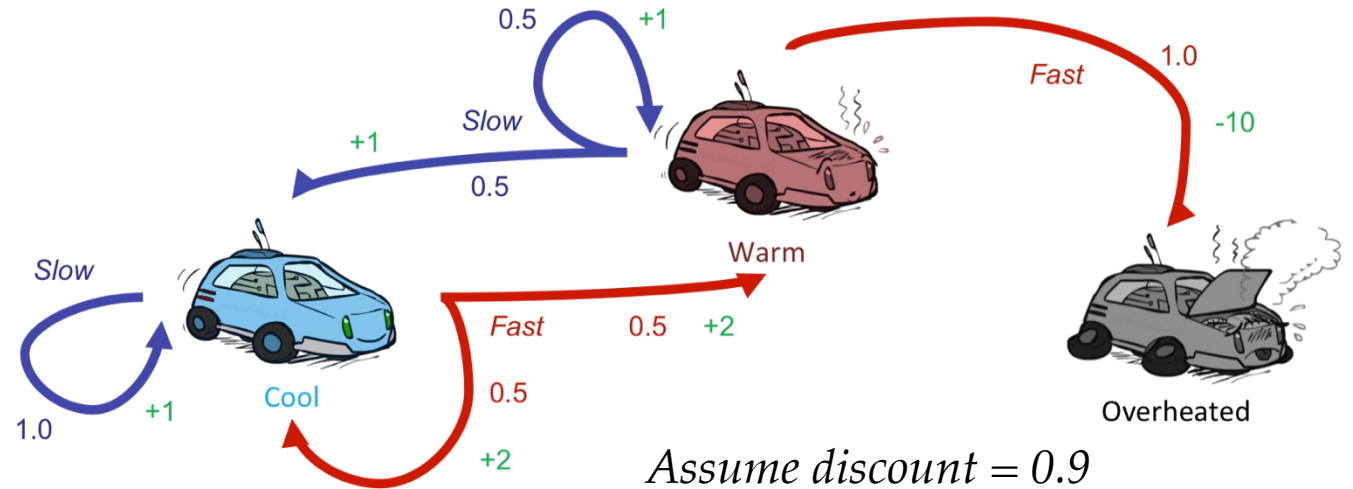


$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

$$\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

			
V_0	0	0	0
V_1	2	1	0
V_2	3.35	2.35	0
	⋮		
V_{91}	15.499	14.499	0

مثال : Value Iteration



$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

مقایسه

□ الگوریتم های تکرار مقدار و تکرار سیاست هر دو یک کمیت را محاسبه می کنند: (همه مقادیر بهینه)

□ در الگوریتم تکرار مقدار:

- در هر تکرار، مقادیر و سیاست هر دو به روز رسانی می شوند.
- ما به دنبال سیاست نیستیم اما با محاسبه ی ماکزیمم بر روی عملیات، به طور ضمنی آن را محاسبه می کنیم.

□ در الگوریتم تکرار سیاست:

- چند گذر وجود دارد که در آن ها مقادیر سودمندی را تنها برای یک سیاست ثابت به روز رسانی می کنیم.
- پس از ارزیابی سیاست، یک سیاست جدید انتخاب می شود. (استخراج سیاست)
- سیاست جدید نسبت به سیاست قبلی بهتر است. [در غیر این صورت الگوریتم همگرا شده است]

□ هر دو، رویکردهای برنامه ریزی پویا برای حل MDP ها هستند.

خلاصه: الگوریتم‌های MDP

□ به طور خلاصه، اگر میخواهید:

□ مقادیر بهینه‌ی حالت‌ها را محاسبه کنید: از الگوریتم تکرار مقدار یا تکرار سیاست استفاده کنید.

□ مقادیر را برای یک سیاست خاص محاسبه کنید: از الگوریتم ارزیابی سیاست استفاده کنید.

□ از روی مقادیر، سیاست را به دست آورید: از الگوریتم استخراج سیاست استفاده کنید. (نگاه روبه جلو)

□ این الگوریتم‌ها یکسان به نظر می‌رسند!

- همه‌ی این الگوریتم‌ها گونه‌های مختلف از به روز رسانی معادلات بلمن هستند.
- تفاوت آنها در این است که آیا از یک سیاست ثابت استفاده می‌کنیم یا در هر حالت بر روی عملیات ممکن ماکزیمم‌گیری می‌کنیم.