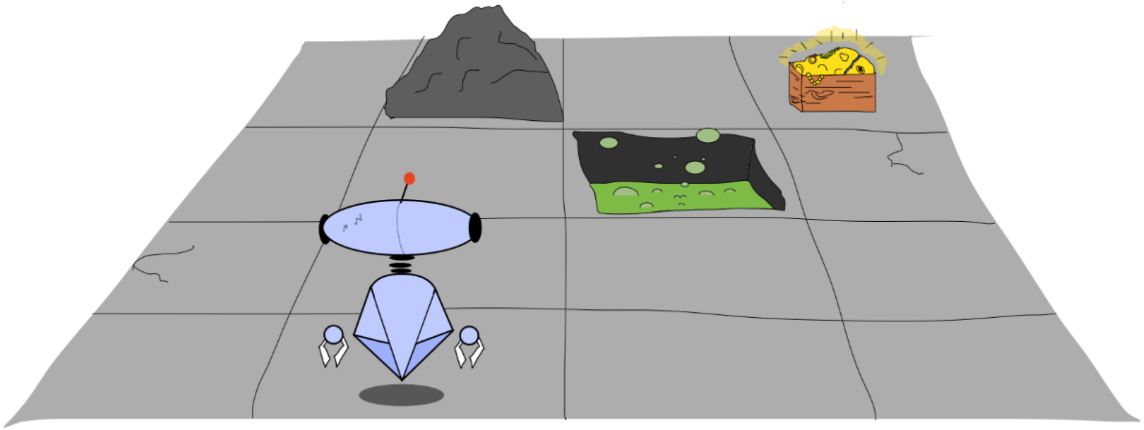
مبانی و کاربردهای هوش مصنوعی

فرایند تصمیم‌گیری مارکوف - 1 (فصل 17.1 الی 17.3)

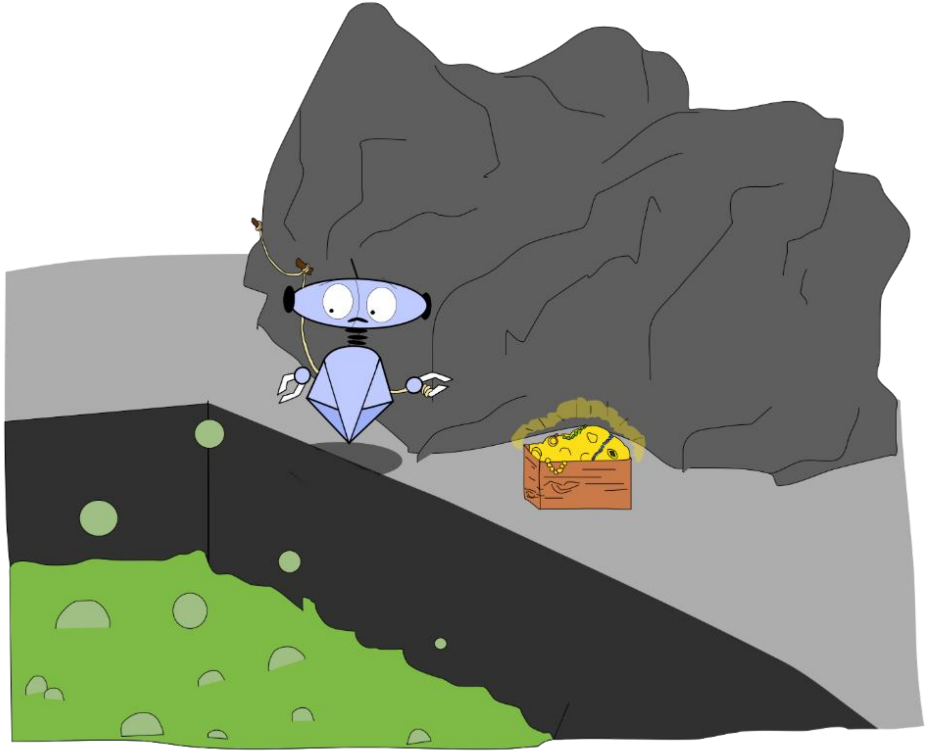


مدرس: مهدی جوانمردی

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر

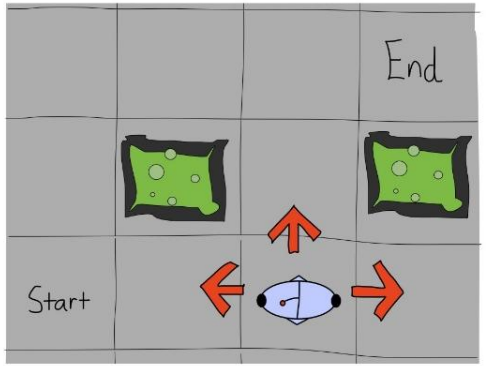
(الهام گرفته از محتوای درس هوش مصنوعی دانشگاه برکلی)

جستجوی غیر قطعی



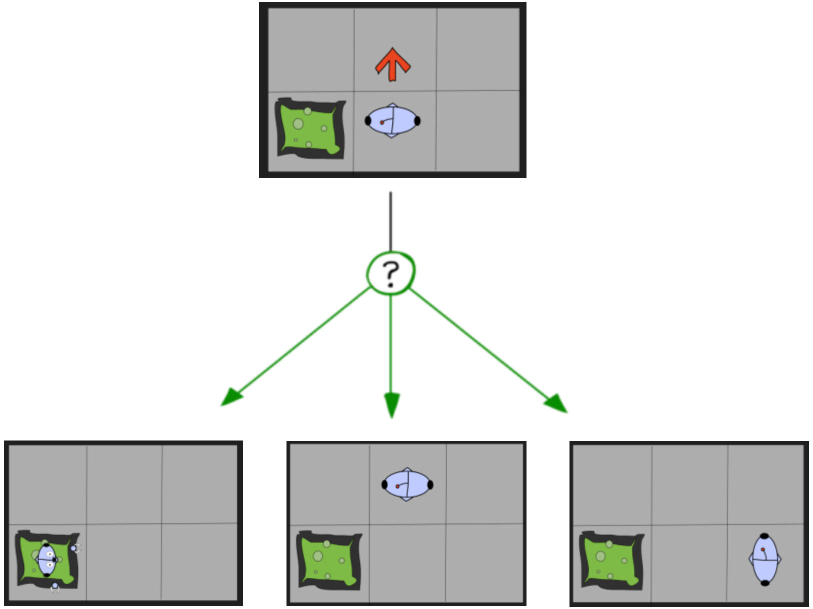
مثال: جهان مشبک (Grid World)

- یک مسئله هزارتو مانند
  - عامل در یک محیط مشبک زندگی می کند
  - دیوارها مسیر حرکت عامل را سد می کنند
- حرکت نویزدار: حرکات همیشه طبق برنامه‌ریزی پیش نمی روند
  - 80% مواقع ، حرکت شمال عامل را به شمال می برد (اگر دیوار نباشد)
  - 10% مواقع، شمال عامل را به غرب می برد، 10% شرق
  - اگر در جهتی که عامل انتخاب کرده بود دیوار باشد، عامل سرجایش می ماند
- عامل هر گام زمانی جایزه می گیرد
  - جایزه کوچک "زنده بودن" در هر گام زمانی (می تواند منفی باشد)
  - جایزه بزرگ در آخر است (خوب یا بد)
- هدف: بیشینه کردن مجموع جوایز

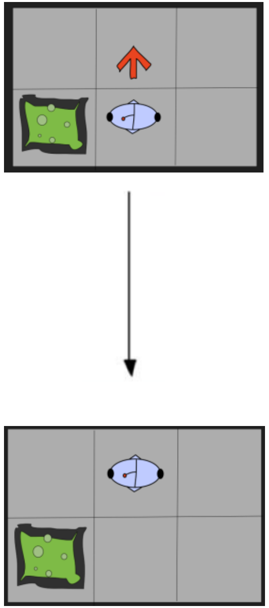


حرکت‌های جهان مشبک

جهان مشبک تصادفی



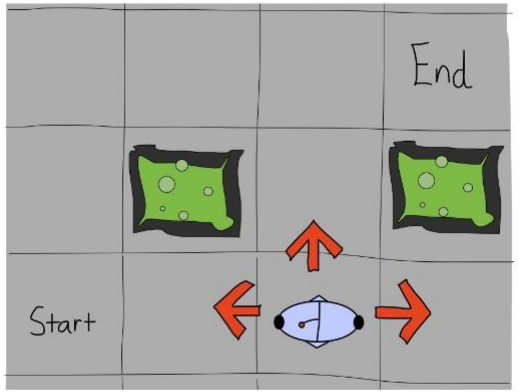
جهان مشبک قطعی



فرآیندهای تصمیم گیری مارکوف

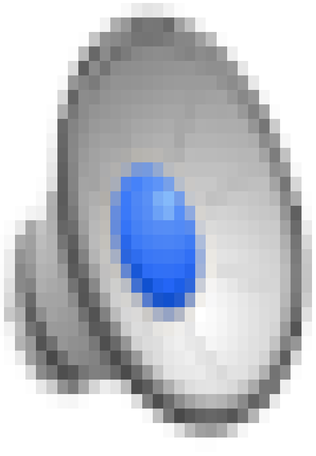
- یک MDP به این صورت تعریف می شود:
  - یک مجموعه از حالت‌ها $s \in S$
  - یک مجموعه از اعمال $a \in A$
  - یک تابع انتقال حالت T(s, a, s’)
    - احتمال اینکه a از s منجر به s’ شود که یعنی P(s’| s, a)
    - به آن مدل یا دینامیک هم گفته می شود
  - یک تابع پاداش R(s, a, s’)
    - بعضی اوقات فقط R(s) یا R(s’)
  - یک حالت شروع
  - شاید یک حالت پایانی

- MDPها مسائل جستجوی غیر قطعی هستند
  - یک راه برای حل آن‌ها جستجوی Expectimax است.
  - ما به زودی یک ابزار جدید خواهیم داشت.



[Demo – gridworld manual intro (L8D1)]

Gridworld Manual Intro ویدیوی نمایشی



در MDPها چه چیزی مارکوف است؟

- به صورت کلی "مارکوف" به این معناست که با داشتن حالت فعلی، گذشته و آینده نسبت به هم مستقل اند
- در فرآیند تصمیم گیری مارکوف، "مارکوف" به این معناست که خروجی تنها به حالت فعلی وابسته است

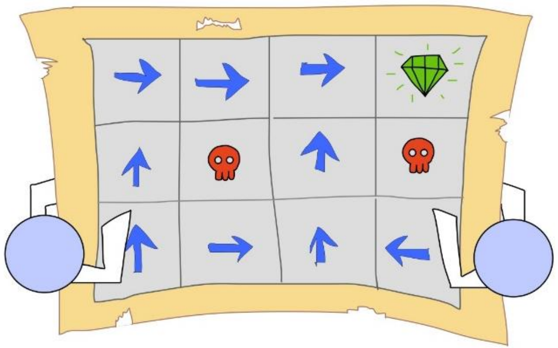
Andrey Markov
(1856-1922)

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$
$$=$$
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- این دقیقا مشابه مسائل جستجو است که تابع پسین (successor function) فقط می تواند به حالت فعلی وابسته باشد (نه تاریخچه)

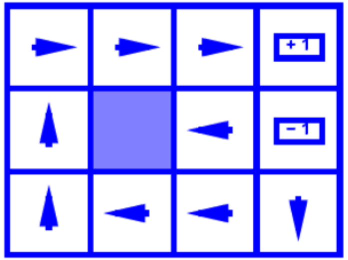
سیاست‌ها

- در مسائل جستجوی تک عاملی قطعی، ما به دنبال یک برنامه بهینه، یا دنباله‌ای از اعمال، بودیم
- برای MDPها، دنبال یک سیاست بهینه هستیم $\pi^*: S \rightarrow A$
  - یک سیاست π برای هر حالت یک عمل می دهد
  - یک سیاست بهینه، سیاستی است که در اگر دنبال شود سودمندی مورد انتظار را بیشینه کند
  - یک سیاست صریح، عامل ما را تبدیل به یک عامل واکنشی می‌کند
- جستجوی expectimax سیاست‌ها را به طور کامل محاسبه نمی کرد
  - تنها برای یک حالت عمل را محاسبه می کرد

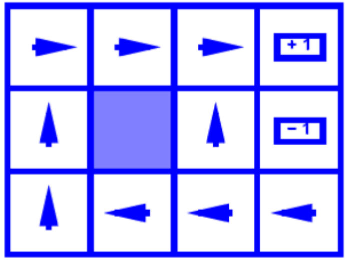


سیاسیت بهینه وقتی که R(s, a, s') = -0.03 برای حالات غیر پایانی s

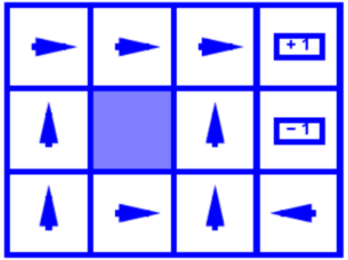
سیاست‌های بهینه



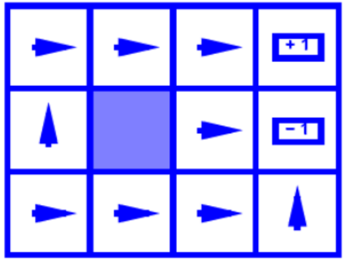
R(s) = -0.01



R(s) = -0.03



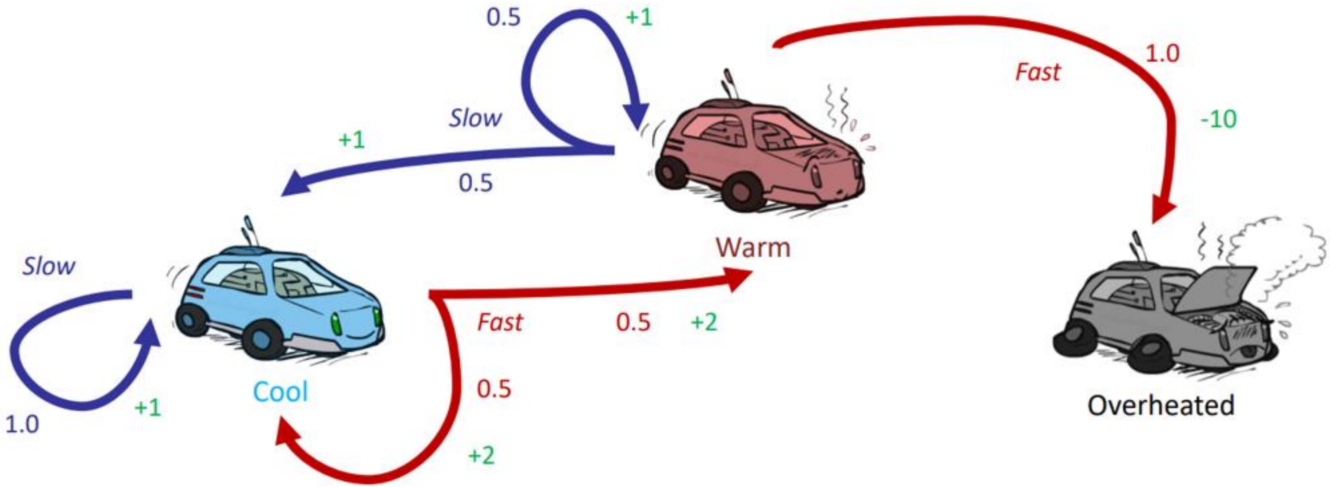
R(s) = -0.4



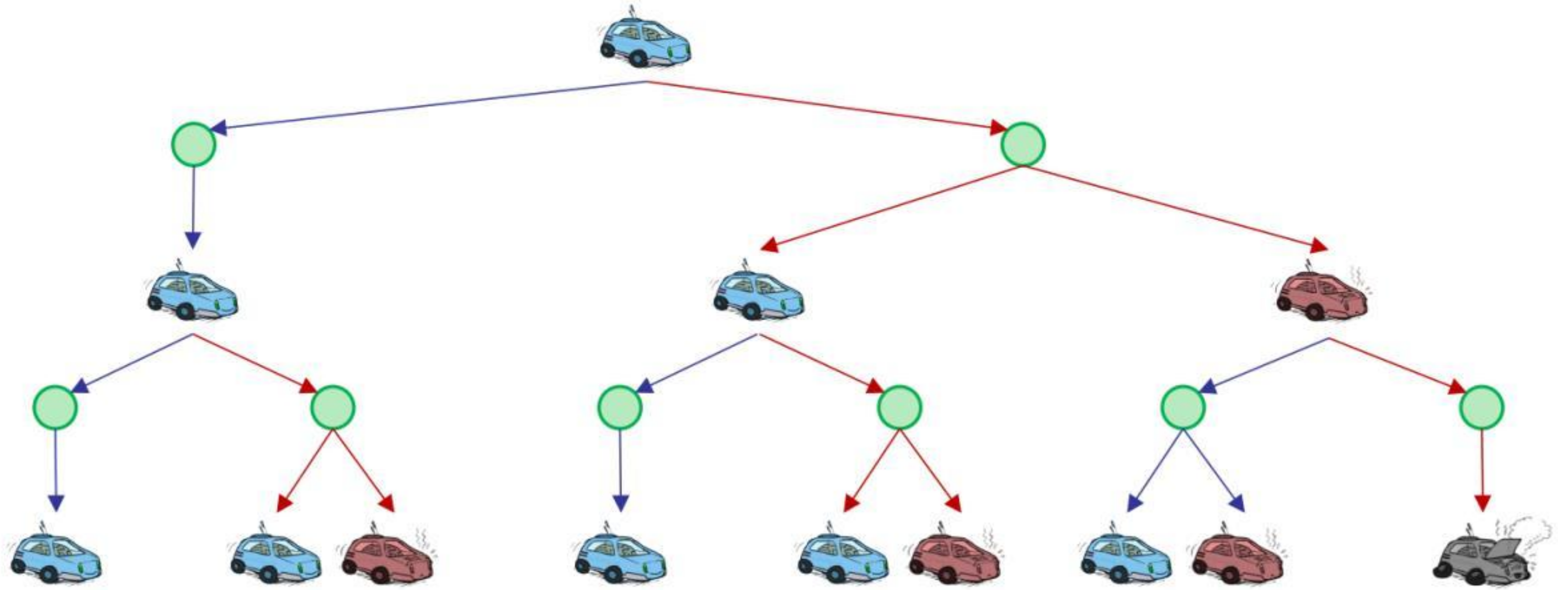
R(s) = -2.0

مثال: مسابقه

- یک اتومبیل روباتی می خواهد یک مسیر طولانی را به سرعت بپیماید
- سه حالت Cool, Warm, Overheated
- دو عمل *Slow*, *Fast*
- سریعتر رفتن پاداش دو برابر دارد

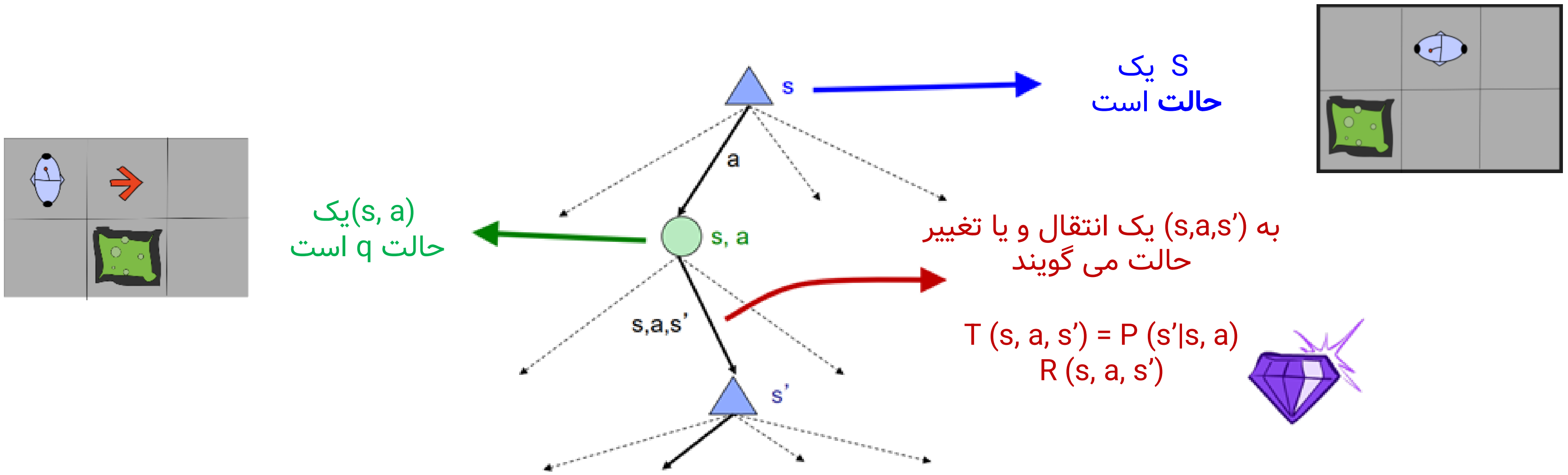


درخت جستجوی مسابقه

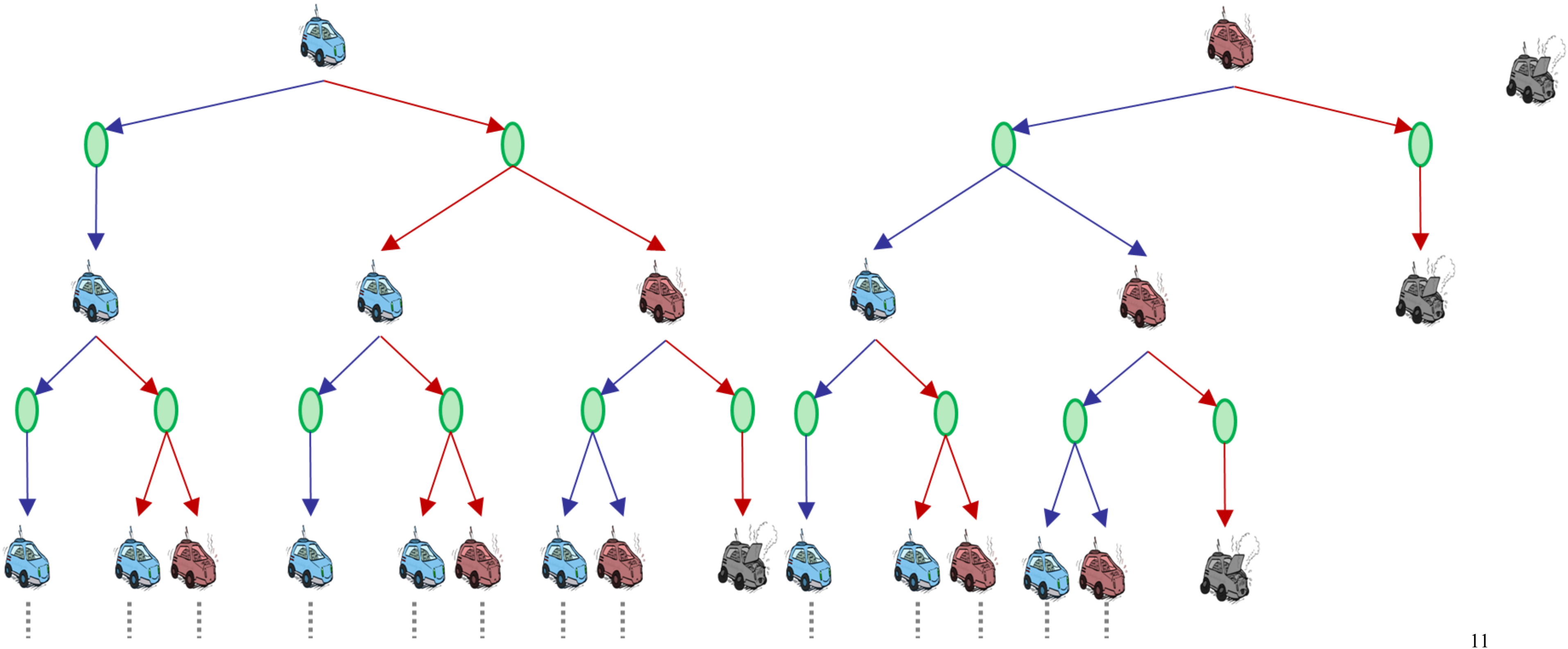


درخت جستجوی MDP

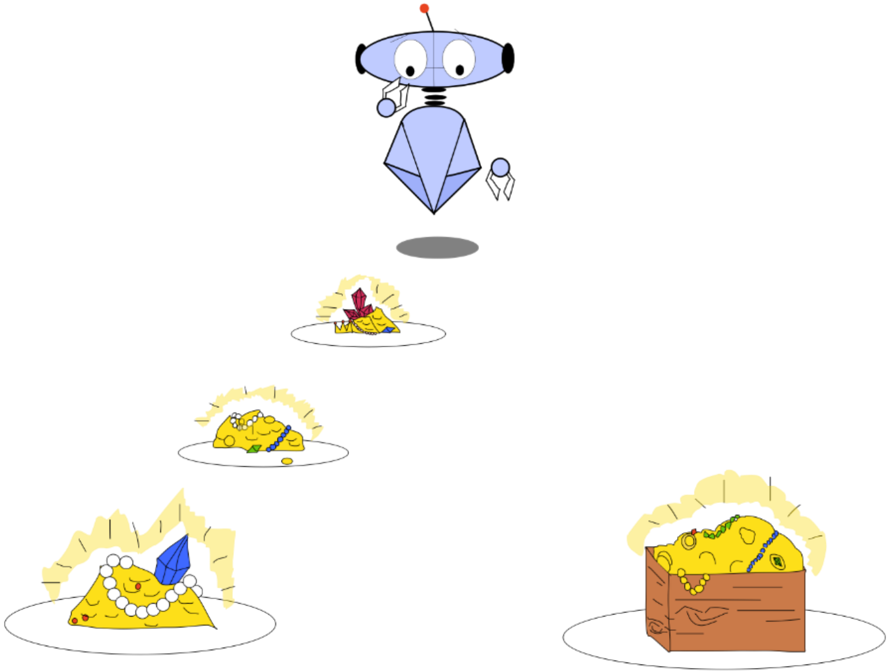
- هر حالت در MDP یک درخت جستجو مشابه Expectimax را نمایش می دهد



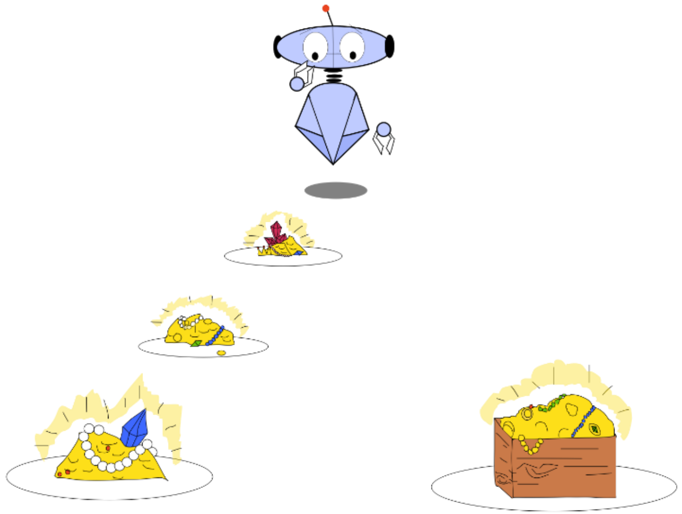
درخت جستجوی مسابقه



سودمندی‌های دنباله‌ها



سودمندی‌های دنباله‌ها



- عامل باید کدام دنباله از پاداش‌ها را ترجیح دهد؟
- کمتر یا بیشتر؟ [1, 2, 2] یا [2, 3, 4]
- حالا یا بعدا؟ [0, 0, 1] یا [1, 0, 0]

تخفیف دادن

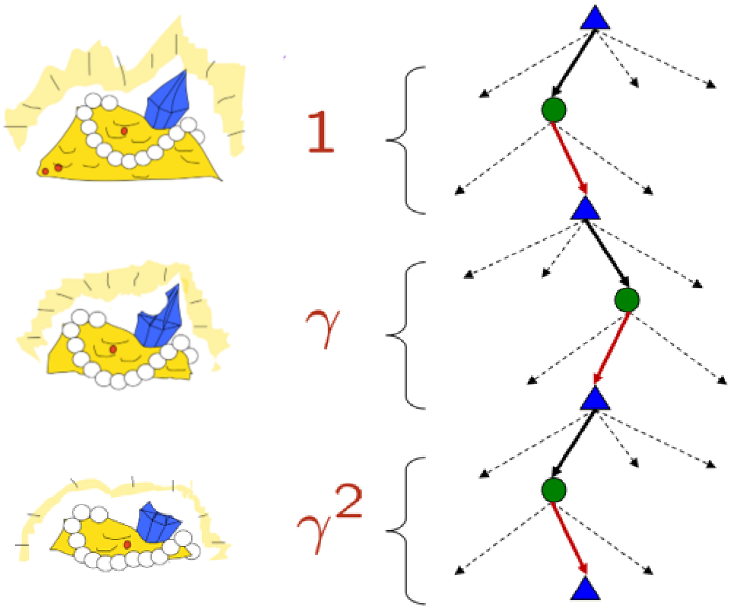
- منطقی است که مجموع پاداش‌ها بیشینه شود
- همچنین منطقی است که پاداش‌های الان به پاداش‌های آینده ترجیح داده شوند
- یک راه حل: مقدار پاداش‌ها نمایی تخفیف یابد

1 — ارزش فعلی

γ — ارزش در گام بعدی

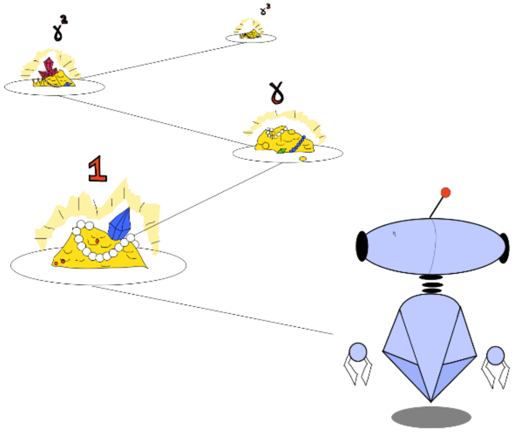
γ^2 — ارزش در دو گام بعد

تخفیف دادن



- چگونه تخفیف بدهیم؟
  - هر بار که یک سطح پایین می رویم، ضریب تخفیف را یک بار ضرب می کنیم
- چرا باید تخفیف دهیم؟
  - پاداش‌های فوری احتمالا سودمندی بالاتری نسبت به پاداش‌های دارای تأخیر دارند
  - همچنین کمک می کند الگوریتم‌ها همگرا شوند
- مثال: تخفیف 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3
  - U([1,2,3]) < U([3,2,1])

اولویت ایستا



- قضیه: برای اولویت‌های ایستا:

$$[a_1, a_2, \ldots] \succ [b_1, b_2, \ldots]$$
$$\Updownarrow$$
$$[r, a_1, a_2, \ldots] \succ [r, b_1, b_2, \ldots]$$

- آنگاه اگر سودمندی‌ها بصورت زیر تعریف شود ویژگی ایستایی اولویت حفظ می‌شود
  - سودمندی جمع شونده: $$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$$
  - سودمندی تخفیفی: $$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

آزمونک: تخفیف دادن

- با فرض:

a	b	c	d	e
10				1

- اعمال: شرق ، غرب و خروج (فقط از طریق حالت‌های خروج a و e)
- انتقال: قطعی

- آزمونک 1: برای ضریب $\gamma = 1$ سیاست بهینه چیست؟

10				1

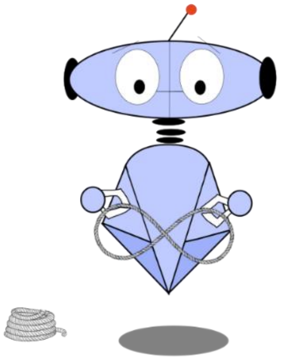
- آزمونک 2: برای ضریب $\gamma = 0.1$ سیاست بهینه چیست؟

10				1

- آزمونک 3: برای چه γ وقتی در d هستیم غرب و شرق به یک اندازه خوب هستند؟

سودمندی‌های بینهایت؟!

- مشکل: اگر بازی تا ابد ادامه داشته باشد چه می شود؟ آیا پاداش بینهایت می گیریم؟
- راه حل‌ها:
  - افق متناهی: (مشابه جستجو با عمق محدود)
    - پایان بخشیدن به بازی بعد از T گام ثابت (مثل زندگی)
    - باعث سیاست‌های غیر ایستا می شود (π بستگی به زمان باقیمانده دارد)
  - تخفیف: استفاده از $0 < \gamma < 1$



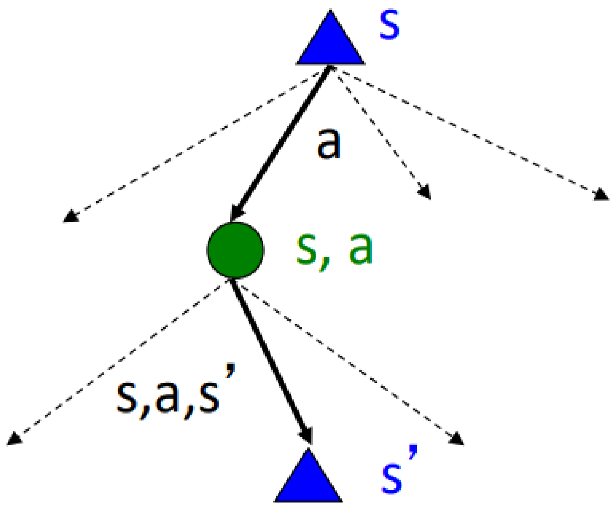
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R\text{max}/(1-\gamma)$$

    - γ کوچکتر یعنی افق کوچکتر - تمرکز کوتاه مدت‌تر
  - حالت‌های جذب کننده: ضمانت می کند که برای هر سیاست، در نهایت یک حالت پایانی خواهد رسید (مثال "حالت جوش آمدن" برای ماشین مسابقه)

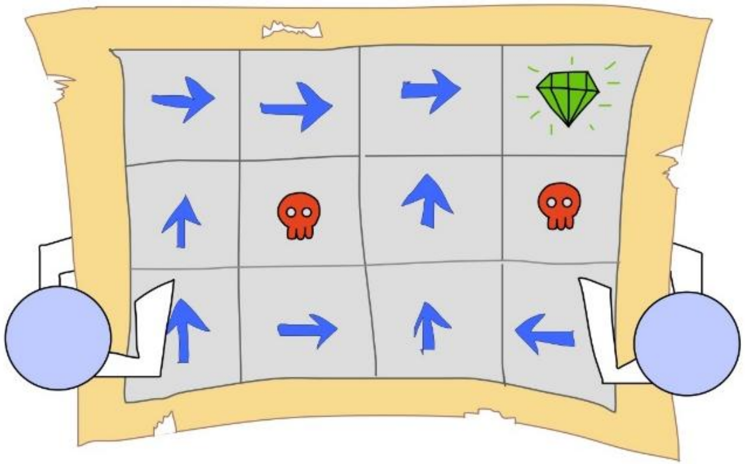
یادآوری: تعریف MDPها

- فرآیندهای تصمیم گیری مارکوف:
  - مجموعه‌ای از حالت‌ها **S**
  - حالت شروع s_0
  - مجموعه از اعمال **A**
  - تابع انتقال T(s, a, s’) (یا P(s’| s, a))
- تابع پاداش R(s, a, s’) (و ضریب تخفیف γ)

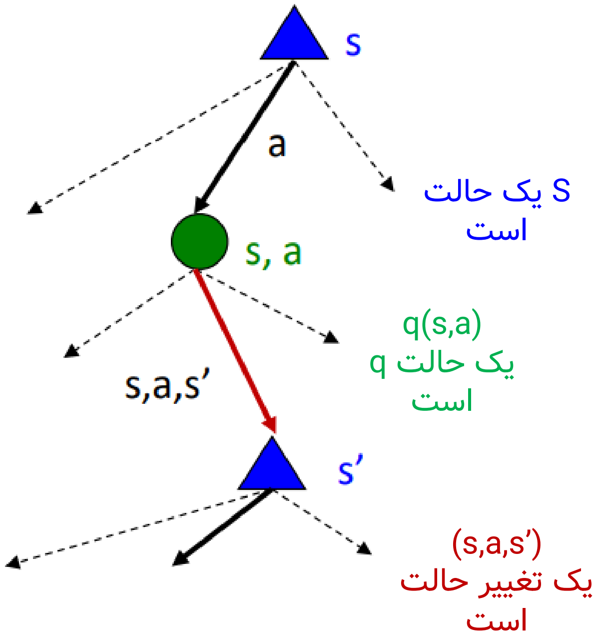
- کمیت‌های مهم تا اینجا:
  - سیاست: انتخاب عمل برای هر حالت
  - سودمندی: مجموع (تخفیف یافته) پاداش‌ها



حل MDPها



کمیت‌های بهینه



[Demo – gridworld values (L8D4)]

- مقدار (سودمندی) یک حالت s :

  V*(s) = سودمندی مورد انتظار با شروع از s و بهینه عمل کردن

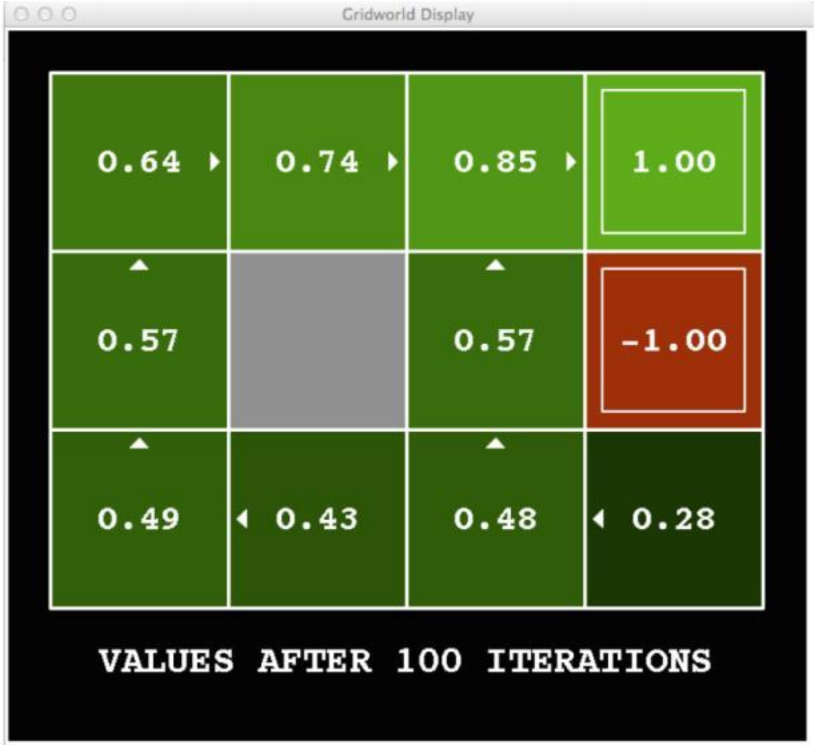
- مقدار (سودمندی) یک حالت q (s,a):

  Q*(s,a) = سودمندی مورد انتظاربا شروع از s و انتخاب a و (از آنجا به بعد) بهینه عمل کردن

- سیاست بهینه:

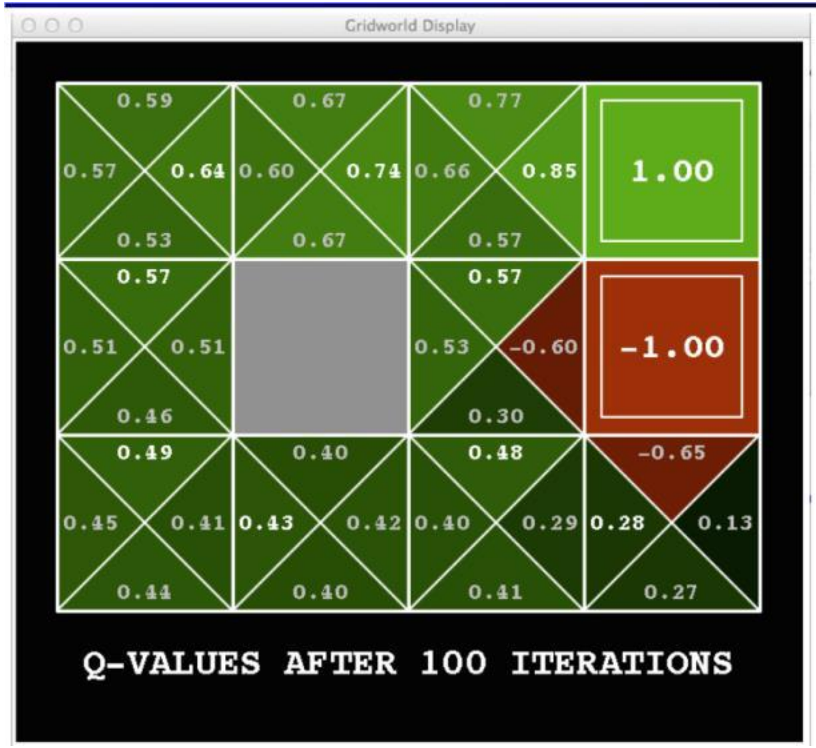
  π *(s) = حرکت بهینه از حالت s

Snapshot of Demo – Gridworld V Values



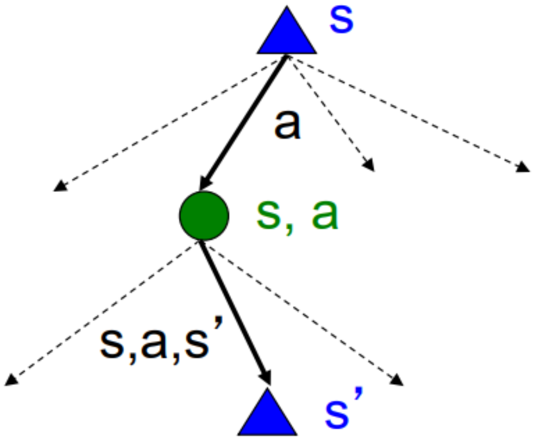
Noise = 0.2
Discount = 0.9
Living reward = 0

Snapshot of Demo – Gridworld Q Values



Noise = 0.2
Discount = 0.9
Living reward = 0

مقدار حالت‌ها



- عملیات پایه‌ای: محاسبه مقدار بهینه یک حالت $V^*(s)$
  - سودمندی مورد انتظار با حرکت بهینه
  - مجموع (تخفیف یافته) پاداش‌ها
  - این همان چیزیست که expectimax محاسبه می‌کند!

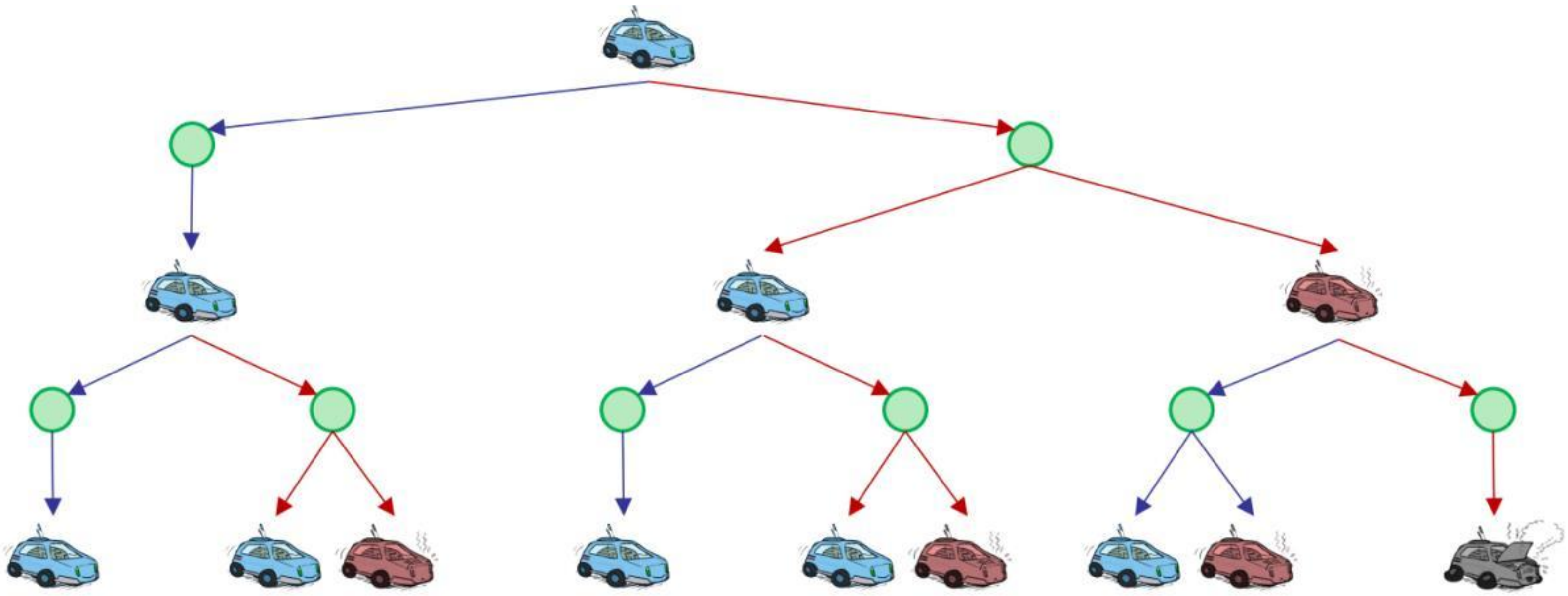
- تعریف بازگشتی مقدار

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

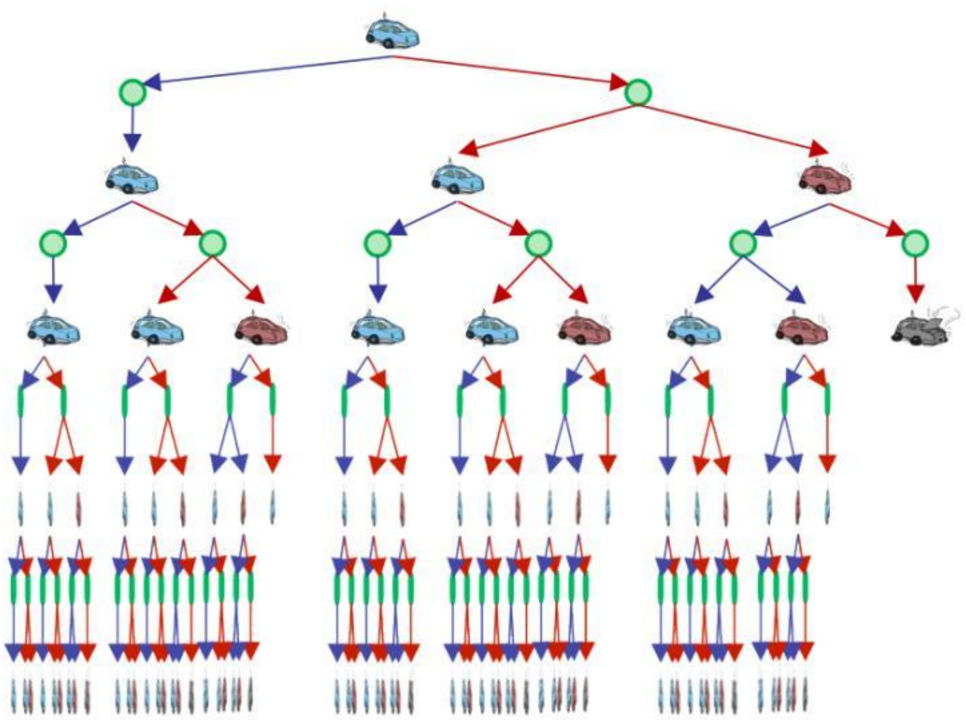
$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

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

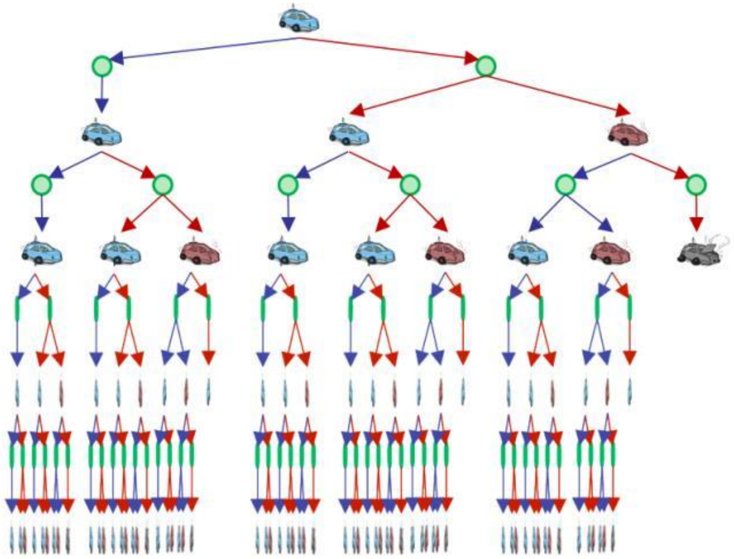
درخت جستجوی مسابقه



درخت جستجوی مسابقه



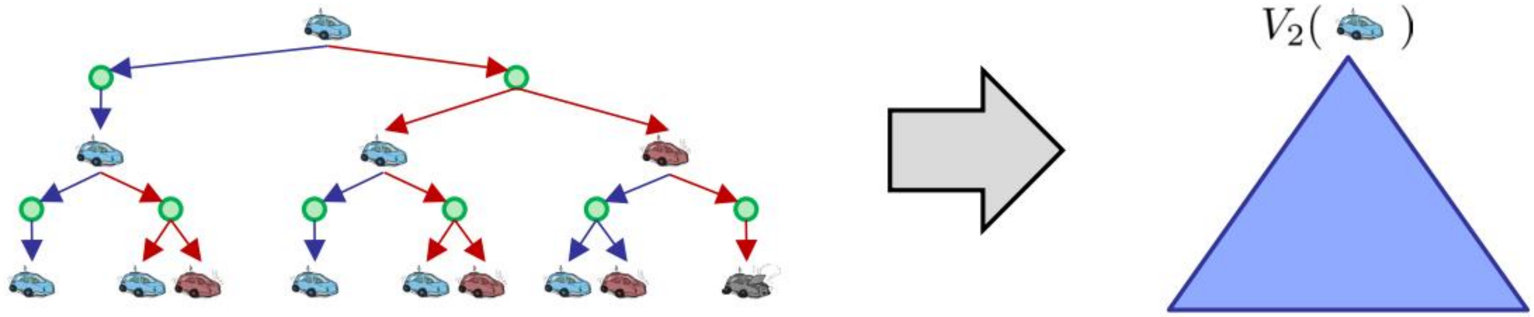
درخت جستجوی مسابقه



- با استفاده از expectimax ما بیش از اندازه کار می کنیم!
- مشکل: حالت‌ها تکرار میشوند
  - ایده : مقادیر مورد نیاز را فقط یکبار محاسبه کن
  - راه حل: برنامه نویسی پویا
- مشکل: درخت تا بینهایت ادامه دارد
  - ایده : از محاسبات با عمق محدود استفاده کن و آنقدر ادامه بده تا تغییرات کوچک شود
  - نکته : اگر $1 > \gamma$ قسمت‌های عمیق درخت در نهایت اهمیتی ندارند

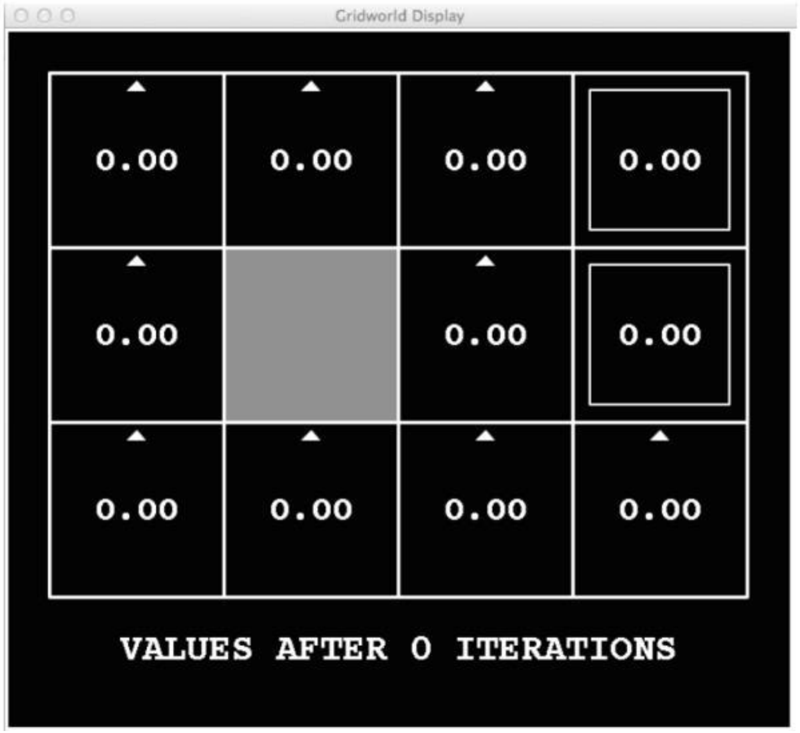
مقدار زمان محدود

- ایده کلیدی: مقدار زمان محدود (time-limited values)
- تعریف $V_k(s)$ به عنوان مقدار بهینه s اگر بازی پس از k مرحله‌ی دیگر خاتمه یابد
  - معادل اجرای expectimax با عمق محدود k



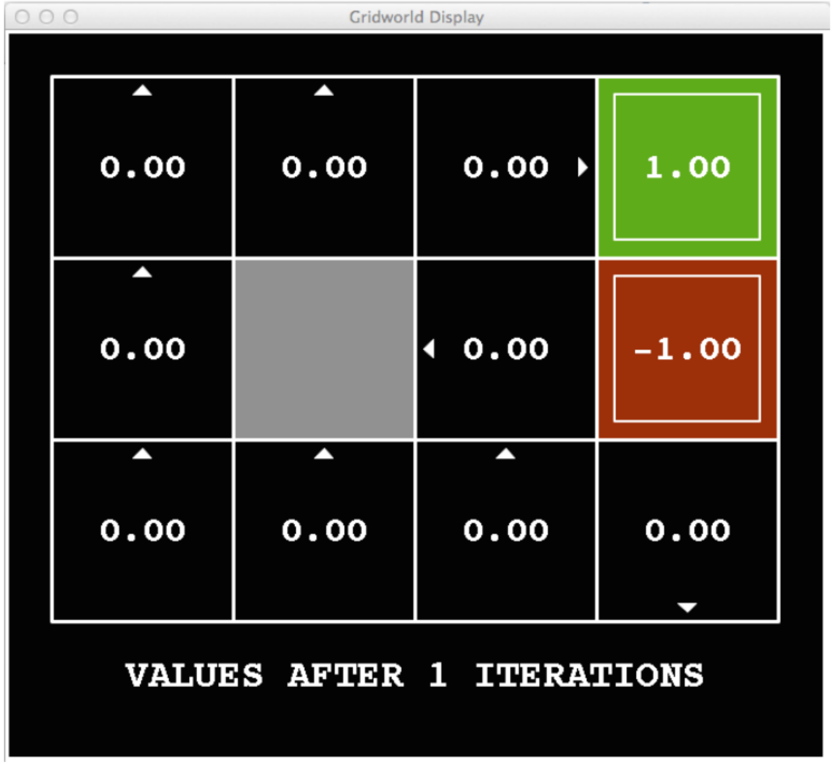
[Demo – time-limited values (L8D6)]

k = 0



Noise = 0.2
Discount = 0.9
Living reward = 0

k = 1



Noise = 0.2
Discount = 0.9
Living reward = 0

k = 2

Noise = 0.2
Discount = 0.9
Living reward = 0

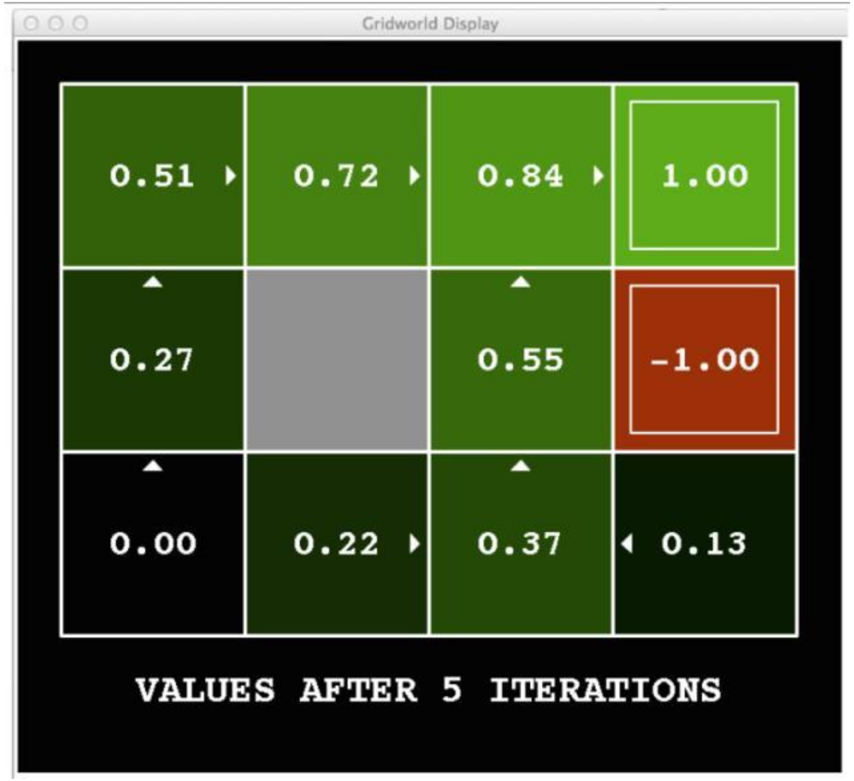
k = 3

Noise = 0.2
Discount = 0.9
Living reward = 0

k = 4

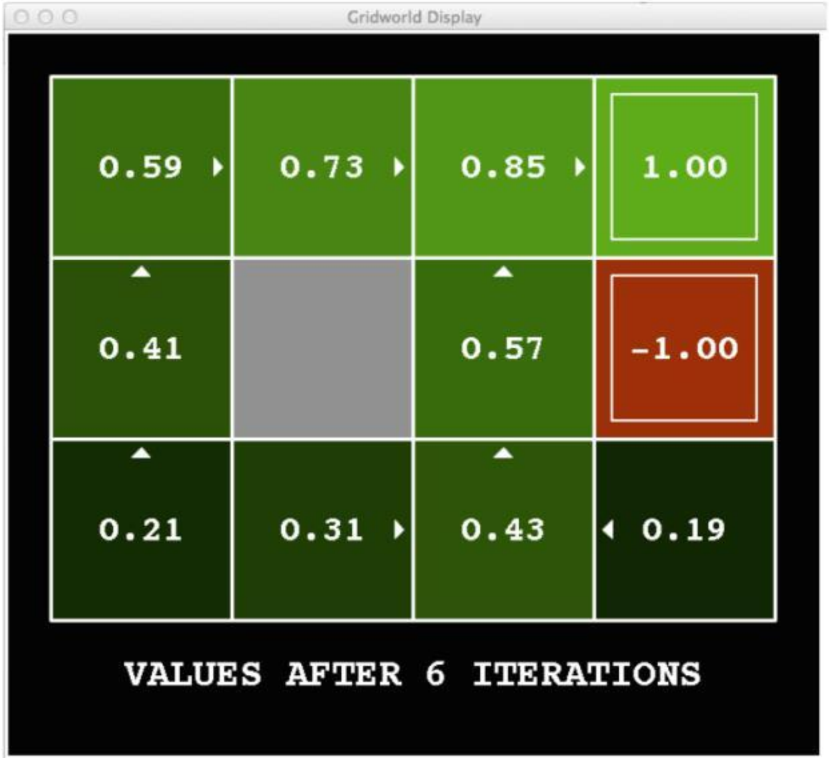
Gridworld Display
0.37
0.66
0.83
1.00
0.00
0.51
-1.00
0.00
0.00
0.31
0.00
VALUES AFTER 4 ITERATIONS

k = 5

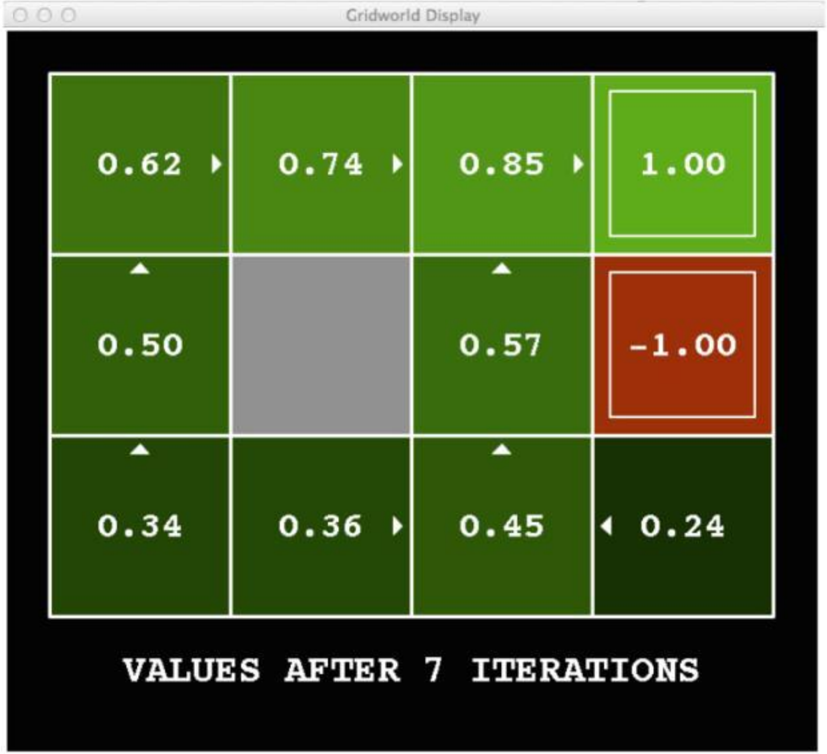


Gridworld Display
0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS

k = 6


Gridworld Display
0.59
0.73
0.85
1.00
0.41
0.57
-1.00
0.21
0.31
0.43
0.19
VALUES AFTER 6 ITERATIONS

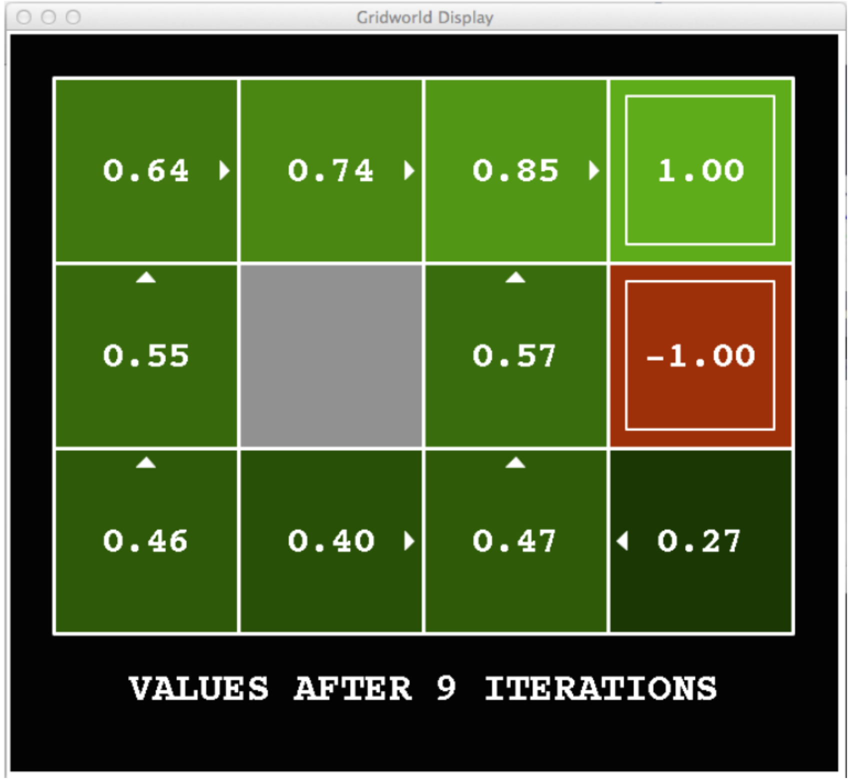
k = 7


Gridworld Display
0.62
0.74
0.85
1.00
0.50
0.57
-1.00
0.34
0.36
0.45
0.24
VALUES AFTER 7 ITERATIONS

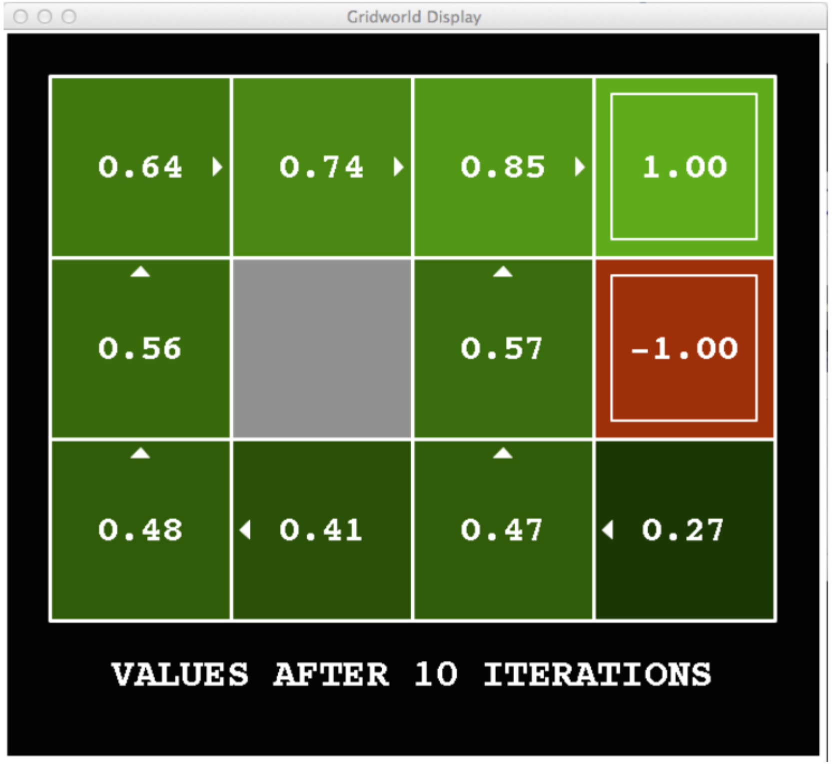
k = 8


Gridworld Display
0.63
0.74
0.85
1.00
0.53
0.57
-1.00
0.42
0.39
0.46
0.26
VALUES AFTER 8 ITERATIONS

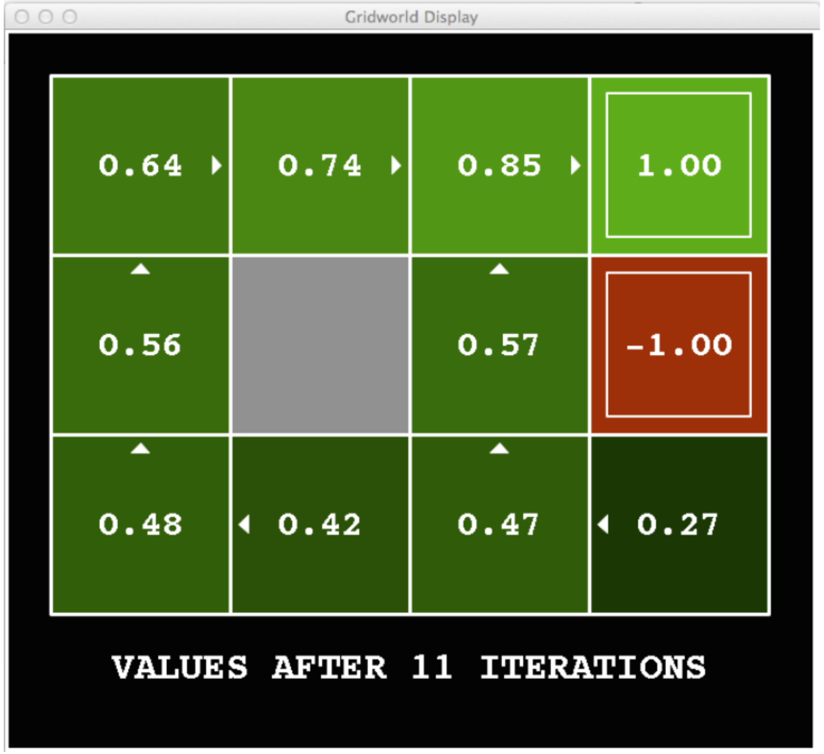
k = 9


Gridworld Display
0.64
0.74
0.85
1.00
0.55
0.57
-1.00
0.46
0.40
0.47
0.27
VALUES AFTER 9 ITERATIONS

k = 10


Gridworld Display
0.64
0.74
0.85
1.00
0.56
0.57
-1.00
0.48
0.41
0.47
0.27
VALUES AFTER 10 ITERATIONS

k = 11

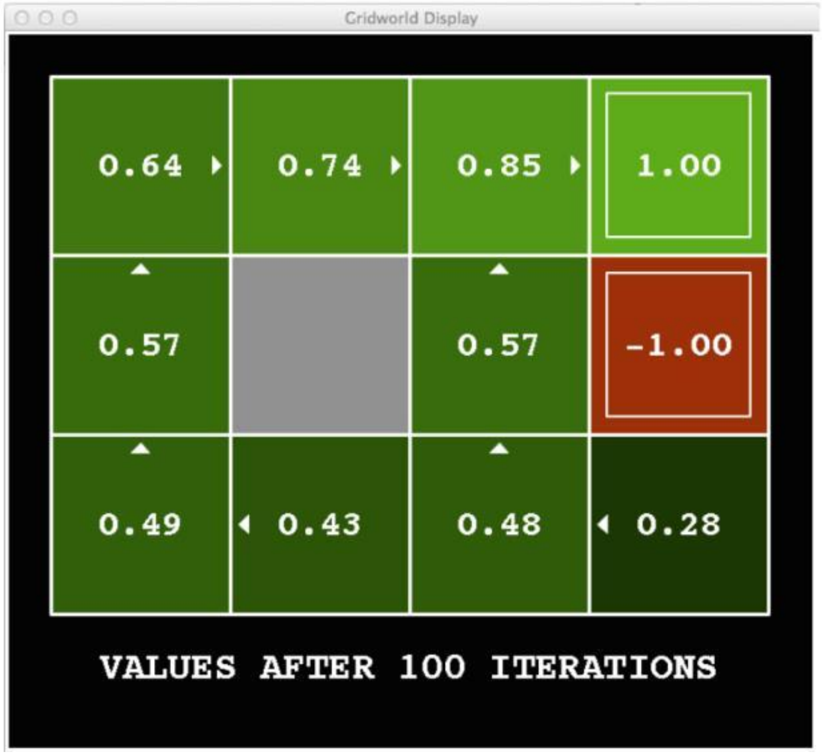


Gridworld Display
0.64
0.74
0.85
1.00
0.56
0.57
-1.00
0.48
0.42
0.47
0.27
VALUES AFTER 11 ITERATIONS

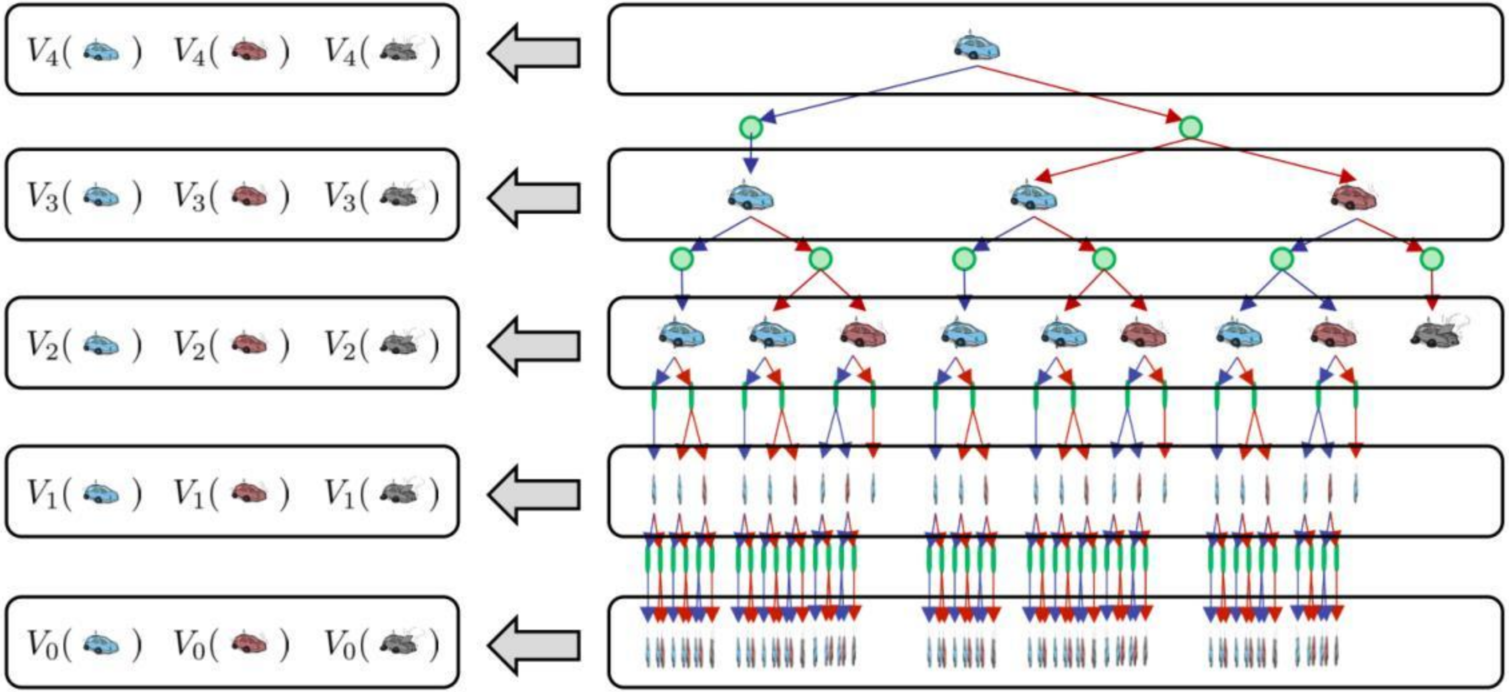
k = 12


Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.42
0.47
0.28
VALUES AFTER 12 ITERATIONS

k = 100

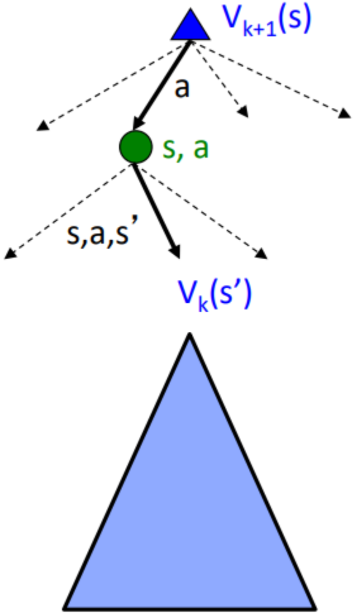

Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

محاسبه مقدارهای با زمان محدود



تکرار مقدار Value Iteration

تکرار مقدار Value Iteration

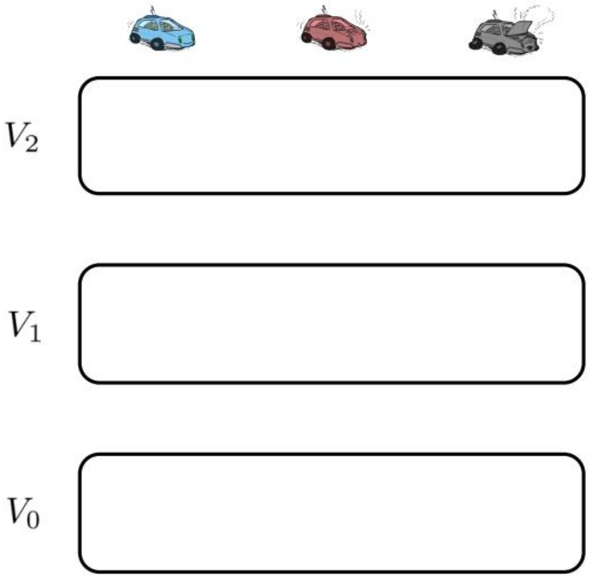


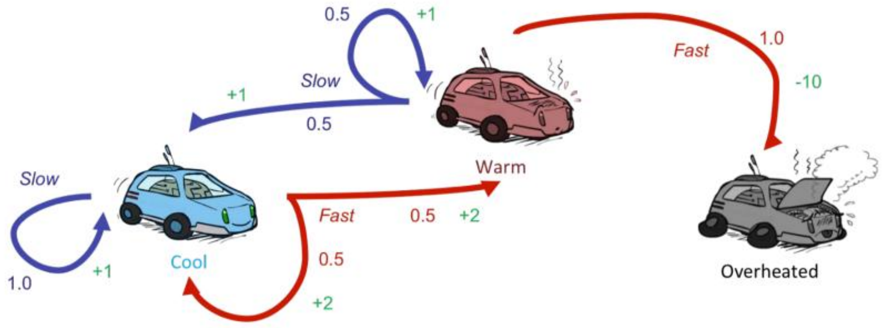
- شروع با $V_0(s) = 0$: باقی نماندن هیچ گام زمانی به معنای مجموع پاداش مورد انتظار صفر است
- با داشتن بردار $V_k(s)$ برای هر حالت یک لایه از expectimax را انجام بده:

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

- تا همگرایی تکرار کن
- پیچیدگی هر تکرار $O(S^2 A)$
- قضیه: به مقادیر بهینه یکتا همگرا خواهد شد
  - ایده پایه: تخمین‌ها به سمت مقادیر بهینه اصلاح می‌یابند
  - سیاست ممکن است خیلی زودتر از مقادیر همگرا شود

مثال: تکرار مقدار Value Iteration



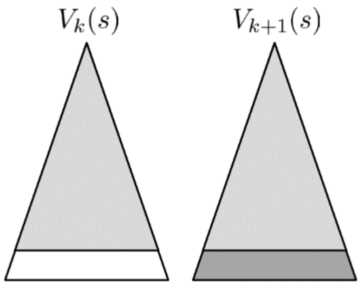


با فرض بدون تخفیف!

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

مثال: تکرار مقدار

- از کجا می دانیم که بردارهای V_k همگرا خواهند شد؟
- حالت 1: اگر درخت دارای بیشینه عمق M باشد، آنگاه V_m شامل مقادیر واقعی است بدون آن که درخت برش خورده باشد
- حالت 2: اگر ضریب تخفیف کمتر از 1 باشد
  - طرح کلی : به ازای هر حالت، مقادیر V_k و V_{k+1} را می توان به عنوان دو درخت expectimax با عمق $k + 1$ تقریباً یکسان در نظر گرفت
  - تفاوت این دور درخت در این است که در آخرین لایه V_{k+1} شامل پاداش‌های واقعی است اما V_k شامل پاداش‌های مساوی صفر است
  - پاداش‌ها در این لایه همگی حداکثر R_{max} هستند
  - و حداقل برابر با R_{min} هستند.
  - اما همه چیز با ضریب γ^k تخفیف یافته است
  - بنابراین اخلاف V_k و V_{k+1} حداکثر برابر با $\gamma^k \max|R|$ است
  - در نتیجه با افزایش k مقدارهای محاسبه شده همگرا خواهد شد



دفعه بعد: روش‌های مبتنی بر سیاست