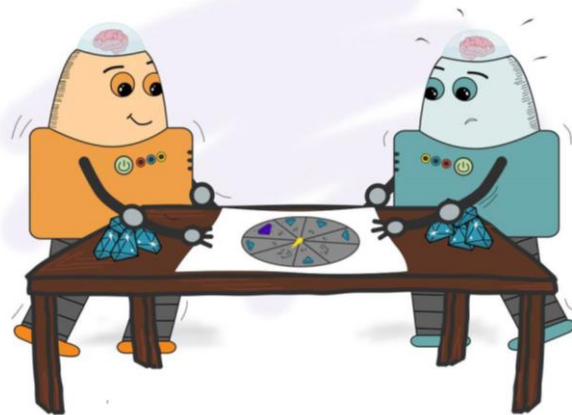


# مبانی و کاربردهای هوش مصنوعی

## جستجوی خصمانه 2 - عدم قطعیت (فصل 5.2 الی 5.5)



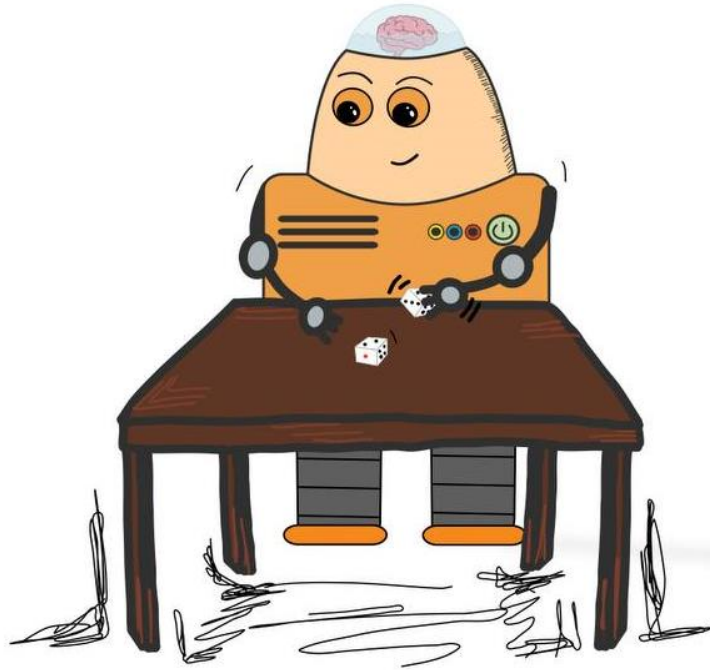
مدرس: مهدی جوانمردی

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر

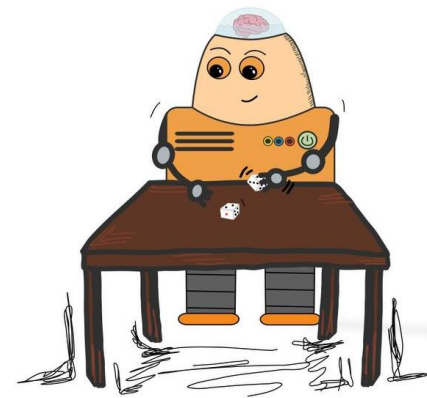
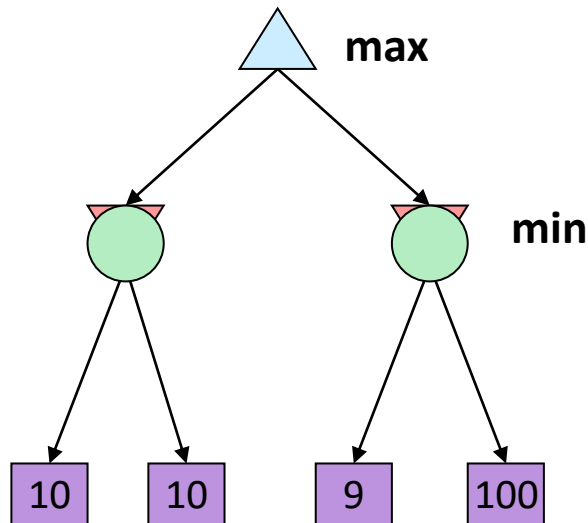
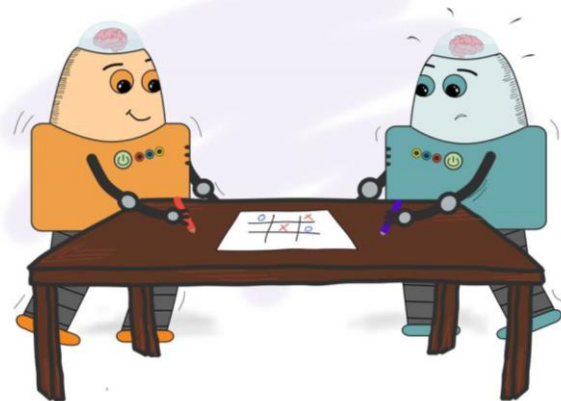


(الهام گرفته از محتوای درس هوش مصنوعی دانشگاه برکلی)

# عدم قطعیت در خروجی



# بدترین حالت در برابر حالت میانگین



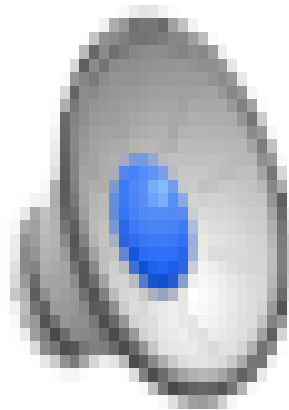
ایده: نتایج غیر قطعی توسط شانس کنترل می‌شوند، نه خصومت!

# جستجوی Expectimax

- چرا ما نتیجه یک عمل را نمی‌دانیم؟
  - تصادفی بودن محض: پرتاب تاس
  - رقبای غیر قابل پیش‌بینی: روح‌ها تصادفی پاسخ می‌دهند
  - اعمال ممکن است شکست بخورند:
  - موقع حرکت ربات ممکن است چرخ‌ها لیز بخوردند
- مقادیر باید نمایانگر نتایج میانگین باشند  
(Expectimax)، نه بدترین حالت نتایج (Minimax)
- جستجوی Expectimax: محاسبه امتیاز میانگین در صورت بازی بهینه
  - گره‌های پیشینه مشابه جستجوی کمینه-بیشینه
  - گره‌های شانس شبیه به گره‌های کمینه ولی نتیجه غیرقطعی
  - محاسبه سودمندی مورد انتظار
  - یعنی میانگین وزنی (مورد انتظار) فرزندان را می‌گیریم
- در ادامه، یاد خواهیم گرفت که چگونه مسائل نتایج نامعلوم را به صورت فرآیندهای تصمیم‌گیری مارکوف رسمی‌سازی کنیم.

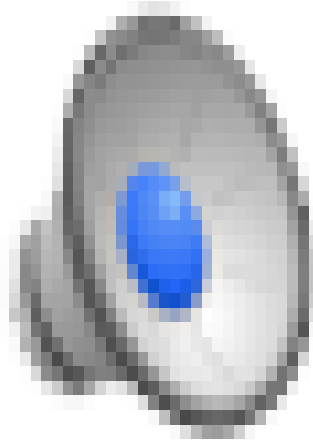
# دموی Minimax در برابر Expectimax (Minimax)

---



# دموی Minimax در برابر Expectimax (Expectimax)

---



# شبه کد Expectimax

```
def value(state):
```

if the state is a terminal state: return the state's utility

if the next agent is MAX: return max-value(state)

if the next agent is EXP: return exp-value(state)

```
def max-value(state):
```

initialize  $v = -\infty$

for each successor of state:

$v = \max(v, \text{value}(\text{successor}))$

return  $v$

```
def exp-value(state):
```

initialize  $v = 0$

for each successor of state:

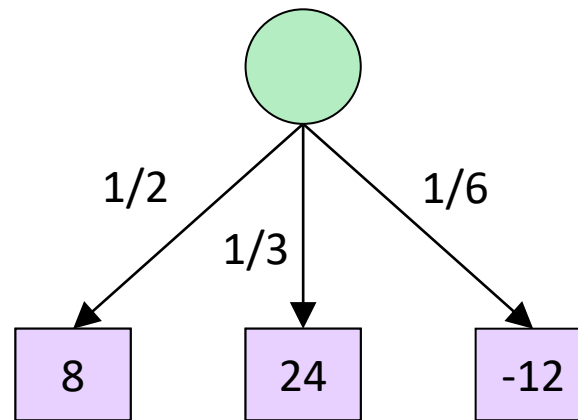
$p = \text{probability}(\text{successor})$

$v += p * \text{value}(\text{successor})$

return  $v$

# شبه کد Expectimax

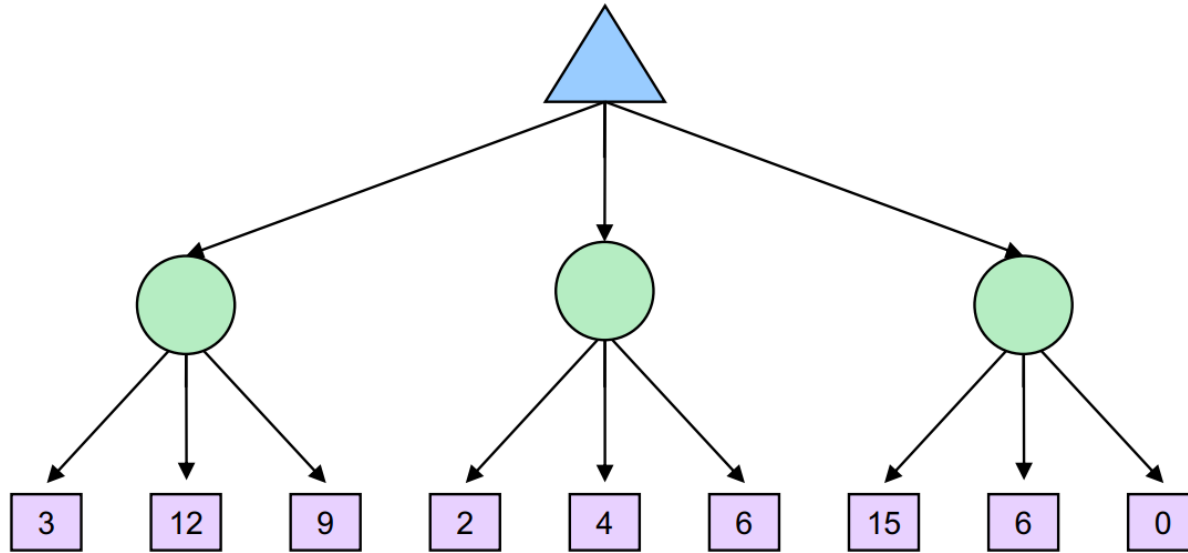
```
def exp-value(state):  
    initialize v = 0  
    for each successor of state:  
        p = probability(successor)  
        v += p * value(successor)  
    return v
```



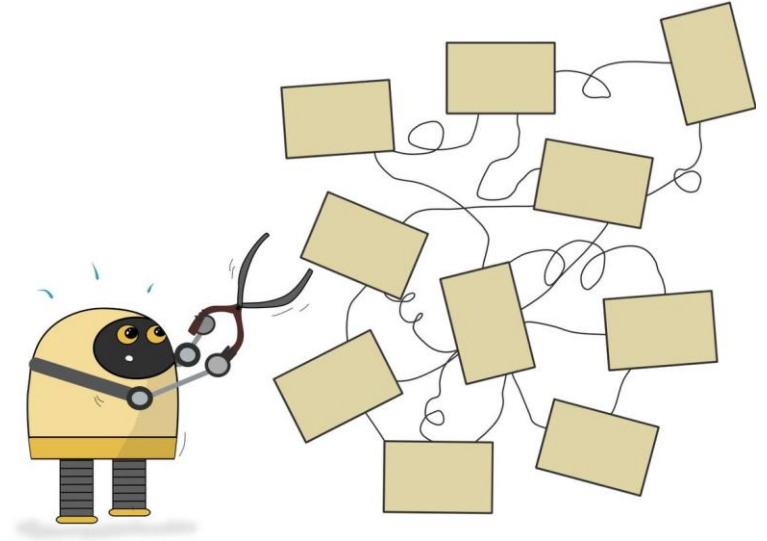
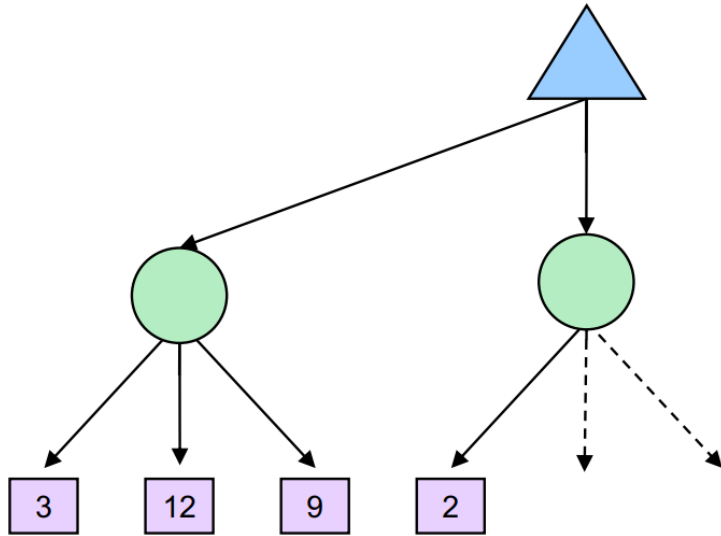
$$v = (1/2) (8) + (1/3) (24) + (1/6) (-12) = 10$$



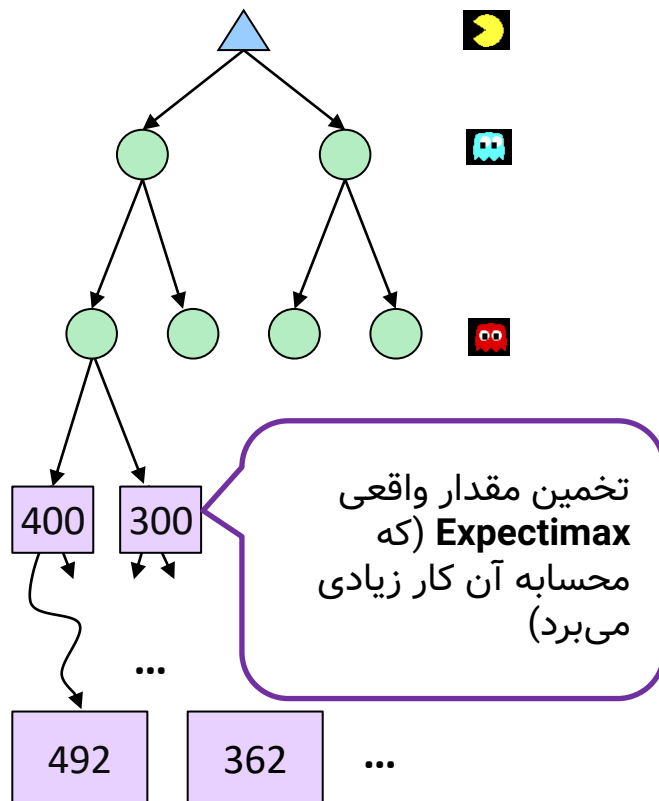
# مثال Expectimax



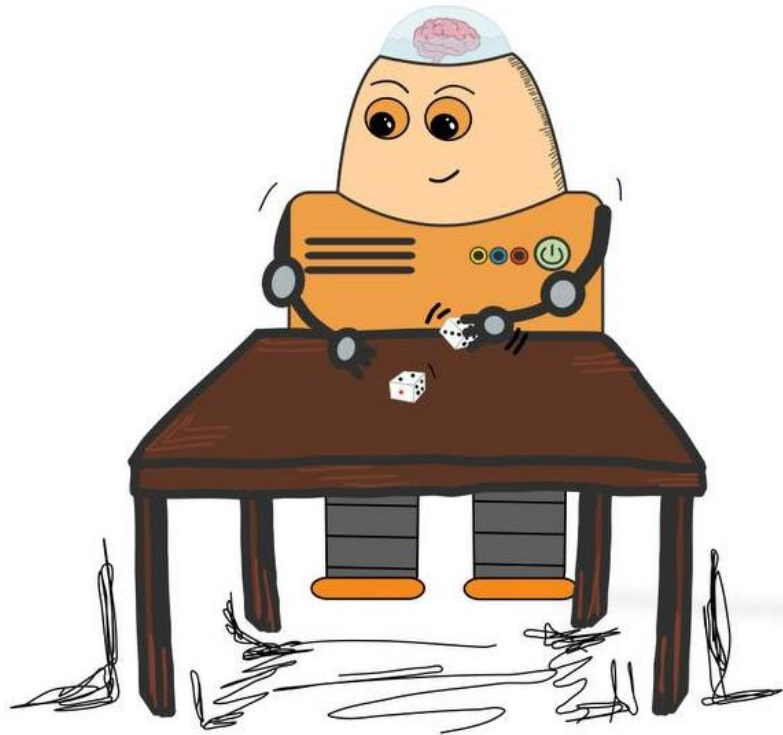
# هرس Expectimax ؟



# Expectimax با عمق محدود (Depth-Limited)



# احتمالات (Probabilities)



# یادآوری: احتمالات

مثال: ترافیک در آزادراه

▪ یک متغیر تصادفی نمایانگر یک اتفاق است که نتیجه‌ی آن نامعلوم است.

▪ متغیر تصادفی:  $T$  = آیا ترافیک است؟

▪ یک توزیع احتمال یک تخصیص وزن به نتایج است.

▪ نتایج:  $T$  در  $\{heavy, light, none\}$

▪ تعدادی از قوانین احتمال (بعداً، تعدادی بیشتر)

▪ توزیع:  $P(T=none) = 0.25$

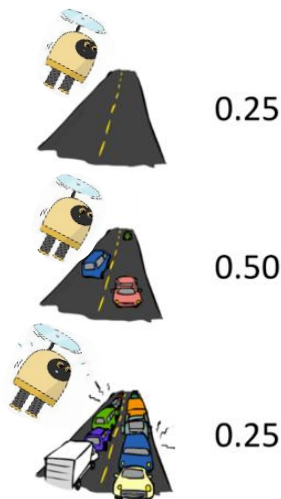
▪ احتمالات همیشه غیرمنفی هستند.

$P(T=light) = 0.50$

▪ مجموع احتمالات همه‌ی نتایج، یک است.

$P(T=heavy) = 0.25$

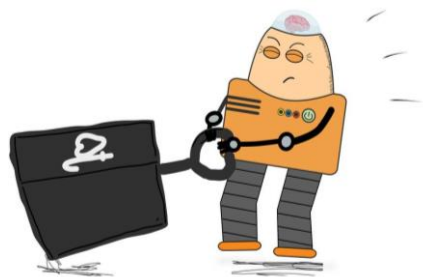
▪ همینطور که شواهد بیشتر می‌شود، احتمالات ممکن است تغییر کند.



▪  $P(T=heavy) = 0.25, P(T=heavy | Hour=8am) = 0.60$

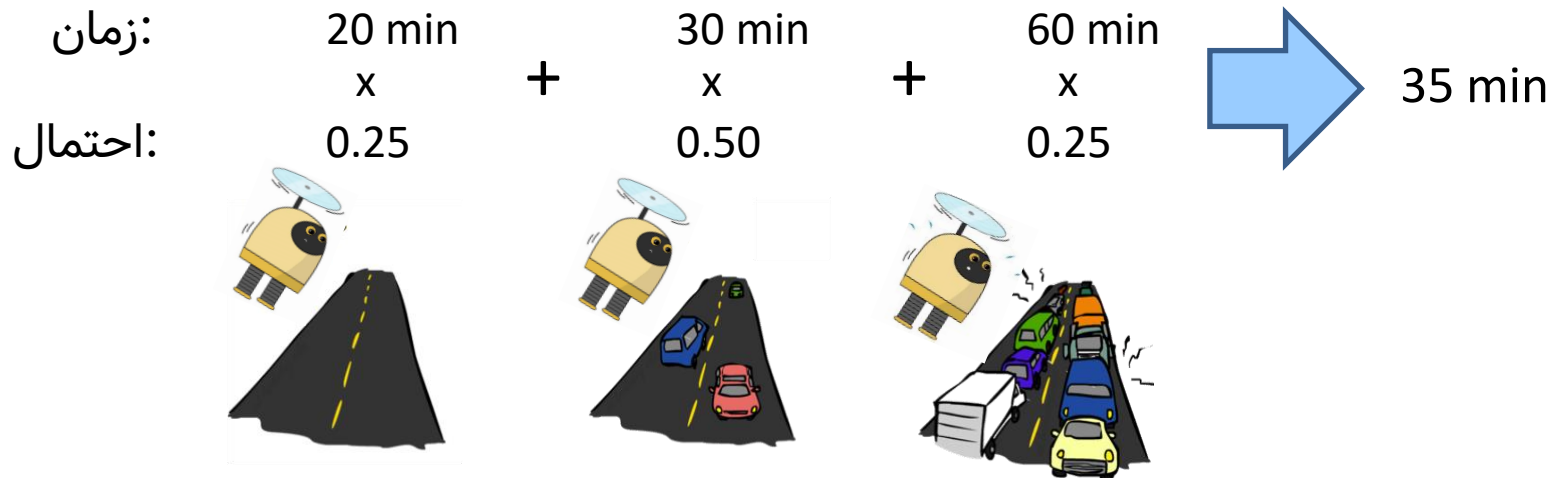
▪ بعداً در مورد روش‌های استدلال و به روزرسانی احتمالات بیشتر صحبت خواهیم کرد.

# یادآوری: انتظارات

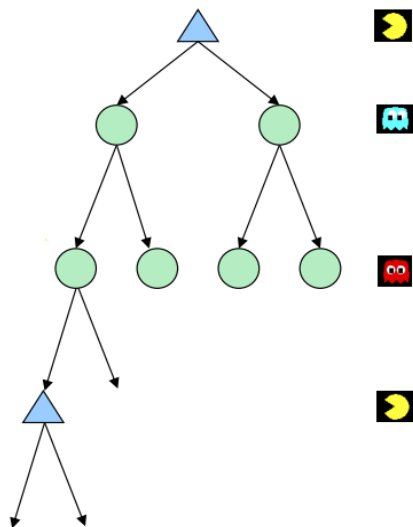


مقدار مورد انتظار تابع یک متغیر تصادفی، میانگین نتایج با وزنهای توزیع احتمال است.

مثال: چقدر طول می کشد تا به فرودگاه برسیم؟



# از چه احتمالاتی استفاده شود؟



در جستجوی Expectimax ما یک مدل احتمالاتی از نحوه رفتار رقیب (یا محیط) در هر حالت داریم.

▪ مدل می‌تواند یک توزیع ساده یکسان باشد (پرتاب تاس)

▪ مدل می‌تواند پیچیده باشد و به مقدار زیادی محاسبات نیاز داشته باشد

▪ ما از گره شانس (Chance Node) برای هر نتیجه‌ی خارج از کنترلمان داریم: رقیب یا محیط

▪ ممکن است مدل بگوید که اعمال خصمانه با احتمالی ممکن است  $\text{expectimax} < -$

▪ فعلاً، فرض کنید هر گره شانس به صورت جادویی توزیع احتمالات نتایج را دارد

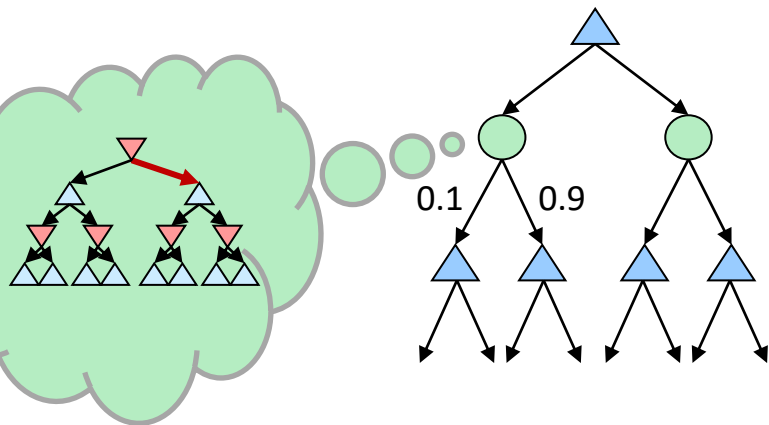
داشتن یک باور احتمالاتی در مورد  
اعمال یک عامل دیگر به معنای این  
نیست که عامل سکه می‌اندازد!

# آزمونک: احتمالات آگاهانه (Informed Probabilities)

- فرض کنید می‌دانید که حریف شما در اصل یک کمینه-بیشینه با عمق دو اجرا می‌کند، 80% مواقع از نتایج آن استفاده می‌کند، در غیر اینصورت تصادفی حرکت می‌کند
- سوال: از کدام جستجوی درختی استفاده می‌کنید؟

▪ جواب: Expectimax

- برای پیدا کردن احتمال هر گره شانس، شما باید یک شبیه‌سازی از رقیب اجرا کنید
- این تیپ شبیه‌سازی‌ها به سرعت کند می‌شود
- حتی بدتر اگر شبیه‌سازی رقیب از خودتان را شبیه‌سازی کنید...
- ... به جز برای کمینه-بیشینه، که ویژگی خوبی دارد که همه‌ی آن درون یک درخت بازی خلاصه می‌شود





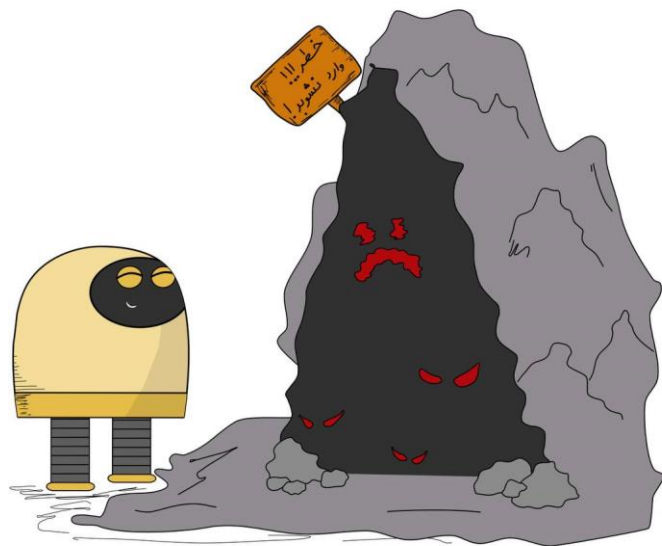
# مفروضات مدل سازی (Modeling Assumptions)



# خطرات خوش بینی و بدبینی

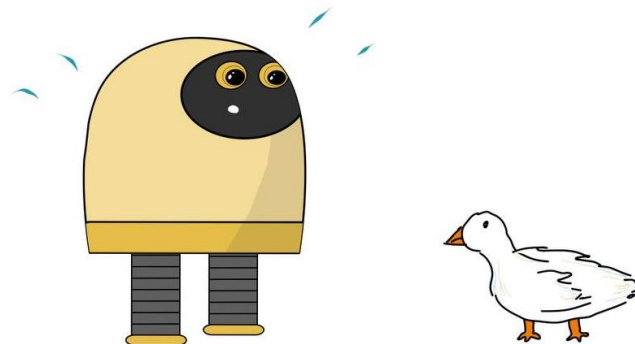
خوش بینی خطرناک

شانس را در نظر گرفتن وقتی که جهان خصمانه است

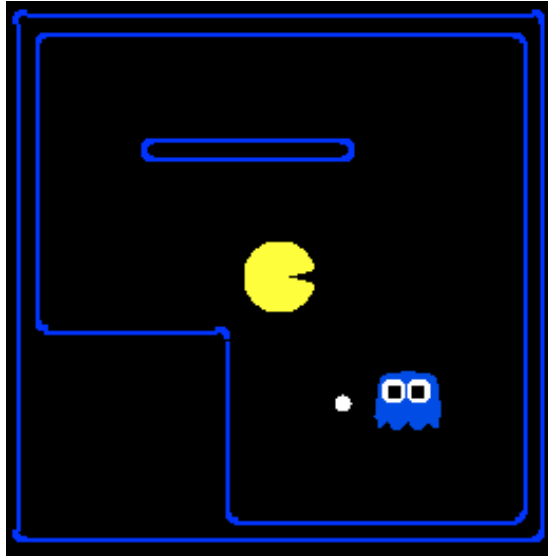


بدبینی خطرناک

بدترین حالت را در نظر گرفتن وقتی که محتمل نیست



# مفروضات در برابر واقعیت



	Adversarial Ghost	Random Ghost
Minimax Pacman		
Expectimax Pacman		

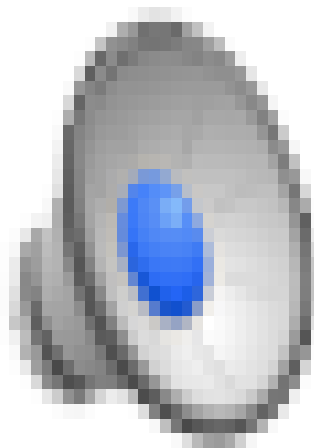
نتایج بدست آمده از 5 بازی

استفاده پکمن از جستجوی عمق 4 با تابع ارزیابی‌ای که از مشکلات پرهیز کند  
استفاده روح از جستجوی عمق 2 با تابع ارزیابی‌ای که به دنبال پکمن است

[Demos: world assumptions (L7D3,4,5,6)]

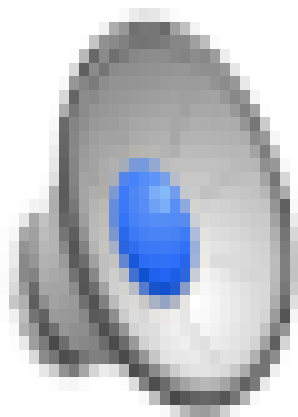
## World Assumptions Random Ghost – Expectimax Pacman دموى

---



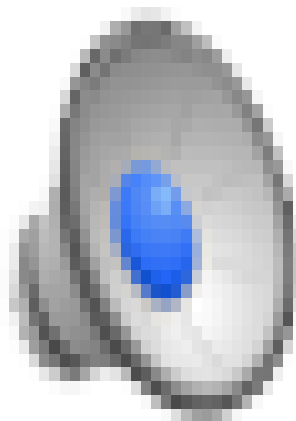
# World Assumptions Adversarial Ghost – Minimax Pacman دموى

---



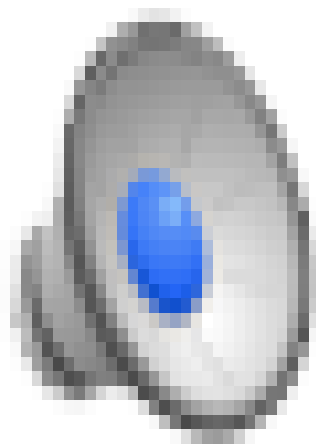
# World Assumptions Adversarial Ghost – Expectimax Pacman ویدیوی نمایشی

---

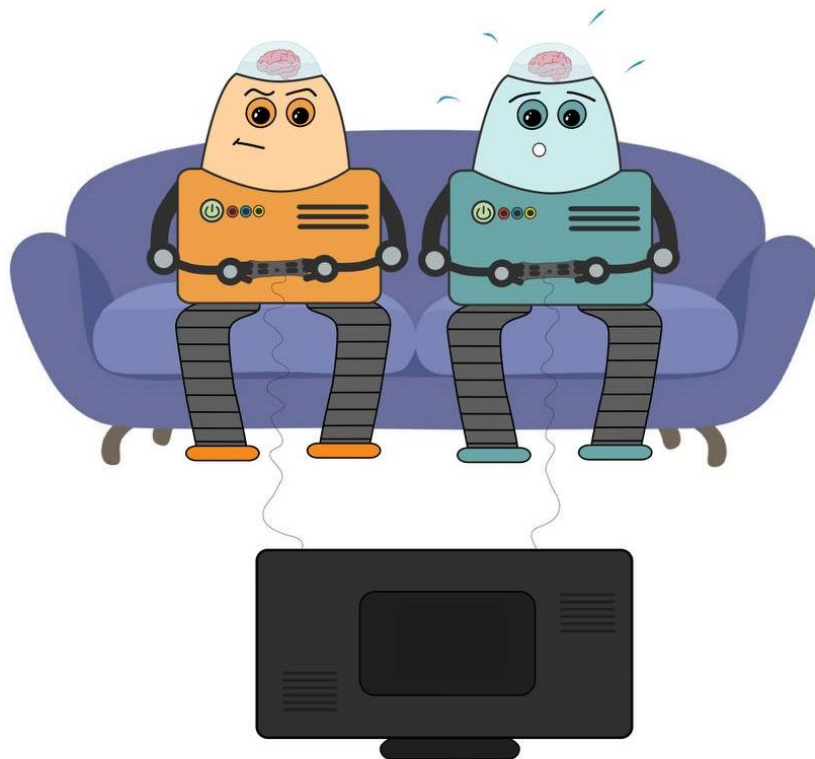


# World Assumptions Random Ghost – Minimax Pacman ویدیوی نمایشی

---



# انواع دیگر بازی‌ها



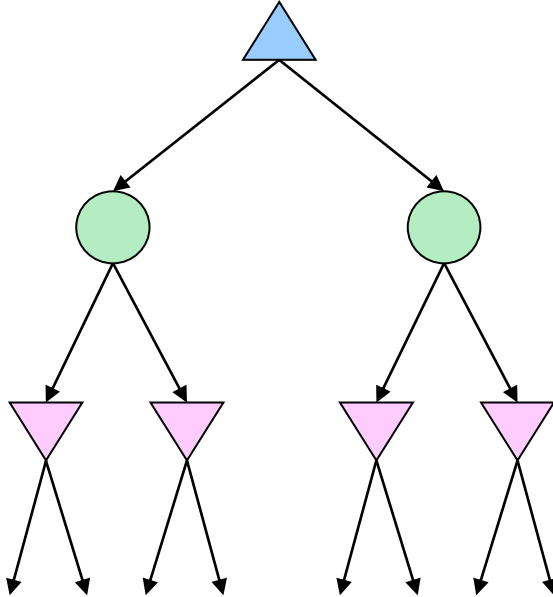


# گونه‌هایی با لایه‌های مختلط (Mixed Layer Types)

▪ مثل تخته نرد (Backgammon)

▪ Expectiminimax

- محیط، یک بازیکن "عامل تصادفی" اضافه است که بعد از هر عامل کمینه-بیشینه حرکت می‌کند
- هر گره ترکیب مناسب فرزندانش را محاسبه می‌کند



## مثال: تخته نرد



- با دو تاس، پرتاب‌های تاس به اندازه  $b=21$  پرتاب ممکن افزایش می‌یابد.

- تخته نرد  $\approx 20$  حرکت قانونی

- عمق  $20 \times (21 \times 20)^3 = 1.2 \times 10^9 = 2$

- همینطور که عمق افزایش می‌یابد، احتمال رسیدن به یک گره جستجوی داده شده کم می‌شود

- بنابراین سودمندی جستجو کم می‌شود

- بنابراین کم کردن عمق کمتر آسیب می‌زند

- ولی هرس سخت‌تر می‌شود...

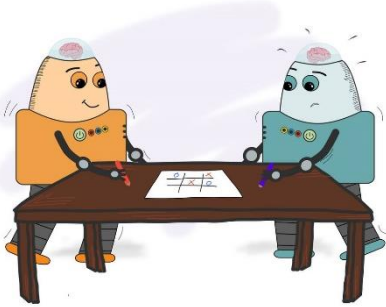
هوش مصنوعی تاریخی: TDGammon از جستجوی با عمق 2 +

تابع ارزیابی خیلی خوب + یادگیری تقویتی استفاده می‌کند:

بازی در سطح قهرمانی جهانی

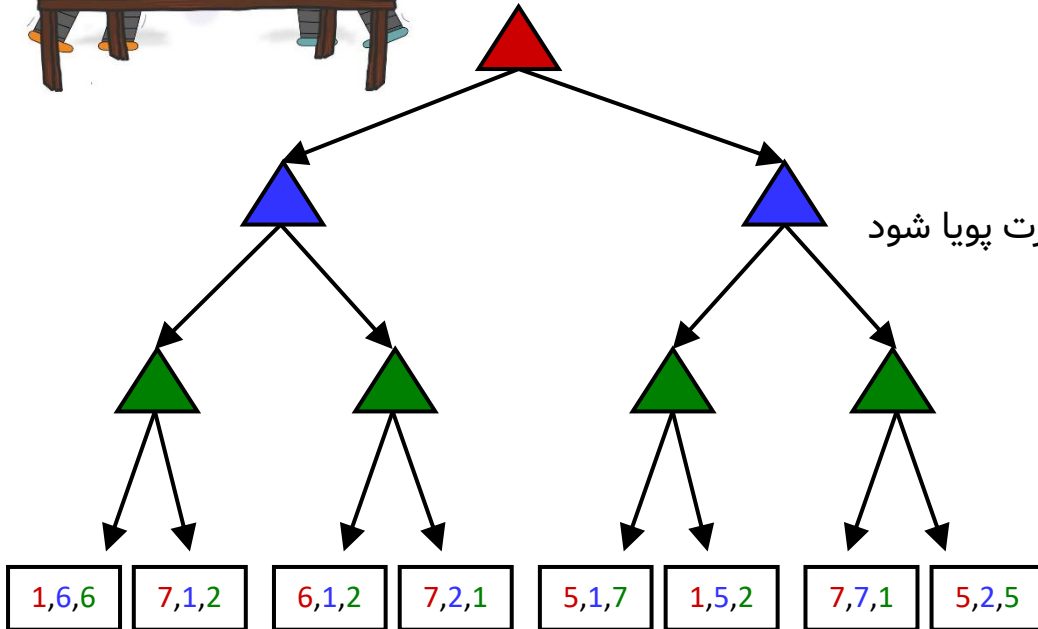
اولین قهرمان جهان هوش مصنوعی در هر بازی!

# سودمندی‌های چند عامله



- اگر بازی مجموع صفر نبود یا چند بازیکن داشت چی؟
- تعمیم کمینه-بیشینه:

- خروجی‌ها ردیف‌های سودمندی دارند
- مقادیر گره‌ها نیز ردیف‌های سودمندی دارند
- هر بازیکنی بخش خودش را بیشینه می‌کند
- می‌تواند باعث رشد همکاری و رقابت به صورت پویا شود



# دفعه بعد: MDP ها!

---