جزوه درس هوش مصنوعی - جلسه سوم

مقدمه و اطلاعات کلی درس

راه‌های ارتباطی

- اصلی‌ترین راه ارتباطی با استاد و دستیاران :**LMS (سیستم مدیریت یادگیری)**
- یا ایمیل SMS امکان ارسال پیام و دریافت اعلان از طریق :**Quera**
- **ایمیل دانشگاهی**: برای مکاتبات رسمی و دسترسی در شرایط محدودیت اینترنت

توصیه استاد: استفاده از LMS به دلیل بررسی مکرر توسط استاد

شرایط خاص ترم

- به دلیل شرایط ویژه، برنامه‌ریزی‌های اولیه تغییر خواهد کرد
- تاریخ امتحان میان‌ترم قطعی نیست و ممکن است تغییر کند یا به پایان‌ترم موکول شود
- تمام ضرب‌الاجل‌ها قابل تمدید هستند
- اولویت با سلامتی و امنیت دانشجویان است

فصل دوم: عامل‌ها (Agents) و محیط‌های کاری

Rational Agent (عامل خردمند)

تعریف Rational Agent

عامل خردمند (Rational Agent) عاملی است که در هر شرایطی که قرار می‌گیرد، از مجموع اقداماتی که می‌تواند انجام دهد، اقدامی را انتخاب می‌کند که **سودمندی مورد انتظار را بیشینه** می‌کند.

مفهوم Rationality (خردمندی)

خردمندی یک عامل بستگی دارد به:

- **سنسورها (Sensors)**: ابزارهای دریافت اطلاعات از محیط
- **اکچویتورها (Actuators)**: ابزارهای اثرگذاری بر محیط
- **محیط کار**: نوع و ویژگی‌های محیطی که عامل در آن فعالیت می‌کند
- **کفایت و لزوم**: آیا سنسورها و اکچویتورها برای آن محیط کافی و مناسب هستند

نکته مهم اخلاقی

تعریف سودمندی نباید صرفاً به سود یک عامل محدود شود. باید در نظر گرفت که:

- عامل‌های دیگر نیز در محیط فعالیت می‌کنند
- نباید سود خود را بیشینه کنیم به قیمت آسیب به دیگران
- مسائل اخلاقی (Ethics) در طراحی عامل‌های هوشمند اهمیت دارد

ساختار کلی یک عامل

مدل انتزاعی عامل

محیط (Environment)

```
↓ (Percepts - ادراکات)
```

سنسورها (Sensors)

↓

عملکرد داخلی (Agent Function)

↓

اکچویتورها (Actuators)

↓ (Actions - اقدامات)

محیط (Environment)

خلاصه: عامل محیط را از طریق سنسورها درک می‌کند و از طریق اکچویتورها بر آن اثر می‌گذارد.

مثال: انسان به عنوان عامل

خود ما (انسان‌ها) نیز عامل محسوب می‌شویم:

- **سنسورها**: چشم، گوش، پوست، بینی، زبان
- **اکچویتورها**: دست، پا، زبان (برای صحبت)، عضلات
- **محیط**: دنیای فیزیکی اطراف ما
- **عملکرد**: مغز و سیستم عصبی

✨ مثال: ماشین‌حساب

حتی یک ماشین‌حساب ساده نیز یک عامل است:

- **سنسورها**: دکمه‌ها (ورودی اعداد و عملگرها)
- **اکچویتورها**: صفحه نمایش
- **محیط**: کاربر و اعداد ورودی
- **عملکرد**: محاسبات ریاضی

Agent Function vs Agent Program

📖 Agent Function (تابع عامل)

تابع عامل نگاشتی است از **دنباله ادراکات** به **اقدامات**:

$$f : P^* \to A$$

که در آن:

- P^*: دنباله‌ای از ادراکات (Percept Sequence)
- A: مجموعه اقدامات ممکن (Actions)

📖 Agent Program (برنامه عامل)

پیاده‌سازی عملی تابع عامل است که:

- می‌تواند نرم‌افزاری یا سخت‌افزاری باشد
- بستگی به ماشین اجرایی دارد
- محدودیت‌های عملی دارد (حافظه، سرعت پردازش، تأخیر)

💡 *محدودیت‌های عملی*

عملکرد یک عامل تحت تأثیر عوامل زیر است:

- **محدودیت حافظه**: نمی‌توان تمام تاریخچه را ذخیره کرد
- **محدودیت پردازش**: سرعت محاسبات محدود است
- **محدودیت مکانیکی**: سنسورها و اکچویتورها محدودیت فیزیکی دارند
- **تأخیر شبکه**: در سیستم‌های توزیع‌شده

به همین دلیل در عمل با **Bounded Rationality** (خردمندی محدود) سر و کار داریم.

سؤال مهم

آیا هر تابعی می‌تواند توسط برنامه‌ای پیاده‌سازی شود؟

پاسخ: خیر. ممکن است تابع عامل مورد نظر ما در شرایط حاضر قابل پیاده‌سازی نباشد به دلایل:

- محدودیت‌های محاسباتی
- محدودیت‌های فناوری
- پیچیدگی محاسباتی بالا

مثال کاربردی: جاروبرقی خودکار (Vacuum Cleaner)

محیط ساده جاروبرقی

توضیح محیط:

- دو اتاق (یا دو خانه): A و B
- هر خانه می‌تواند تمیز یا کثیف باشد
- جاروبرقی می‌تواند در یکی از دو خانه باشد

بردار ادراک (Percept Vector):

- مکان فعلی: $\{A, B\}$
- وضعیت کثیفی: $\{Clean, Dirty\}$

اقدامات ممکن (Actions):

- $Left$: حرکت به چپ
- $Right$: حرکت به راست
- $Suck$: مکیدن (تمیز کردن)
- $NoOp$: عدم انجام کار

Simple Reflex Agent (عامل واکنشی ساده)

عامل واکنشی ساده بر اساس **قوانین شرطی If-Then** عمل می‌کند:

```
IF مکان = A AND وضعیت = Dirty THEN Suck
IF مکان = A AND وضعیت = Clean THEN Right
IF مکان = B AND وضعیت = Dirty THEN Suck
IF مکان = B AND وضعیت = Clean THEN Left
```

ویژگی: تمام حالات ممکن از پیش تعریف شده‌اند و عامل صرفاً واکنش نشان می‌دهد.

جدول تابع عامل

برای جاروبرقی با دو خانه، جدول کامل تابع عامل شامل تمام دنباله‌های ممکن ادراکات است:

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
...	...

معیار عملکرد (Performance Measure)

Performance Measure

معیار عملکرد مشخص می‌کند که چگونه موفقیت یک عامل را ارزیابی کنیم. این معیار بستگی به **مسئله** و **محیط کار** دارد.

معیارهای مختلف برای جاروبرقی

برای یک جاروبرقی خودکار می‌توان معیارهای مختلفی تعریف کرد:

- میزان الکتریسیته مصرفی (کمتر بهتر)
- تعداد حرکات انجام شده (کمتر بهتر)
- میزان تمیزی محیط (بیشتر بهتر)
- تعداد دفعات مکش (کمتر بهتر)
- زمان صرف شده (کمتر بهتر)

نکته: هر کاربر ممکن است اولویت‌های متفاوتی داشته باشد.

مثال: هوش ماهی

نمی‌توان هوش ماهی را بر اساس توانایی بالا رفتن از درخت سنجید!

درس: هر عامل برای محیط و وظیفه خاصی طراحی شده است. معیار عملکرد باید متناسب با آن محیط و وظیفه باشد.

ویژگی‌های عامل‌های خردمند

Omniscience (عالم مطلق) نیستند

عامل‌های خردمند لزوماً همه‌چیز را نمی‌دانند:

- محدود به اطلاعاتی هستند که از سنسورها دریافت می‌کنند
- نمی‌توانند آینده را با قطعیت پیش‌بینی کنند
- با عدم قطعیت دست و پنجه نرم می‌کنند

Learning (یادگیری)

یادگیری یکی از **درجات اصلی هوشمندی** است:

- محیط‌های کاری وجود دارند که نمی‌توان تمام حالات را از پیش تعریف کرد
- رویدادهای غیرمنتظره رخ می‌دهند
- عامل باید **کشف (Explore)** کند و **یاد بگیرد (Learn)**

مثال: اگر طراح تمام شرایط را پیش‌بینی نکرده باشد، عامل باید بتواند با شرایط جدید سازگار شود.

Autonomy (خودمختاری)

بالاترین درجه هوشمندی:

- عامل بدون مداخله مستمر صاحب خود کار می‌کند
- مستقل تصمیم می‌گیرد
- نیازی به تعامل دائمی با کاربر ندارد

توجه: خودمختاری به معنای مخرب بودن نیست، بلکه به معنای استقلال در انجام وظیفه محول شده است.

اشتباه کردن

آیا عامل‌های خردمند اشتباه می‌کنند؟

- اگر عامل طبق طراحی عمل کند، اشتباه نکرده است
- اگر چیزی انجام نشود، ممکن است به دلیل محدودیت در طراحی باشد
- عامل باید بتواند از طریق یادگیری خود را بهبود دهد

مدل PEAS

📖 مدل PEAS

برای تعریف کامل یک مسئله عامل، باید چهار جزء را مشخص کنیم:

- Performance measure: معیار عملکرد
- Environment: محیط کار
- Actuators: اکچویتورها (ابزارهای اثرگذاری)
- Sensors: سنسورها (ابزارهای دریافت اطلاعات)

مثال‌های PEAS

✨ PEAS برای Pac-Man

Performance:

- -1 به ازای هر حرکت (مصرف انرژی)
- $+10$ به ازای خوردن هر غذا
- $+200$ به ازای گرفتن هر روح
- -500 به ازای مرگ

Environment:

- ماز (Maze) با دیوارها
- غذاها (Pellets)
- روح‌ها (Ghosts) - می‌توانند ثابت یا متحرک باشند
- Power Pellets

Actuators:

- حرکت به چپ، راست، بالا، پایین

Sensors:

- موقعیت Pac-Man
- موقعیت غذاها
- موقعیت روح‌ها
- وضعیت Power Pellet (مدت زمان آن قابل مشاهده نیست برای Pac-Man)

✨ PEAS برای تاکسی خودکار (Automated Taxi)

Performance:

- ایمنی مسافران
- رضایت مسافران (راحتی، سرعت)
- درآمد کسب شده
- هزینه‌های عملیاتی (سوخت، استهلاک)
- رعایت قوانین راهنمایی و رانندگی
- جریمه‌ها
- هزینه بیمه

Environment:

- خیابان‌ها و جاده‌ها
- سایر رانندگان و وسایل نقلیه
- عابران پیاده
- شرایط جوی (باران، برف، مه)
- پلیس
- چراغ‌های راهنمایی

Actuators:

- فرمان
- گاز
- ترمز
- چراغ‌ها
- بوق
- صفحه نمایش (تعامل با مسافر)

Sensors:

- دوربین‌ها (Camera)
- رادار
- سنسور سرعت
- GPS
- سنسور دمای موتور
- میکروفون
- سنسورهای فاصله

✨ PEAS برای سیستم تشخیص پزشکی

Performance:

- سلامت بیمار
- هزینه‌های درمانی
- دقت تشخیص
- اعتماد کاربران (پزشکان و بیماران)
- عدم بروز خطاهای پزشکی

Environment:

- بیماران
- کادر پزشکی
- بیمارستان
- سیستم بیمه
- سیستم قانونی (مسئولیت پزشکی)

Actuators:

- صفحه نمایش
- چاپگر
- ایمیل
- سیستم‌های هشدار

Sensors:

- کیبورد و ماوس
- دستگاه‌های تشخیص پزشکی (آزمایش‌ها، تصاویر)
- پرونده الکترونیک بیمار

انواع محیط‌ها (Environment Types)

💡 ابعاد طبقه‌بندی محیط‌ها

محیط‌های کاری را می‌توان بر اساس شش بُعد طبقه‌بندی کرد. هر محیط در هر بُعد می‌تواند در یکی از دو سمت قرار گیرد:

بُعد	سمت چپ (ساده‌تر)	سمت راست (پیچیده‌تر)
قابلیت مشاهده	Fully Observable	Partially Observable
تعداد عامل	Single Agent	Multi-Agent
قطعیت	Deterministic	Stochastic
پویایی	Static	Dynamic
گسستگی	Discrete	Continuous

نکته مهم: سمت چپ ساده‌ترین نوع محیط است. هر حرکت به سمت راست، پیچیدگی مسئله را افزایش می‌دهد.

1. Observable بودن محیط

🕮 Fully Observable vs Partially Observable

Fully Observable: عامل در هر لحظه به **تمام اطلاعات مرتبط** با تصمیم‌گیری دسترسی دارد.

Partially Observable: عامل فقط بخشی از اطلاعات محیط را می‌بیند.

نکته کلیدی: این ویژگی به **محیط** مربوط نمی‌شود، بلکه به **رابطه عامل با محیط** بستگی دارد.

✨ مثال: بستن چشم در اتاق

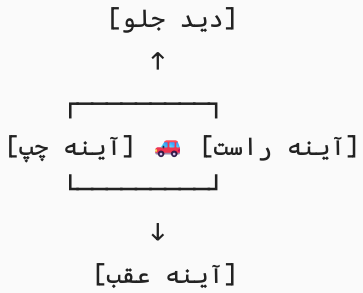
فرض کنید در یک اتاق روشن نشسته‌ایم:

- **شخص با چشم باز**: محیط برایش Fully Observable است
- **شخص با چشم بسته**: همان محیط برایش Partially Observable یا Unobservable است

نتیجه: Observable بودن به سنسورهای عامل بستگی دارد، نه خود محیط.

✨ مثال: نقاط کور خودرو

یک خودرو با سنسورهای محدود:



نقاط کور:

- گوشه‌های کنار خودرو
- مناطق بین دید آینه‌ها
- زیر خودرو
- بالای خودرو

راه‌حل: افزودن سنسورهای بیشتر (دوربین، رادار، سنسور فاصله) برای تبدیل محیط به Fully Observable

🔥 نکته رانندگی

به همین دلیل مربی رانندگی می‌گوید:

- هنگام سبقت گرفتن، فقط به آینه‌ها اکتفا نکنید
- سر خود را بچرخانید و از شیشه کناری نگاه کنید
- آینه‌ها نقطه کور دارند و ممکن است ماشین کناری را نبینید

2. Single Agent vs Multi-Agent

📖 Single Agent

تنها یک عامل در محیط فعالیت می‌کند و سایر اجزای محیط صرفاً اشیاء غیرفعال هستند.

📖 Multi-Agent

چندین عامل در محیط فعالیت می‌کنند که ممکن است:

- **همکاری** کنند (Cooperative)
- **رقابت** کنند (Competitive)
- ترکیبی از هر دو

✨ مثال Multi-Agent

- **شطرنج**: دو عامل رقیب
- **فوتبال**: دو تیم با اعضای متعدد (همکاری درون تیم، رقابت بین تیم‌ها)
- **ترافیک**: خودروهای متعدد که باید هماهنگ شوند

3. Deterministic vs Stochastic

📖 Deterministic (قطعی)

اگر عامل در حالت S اقدام A را انجام دهد، **همیشه** به حالت S' مشخصی می‌رسد.

$$S \xrightarrow{A} S' \quad \text{(همیشه)}$$

📖 Stochastic (تصادفی/احتمالی)

اگر عامل در حالت S اقدام A را انجام دهد، ممکن است به حالت‌های مختلفی برسد:

$$S \xrightarrow{A} \begin{cases} S_1 & \text{با احتمال } p_1 \\ S_2 & \text{با احتمال } p_2 \\ S_3 & \text{با احتمال } p_3 \end{cases}$$

✨ مثال: خانه عجیب لورل و هاردی

در فیلم‌های لورل و هاردی، خانه‌ای عجیب وجود دارد:

حالت Deterministic (اما عجیب):

- باز کردن در یخچال → **همیشه** آواز پخش می‌شود
- روشن کردن گاز → **همیشه** از جای دیگری آتش در می‌آید
- با اینکه عجیب است، اما قابل پیش‌بینی است

حالت Stochastic:

- باز کردن در یخچال → یک بار آواز، یک بار آتش، یک بار باران
- غیرقابل پیش‌بینی و تصادفی

✨ مثال: گام برداشتن روی سنگ‌های رودخانه

Deterministic:

- سنگ‌ها ثابت هستند
- هر بار که پا روی سنگ می‌گذارید، همان نتیجه حاصل می‌شود

Stochastic:

- سنگ‌ها ممکن است لیز باشند
- ممکن است سنگ حرکت کند
- نتیجه قابل پیش‌بینی نیست

4. Static vs Dynamic

📖 Static (ایستا)

محیط در حین تصمیم‌گیری عامل **تغییر نمی‌کند**. عامل می‌تواند هر قدر بخواهد فکر کند، محیط ثابت می‌ماند.

📖 Dynamic (پویا)

محیط **در حین تصمیم‌گیری** عامل تغییر می‌کند. اگر عامل زیاد فکر کند، ممکن است فرصت را از دست بدهد.

✨ مثال: عبور از رودخانه

Static:

- سنگ‌های رودخانه ثابت هستند
- می‌توانید هر قدر بخواهید فکر کنید و برنامه‌ریزی کنید
- سنگ‌ها منتظر شما می‌مانند

Dynamic:

- در حین فکر کردن، یک سنگ فرو می‌رود
- سنگ دیگری جای آن ظاهر می‌شود
- باید سریع تصمیم بگیرید وگرنه محیط تغییر می‌کند

💡 تفاوت Deterministic و Static

Deterministic vs Stochastic: درباره **نتیجه اقدام** است

- آیا نتیجه قطعی است یا تصادفی؟

Static vs Dynamic: درباره **تغییر محیط در طول زمان** است

- آیا محیط در حین تصمیم‌گیری تغییر می‌کند؟

مثال ترکیبی:

- خانه عجیب لورل و هاردی: Deterministic اما می‌تواند Dynamic باشد
- شطرنج: Deterministic و Static (حریف منتظر می‌ماند)
- رانندگی: Stochastic و Dynamic (ماشین‌های دیگر حرکت می‌کنند)

ابعاد محیط‌های کاری (Task Environment Dimensions)

در این بخش به بررسی ابعاد مختلف محیط‌هایی که عامل‌ها در آن فعالیت می‌کنند می‌پردازیم. هر محیط می‌تواند از نظر چندین بُعد مختلف طبقه‌بندی شود که هر کدام تأثیر مستقیمی بر طراحی و پیاده‌سازی عامل دارند.

1. قابلیت مشاهده (Observability)

📖 Fully Observable vs Partially Observable

- **Fully Observable**: عامل در هر لحظه به تمام اطلاعات مرتبط با محیط دسترسی دارد و می‌تواند کل وضعیت محیط را مشاهده کند.
- **Partially Observable**: عامل تنها بخشی از اطلاعات محیط را می‌تواند مشاهده کند و اطلاعات کامل در دسترس نیست.

✨ مثال: Pac-Man

در بازی Pac-Man، محیط **Partially Observable** است زیرا:

- Pac-Man نمی‌داند Power Pellet چه مدت زمانی فعال خواهد بود (Duration)
- این اطلاعات برای ناظر بیرونی قابل مشاهده است اما خود Pac-Man از آن آگاه نیست
- عدم آگاهی از این اطلاعات بر تصمیم‌گیری عامل تأثیر می‌گذارد

نقاط کور (Blind Spots)

در مثال رانندگی، نقاط کوری وجود دارند که راننده نمی‌تواند آن‌ها را ببیند. این باعث می‌شود محیط Partially Observable باشد. برای جبران این کمبود، عامل نیاز به **حافظه (Memory)** دارد تا بتواند اطلاعات گذشته را ذخیره و از آن‌ها استفاده کند.

2. تعداد عامل‌ها (Number of Agents)

Single Agent vs Multi Agent

- **Single Agent**: تنها یک عامل در محیط فعالیت می‌کند
- **Multi Agent**: حداقل دو عامل در محیط وجود دارند که می‌توانند با یکدیگر تعامل داشته باشند

مثال: مرتب کردن صندلی‌های کلاس

فرض کنید یک عامل مسئول مرتب کردن صندلی‌های کلاس است:

حالت Single Agent:

- عامل به تنهایی صندلی‌ها را مرتب می‌کند
- هر شب صندلی‌ها را در کلاس‌های A، B، C می‌چیند
- فردا صبح کلاس‌ها مرتب هستند

حالت Multi Agent:

- عامل دیگری مسئول چیدن صندلی‌ها در تالار برای آزمون است
- عامل اول صندلی‌ها را از کلاس‌ها به تالار می‌برد
- عامل دوم صندلی‌ها را از تالار به کلاس‌ها می‌برد
- هر دو عامل بر تصمیمات و عملکرد یکدیگر تأثیر می‌گذارند
- نمی‌توان عملکرد هر عامل را مستقل از دیگری سنجید

انواع تعاملات در محیط Multi Agent

در محیط‌های چند عاملی، عامل‌ها می‌توانند:

- **همکاری** کنند (Cooperative)
- **رقابت** داشته باشند (Competitive)
- **تقابل** داشته باشند (Adversarial)
- **ناآگاهانه** بر یکدیگر تأثیر بگذارند

نوع تعامل تعیین می‌کند که عامل چگونه باید برنامه‌ریزی شود و چگونه باید با دیگران تعامل کند.

3. قطعیت (Determinism)

🕮 Deterministic vs Stochastic

- **Deterministic**: نتیجه هر عمل کاملاً مشخص و قابل پیش‌بینی است. اگر عامل در وضعیت S_1 عمل A را انجام دهد، همیشه به وضعیت S_2 می‌رسد.
- **Stochastic**: نتیجه اعمال احتمالی است و قطعیت ندارد. انجام یک عمل در یک وضعیت می‌تواند به وضعیت‌های مختلفی منجر شود.

✨ مثال: Vacuum Cleaner

در محیط جاروبرقی:

- اگر عمل "راست برو" همیشه عامل را به سمت راست ببرد → **Deterministic**
- اگر عمل "راست برو" گاهی عامل را به راست و گاهی به چپ ببرد → **Stochastic**

💡 *درجات Stochasticity*

محیط‌ها می‌توانند درجات مختلفی از تصادفی بودن داشته باشند:

- **مثال چمدان سقوط از هواپیما**: احتمال یک در چند میلیون که یک چمدون از هواپیما بیفتد و به عابر پیاده بخورد. این یک رویداد بسیار نادر است اما محیط را تا حدی Stochastic می‌کند.
- در طراحی عامل باید تصمیم گرفت که آیا چنین رویدادهای نادری را در نظر بگیریم یا خیر.

🔥 تفاوت Stochastic و Multi Agent

- **Stochastic** به پیامد اعمال خود عامل مربوط می‌شود
- **Multi Agent** به تأثیر عامل‌های دیگر بر محیط مربوط می‌شود
- مثلاً در رانندگی، اگر ماشین دیگری از خلاف جهت بیاید، این مربوط به Multi Agent است نه Stochastic بودن اعمال ما

4. پویایی محیط (Dynamics)

🕮 Static vs Dynamic

- **Static**: محیط در حین تصمیم‌گیری و اقدام عامل تغییر نمی‌کند
- **Dynamic**: محیط می‌تواند در حین تصمیم‌گیری و اقدام عامل تغییر کند

✨ مثال: عبور از رودخانه

محیط Static:

- تخته‌سنگ‌هایی در رودخانه وجود دارند
- عامل محاسبه می‌کند: قدم اول 20 سانتی‌متر، بعد پرش یک متری، سپس 50 سانتی‌متر
- تخته‌سنگ‌ها ثابت هستند و تغییر نمی‌کنند
- عامل می‌تواند هر زمان که بخواهد (الان، فردا، یک هفته دیگر) همان برنامه را اجرا کند

محیط Dynamic:

- در حین اجرای برنامه، تخته‌سنگ‌ها جابجا می‌شوند
- یک تخته‌سنگ ممکن است پایین برود و از جای دیگری بالا بیاید
- برنامه‌ریزی قبلی دیگر کارایی ندارد
- عامل باید سریعاً تصمیم بگیرد و مطابق با تغییرات محیط عمل کند

💡 *سرعت تصمیم‌گیری*

- در محیط **Static**: می‌توان زمان زیادی صرف تصمیم‌گیری کرد (Offline deliberation)
- در محیط **Dynamic**: باید سریعاً تصمیم گرفت و قبل از تغییر محیط اقدام کرد
- درجه هوشمندی عامل به تطابق با درجه پویایی محیط بستگی دارد

✨ مثال: Pac-Man

Pac-Man در محیط **Dynamic** قرار دارد زیرا:

- ارواح (Ghosts) در حال حرکت هستند
- محیط در حین بازی تغییر می‌کند
- عامل نمی‌تواند زمان نامحدود برای تصمیم‌گیری صرف کند

5. گسسته یا پیوسته بودن (Discrete vs Continuous)

📖 Discrete vs Continuous

- **Discrete**: محیط دارای تعداد محدود و قابل شمارش از حالت‌ها و اعمال است
- **Continuous**: محیط دارای حالت‌ها و اعمال پیوسته است

نکته مهم

تفاوت بین Discrete و Continuous کاملاً واضح نیست زیرا:

- در محاسبات، محیط‌های Continuous را با گسسته‌سازی (Discretization) پیاده‌سازی می‌کنیم
- تفاوت در اندازه گام‌های گسسته‌سازی است
- هر چه گام‌ها ریزتر باشند، به محیط Continuous نزدیک‌تر می‌شویم

مثال‌ها

Discrete:

- شطرنج: از خانه‌ای به خانه دیگر حرکت می‌کنیم
- تخته نرد: تعداد محدودی حالت و حرکت وجود دارد

Continuous:

- تاکسی‌درایور: پیوسته روی سطح حرکت می‌کند
- توجه: حتی تاکسی‌درایور را هم در محاسبات با Discrete بسیار ریز پیاده‌سازی می‌کنیم

Pac-Man: Discrete یا Continuous؟

- از نظر **Location**: مختصات دارد (1,2) → (2,2) → (2,3) پس **Discrete** است
- اگر نقاط را بسیار نزدیک به هم بگیریم می‌توان **Continuous** در نظر گرفت
- معمولاً به دلیل مدل‌سازی مبتنی بر مختصات، **Discrete** در نظر گرفته می‌شود

6. شناخت فیزیک مسئله (Known vs Unknown Physics)

Known vs Unknown Physics

- **Known Physics**: عامل از ساختار محیط، تعداد اتاق‌ها، شکل لابیرنت و... آگاه است
- **Unknown Physics**: عامل باید ساختار محیط را کشف کند

✨ مثال: Vacuum Cleaner

Known Physics:

- عامل می‌داند دو اتاق A و B وجود دارد
- می‌داند هر اتاق می‌تواند Clean یا Dirty باشد

Unknown Physics:

- عامل نمی‌داند چند اتاق وجود دارد
- باید با کاوش (Exploration) تعداد اتاق‌ها را کشف کند

💡 *سطوح مختلف Unknown*

عامل می‌تواند نسبت به موارد مختلفی ناآگاه باشد:

1. **Unknown Environment Structure**: نمی‌داند محیط چند اتاق دارد یا لابیرنت چه شکلی است
2. **Unknown Actions**: حتی نمی‌داند چه اعمالی می‌تواند انجام دهد
   - مثل کودکی که یاد می‌گیرد چطور بخزد، دست و پا بزند
   - عامل یکی یکی اعمال خود را کشف می‌کند
3. **Unknown Performance Measure**: نمی‌داند بر اساس چه معیاری ارزیابی می‌شود
   - مثلاً تاکسی‌درایور نمی‌داند آیا باید سفر هیجان‌انگیز باشد یا آرام
   - آیا هزینه سوخت مهم است یا زمان رسیدن؟

🔥 یادگیری تقویتی (Reinforcement Learning)

در محیط‌هایی که Unknown هستند، عامل می‌تواند از طریق **یادگیری تقویتی** موارد زیر را کشف کند:

- اعمال خود و نتایج آن‌ها
- ساختار محیط
- معیار ارزیابی عملکرد

7. سوال دانشجو: آیا Discrete/Continuous بودن بخشی از فیزیک محیط نیست؟

💡 *پاسخ استاد*

- فیزیک مسئله بیشتر به **ساختار (Structure)** محیط برمی‌گردد
- بودن در ابعاد اصلی قرار دارد نه در فیزیک Continuous یا Discrete
- البته ممکن است عامل بتواند این را هم کشف کند (مثلاً کشف کند محیط Dynamic است یا Static)
- اما معمولاً این ویژگی‌ها جزو تعریف اولیه محیط هستند

جدول طبقه‌بندی محیط‌های نمونه

بیایید چند محیط نمونه را بر اساس ابعاد بالا طبقه‌بندی کنیم:

Pac-Man

بُعد	مقدار	توضیح
Observability	Partially Observable	آگاه نیست Power Pellet فعال بودن Duration از Pac-Man
Number of Agents	Multi Agent	ارواح (Ghosts) نیز در محیط حضور دارند
Determinism	Deterministic	اعمال نتایج مشخصی دارند
Dynamics	Dynamic	ارواح در حال حرکت هستند و محیط تغییر می‌کند
Discrete/Continuous	Discrete	حرکت بر اساس مختصات و از خانه به خانه است

نکته درباره Multi Agent بودن Pac-Man

- اگر ارواح ثابت بودند و حرکت نمی‌کردند، می‌توانستیم آن‌ها را بخشی از محیط در نظر بگیریم
- اما چون ارواح حرکت می‌کنند و رفتار دارند، محیط Multi Agent است
- شامل هر آنچه به غیر از خود عامل است، از جمله عامل‌های دیگر Environment

تاکسی‌درایور (Taxi Driver)

بُعد	مقدار	توضیح
Observability	Partially Observable	نقاط کور وجود دارد، نمی‌تواند همه جا را ببیند
Number of Agents	Multi Agent	رانندگان دیگر در محیط حضور دارند
Determinism	Deterministic (معمولاً)	اگر همه معقول رانندگی کنند، اعمال نتایج مشخصی دارند
Dynamics	Dynamic	همه چیز در حال تغییر است
Discrete/Continuous	Continuous	پیوسته روی سطح حرکت می‌کند

درجات Stochasticity در رانندگی

- اگر همه قوانین را رعایت کنند: **Deterministic**
- اگر برخی از خلاف جهت بیایند یا رفتار غیرمنتظره داشته باشند: **Stochastic**
- در برخی شهرها به معلم‌های رانندگی می‌گویند حواست به بالای سرت هم باشد!
- مثال چمدان سقوط از هواپیما: احتمال بسیار کم اما وجود دارد

تشخیص بیماری (Medical Diagnosis)

بُعد	مقدار	توضیح
Observability	Partially Observable	تمام اطلاعات بیمار در دسترس نیست
Number of Agents	Single Agent (معمولاً)	سیستم تشخیص به تنهایی کار می‌کند
Determinism	Stochastic	نتایج آزمایش‌ها و تشخیص‌ها احتمالی است
Dynamics	Static (نسبتاً)	در حین تشخیص، وضعیت بیمار تغییر چندانی نمی‌کند

توضیح	مقدار	بُعد
برخی پارامترها گسسته و برخی پیوسته هستند	Mixed	Discrete/Continuous

تمرین

این طبقه‌بندی‌ها را می‌توانید برای محیط‌های مختلف تمرین کنید. در یکی از پروژه‌های درس، چیزی مشابه این خواهد بود.

نیازمندی‌های عامل بر اساس نوع محیط

هر نوع محیط، نیازمندی‌های خاصی را برای طراحی عامل ایجاد می‌کند:

1. محیط Partially Observable → نیاز به حافظه (Memory)

Internal State

وقتی محیط Partially Observable است، عامل نیاز به **حافظه** یا **Internal State** دارد تا بتواند:

- اطلاعات گذشته را ذخیره کند
- از اطلاعات چند لحظه قبل استفاده کند
- محیط را مدل‌سازی کند

مثال: نقاط کور در رانندگی

> لحظه t-1: ماشین قرمز پشت سر من است (در آینه می‌بینم)
> لحظه t: ماشین قرمز در هیچ آینه‌ای نیست

بدون حافظه: نمی‌دانم ماشین کجاست
با حافظه: می‌دانم ماشین در نقطه کور است، نباید لاین عوض کنم

هزینه حافظه

- هر رجیستر یا چیپست اضافی، عامل را گران‌تر می‌کند
- باید تعادل بین هزینه و عملکرد برقرار شود
- هر چه بیشتر از گذشته را ذخیره کنیم، عامل هوشمندتر اما گران‌تر می‌شود

و حافظه Fully Observable

در محیط Fully Observable نیازی به حافظه نیست زیرا:

- عامل در هر لحظه تمام اطلاعات لازم را دارد
- نیازی به استدلال بر اساس گذشته نیست
- البته عامل از قبل گران است چون سنسورهای زیادی دارد

2. محیط Stochastic → نیاز به پیش‌بینی احتمالاتی

Probabilistic Reasoning

عامل باید مجهز به توانایی **پیش‌بینی احتمالاتی** باشد:

- محاسبه احتمال وقوع رویدادها
- تصمیم‌گیری بر اساس احتمالات
- مدیریت عدم قطعیت

مثال

اگر عامل به این توانمندی مجهز نباشد:

- در محیط Stochastic به صورت ساده (Naive) عمل می‌کند
- نمی‌تواند عملکرد خوبی داشته باشد
- نمی‌تواند با عدم قطعیت کنار بیاید

3. محیط Multi Agent → نیاز به تصادفی‌سازی یا مدل‌سازی حریف

ساده‌ترین رویکرد: Randomly

در ساده‌ترین حالت، عامل باید **به صورت تصادفی (Randomly)** عمل کند:

- بهتر از تصمیمات کاملاً Deterministic است
- چرا؟ زیرا تصادفی بودن **پوشش (Coverage)** خوبی می‌دهد
- اگر زمان کافی داشته باشیم، می‌تواند به صورت کامل فضا را پوشش دهد
- دست‌برقضا ممکن است نتایج درستی به دست آید

Opponent Modeling :رویکرد پیشرفته

در سیستم‌های چند عاملی پیشرفته:

- عامل باید بتواند **عامل‌های پیرامون را مدل کند**
- ببیند رقیبان چه کاری می‌کنند
- بر اساس رفتار آن‌ها تصمیم بگیرد
- تشخیص دهد آیا رقیبان هوشمند هستند یا تصادفی

مثال: مرتب کردن صندلی‌ها

- اگر عامل دیگر دارد صندلی‌ها را می‌برد، باید سرعت خود را افزایش دهیم
- اگر آن عامل دو صندلی می‌برد، ما باید چهار صندلی بیاوریم
- باید رفتار عامل دیگر را مدل کنیم و بر اساس آن عمل کنیم

4. محیط Static → امکان Deliberation طولانی‌مدت

Offline Planning

در محیط Static:

- می‌توان زمان زیادی صرف تصمیم‌گیری کرد
- الگوریتم‌های پیچیده با کامپلکسیتی بالا قابل استفاده هستند
- محیط تغییر نمی‌کند، پس عجله‌ای نیست
- می‌توان تصمیم بهینه‌تری گرفت

مثال

اگر محیط Static است:

- می‌توانیم الان، فردا، یا یک هفته دیگر تصمیم بگیریم
- نتیجه یکسان خواهد بود
- پس بهتر است زمان بیشتری صرف کنیم و تصمیم بهتری بگیریم

5. محیط Dynamic → نیاز به تصمیم‌گیری سریع

Real-time Decision Making

در محیط Dynamic:

- نمی‌توان به Deliberation طولانی‌مدت متکی بود
- باید در سریع‌ترین زمان ممکن تصمیم گرفت
- قبل از تغییر محیط باید اقدام کرد
- یا باید پیش‌بینی کرد محیط چگونه تغییر خواهد کرد

6. محیط Continuous → نیاز به کنترل پیوسته

Continuous Control

در محیط Continuous:

- باید همه چیز را تنظیم کنیم
- ارتفاع، فشار، دما و... را پیوسته رصد کنیم
- مطابق با تغییرات پیوسته عمل کنیم
- نیاز به سیستم‌های کنترل پیوسته داریم

7. محیط Unknown → نیاز به اکتشاف (Exploration)

Exploration and Learning

در محیط Unknown:

- عامل باید **اکتشاف (Explore)** کند
- با انجام اعمال، محیط را بشناسد
- فیزیک محیط را کشف کند
- اعمال خود را یاد بگیرد
- معیار ارزیابی را درک کند

✨ مثال: کشف فیزیک محیط

```
عامل به سمت راست می‌رود: یک قدم، دو قدم، سه قدم، چهار قدم
→ می‌خورد به دیوار (Bump)

عامل به سمت چپ می‌رود: سریع می‌خورد به دیوار

نتیجه: سمت راست فضای بیشتری دارد، سمت چپ دیوار نزدیک‌تر است
```

یادگیری اعمال

عامل می‌تواند اعمال خود را نیز یاد بگیرد:

- مثل کودکی که یاد می‌گیرد چطور بخزد
- یکی یکی دست و پا و توانمندی‌های خود را کشف می‌کند
- می‌فهمد شش اکشن دارد یا ده اکشن یا هزار اکشن

یادگیری Performance Measure

حتی معیار ارزیابی را هم می‌تواند یاد بگیرد:

- از طریق **تعامل با یک موجود آگاه‌تر** (Supervisor)
- سوپروایزر می‌تواند انسان، عامل دیگر، یا یک مدل باشد
- مشابه **Supervised Learning** در یادگیری ماشین
- از طریق تنبیه و تشویق یاد می‌گیرد چه کاری درست است

یادگیری تقویتی (Reinforcement Learning)

Reinforcement Learning

یادگیری تقویتی (یا تشویقی) روشی است که عامل از طریق **تشویق و تنبیه** یاد می‌گیرد:

- عامل اعمالی انجام می‌دهد
- برای اعمال خوب پاداش (Reward) یا جریمه (Penalty) دریافت می‌کند
- بر اساس این بازخوردها، رفتار خود را بهبود می‌دهد
- به تدریج یاد می‌گیرد کدام اعمال به نتایج بهتری منجر می‌شوند

کاربرد Reinforcement Learning در محیط‌های Unknown

یادگیری تقویتی به ویژه در محیط‌هایی که Unknown هستند کاربرد دارد:

- عامل نمی‌داند محیط چگونه است
- نمی‌داند چه اعمالی می‌تواند انجام دهد
- نمی‌داند معیار ارزیابی چیست
- از طریق تجربه و دریافت پاداش/جریمه، همه این‌ها را یاد می‌گیرد

✨ مثال: یادگیری راه رفتن

کودک تلاش می‌کند راه برود:

- دست و پا می‌زند → می‌افتد → تنبیه (درد)
- حرکت دیگری انجام می‌دهد → یک قدم برمی‌دارد → تشویق (موفقیت)
- به تدریج یاد می‌گیرد کدام حرکات به راه رفتن منجر می‌شوند

Exploration vs Exploitation

در یادگیری تقویتی، عامل باید تعادل بین دو استراتژی برقرار کند:

Exploration (اکتشاف):

- امتحان کردن اعمال جدید
- کشف محیط و امکانات جدید
- ریسک کردن برای یادگیری بیشتر

Exploitation (بهره‌برداری):

- استفاده از دانش موجود
- انجام اعمالی که می‌دانیم خوب هستند
- به حداکثر رساندن پاداش فعلی

نکته مهم

- اگر فقط Explore کنیم، هیچ‌وقت از دانش خود استفاده نمی‌کنیم
- اگر فقط Exploit کنیم، ممکن است راه‌حل‌های بهتری را از دست بدهیم
- باید تعادل مناسبی بین این دو برقرار شود

حافظه (Memory) و Internal State

Internal State

Internal State یا حالت درونی عامل، اطلاعاتی است که عامل از گذشته نگه می‌دارد و برای تصمیم‌گیری‌های آینده استفاده می‌کند.

چرا به حافظه نیاز داریم؟

نیاز به حافظه در Partially Observable

در محیط‌های Partially Observable:

- عامل نمی‌تواند همه چیز را ببیند
- باید اطلاعات گذشته را ذخیره کند
- از این اطلاعات برای استنتاج درباره حال استفاده کند

مثال: رانندگی و نقاط کور

```
t-2: ماشین قرمز در خط چپ است
t-1: ماشین قرمز پشت سر من است (در آینه می‌بینم)
t: ماشین قرمز در هیچ آینه‌ای نیست

استنتاج با حافظه:
- ماشین قرمز احتمالاً در نقطه کور سمت چپ است
- نباید الان لاین عوض کنم
- باید صبر کنم تا ماشین از نقطه کور خارج شود
```

هزینه حافظه

Trade-off بین عملکرد و هزینه

حافظه مزایا و هزینه‌هایی دارد:

مزایا:

- تصمیم‌گیری بهتر
- درک بهتر از محیط
- عملکرد بالاتر

هزینه‌ها:

- نیاز به سخت‌افزار بیشتر (رجیستر، چیپست)
- مصرف انرژی بیشتر
- پیچیدگی بیشتر در طراحی
- قیمت بالاتر

✨ مثال: طراحی ربات جاروبرقی

```
نسخه ارزان:
- حافظه کم
- فقط وضعیت فعلی را می‌بیند
- ممکن است چند بار از یک جا عبور کند

نسخه گران:
- حافظه زیاد
- نقشه محیط را ذخیره می‌کند
- می‌داند کجاها را تمیز کرده
- مسیر بهینه‌تری طی می‌کند
```

🔥 سوال طراحی

در طراحی عامل باید پرسید:

- چقدر حافظه واقعاً لازم است؟
- آیا هزینه اضافی حافظه را می‌ارزد؟
- چه مقدار از گذشته باید ذخیره شود؟

حافظه در Fully Observable

💡 *آیا در Fully Observable به حافظه نیاز داریم؟*

پاسخ: نه، اما...

در محیط Fully Observable:

- عامل در هر لحظه تمام اطلاعات لازم را دارد
- نیازی به استدلال بر اساس گذشته نیست
- می‌تواند فقط بر اساس وضعیت فعلی تصمیم بگیرد

اما:

- عامل از قبل گران است
- چون باید سنسورهای زیادی داشته باشد
- تا بتواند همه چیز را ببیند

مقایسه

```
Partially Observable + Memory:
- سنسورهای کم → ارزان
- حافظه زیاد → هزینه اضافی
- جمع: متوسط

Fully Observable:
- سنسورهای زیاد → گران
- حافظه کم → صرفه‌جویی
- جمع: گران
```

مدل‌سازی حریف (Opponent Modeling)

Opponent Modeling

مدل‌سازی حریف به معنای درک و پیش‌بینی رفتار عامل‌های دیگر در محیط است.

چرا به Opponent Modeling نیاز داریم؟

محیط‌های Multi Agent

در محیط‌های چند عاملی:

- عامل‌های دیگر بر محیط تأثیر می‌گذارند
- نمی‌توان آن‌ها را نادیده گرفت
- باید رفتار آن‌ها را پیش‌بینی کرد
- بر اساس رفتار آن‌ها تصمیم گرفت

سطوح مختلف Opponent Modeling

سطح ۱: تصادفی (Random)

ساده‌ترین رویکرد:

- عامل به صورت تصادفی عمل می‌کند
- هیچ مدلی از حریف ندارد
- فقط امیدوار است که با تصادفی بودن، Coverage خوبی داشته باشد

مزیت:

- ساده است
- اگر زمان کافی باشد، می‌تواند کل فضا را پوشش دهد

معایب:

- کارایی پایین
- نمی‌تواند از رفتار حریف استفاده کند

سطح ۲: مدل ساده

عامل یک مدل ساده از حریف دارد:

- فرض می‌کند حریف به صورت خاصی عمل می‌کند
- مثلاً فرض می‌کند حریف همیشه نزدیک‌ترین هدف را انتخاب می‌کند
- بر اساس این فرض تصمیم می‌گیرد

سطح ۳: یادگیری رفتار حریف

عامل رفتار حریف را یاد می‌گیرد:

- رفتار گذشته حریف را مشاهده می‌کند
- الگوهای رفتاری را شناسایی می‌کند
- پیش‌بینی می‌کند حریف بعداً چه خواهد کرد
- بر اساس این پیش‌بینی برنامه‌ریزی می‌کند

✨ مثال: بازی شطرنج

```
سطح ۱ (Random):
- حرکات تصادفی انجام می‌دهیم
- هیچ استراتژی خاصی نداریم

سطح ۲ (مدل ساده):
- فرض می‌کنیم حریف همیشه مهره‌های ما را می‌خورد
- مهره‌های خود را محافظت می‌کنیم

سطح ۳ (یادگیری):
- متوجه می‌شویم حریف سبک تهاجمی دارد
- استراتژی دفاعی اتخاذ می‌کنیم
- از اشتباهات تکراری حریف استفاده می‌کنیم
```

Opponent Modeling در Pac-Man

در بازی Pac-Man:

- ارواح (Ghosts) الگوهای حرکتی خاصی دارند
- برخی Pac-Man را تعقیب می‌کنند
- برخی سعی می‌کنند راه او را ببندند
- اگر Pac-Man این الگوها را بشناسد، می‌تواند بهتر فرار کند

یادگیری تقویتی و Opponent Modeling

ترکیب دو مفهوم

می‌توان یادگیری تقویتی و مدل‌سازی حریف را ترکیب کرد:

- عامل از طریق تعامل با حریف یاد می‌گیرد
- رفتار حریف را مدل می‌کند
- استراتژی خود را بر اساس مدل حریف تنظیم می‌کند
- به تدریج عملکرد خود را بهبود می‌دهد

✨ مثال: بازی‌های ویدیویی

در بازی‌های مدرن:

- دشمنان هوش مصنوعی دارند
- رفتار بازیکن را یاد می‌گیرند
- اگر بازیکن همیشه از یک استراتژی استفاده کند، دشمنان یاد می‌گیرند
- بازیکن مجبور است استراتژی خود را تغییر دهد

محدودیت‌های عقلانیت (Bounded Rationality)

📖 Bounded Rationality

عقلانیت محدود به این معناست که عامل نمی‌تواند کاملاً بهینه عمل کند زیرا:

- منابع محاسباتی محدود دارد
- زمان محدود دارد
- اطلاعات محدود دارد
- حافظه محدود دارد

💡 *تفاوت Rational و Optimal*

- **Rational**: با توجه به محدودیت‌ها، بهترین تصمیم را می‌گیرد
- **Optimal**: بهترین تصمیم ممکن را می‌گیرد (بدون در نظر گرفتن محدودیت‌ها)

عامل Rational لزوماً Optimal نیست، اما با توجه به محدودیت‌هایش، بهترین کار را انجام می‌دهد.

✨ مثال: شطرنج

```
عامل Optimal:
- تمام حالت‌های ممکن را بررسی می‌کند
- بهترین حرکت را پیدا می‌کند
- نیاز به زمان و حافظه نامحدود دارد
- در عمل غیرممکن است

عامل Rational:
- تعداد محدودی حالت را بررسی می‌کند
- در زمان معقول تصمیم می‌گیرد
- ممکن است بهترین حرکت را پیدا نکند
- اما با توجه به محدودیت‌ها، خوب عمل می‌کند
```

محدودیت‌های رایج

عامل‌ها معمولاً با این محدودیت‌ها مواجه هستند:

1. **محدودیت زمانی**: باید در زمان معین تصمیم بگیرند
2. **محدودیت حافظه**: نمی‌توانند همه چیز را ذخیره کنند
3. **محدودیت محاسباتی**: نمی‌توانند همه حالت‌ها را بررسی کنند
4. **محدودیت اطلاعاتی**: همه اطلاعات در دسترس نیست
5. **محدودیت انرژی**: باید مصرف انرژی را مدیریت کنند

طراحی با در نظر گرفتن محدودیت‌ها

در طراحی عامل باید:

- محدودیت‌های واقعی را شناسایی کنیم
- الگوریتم‌هایی طراحی کنیم که با این محدودیت‌ها کار کنند
- تعادل بین کیفیت تصمیم و سرعت/هزینه برقرار کنیم
- از تکنیک‌هایی مثل Heuristics استفاده کنیم

خودمختاری (Autonomy)

Autonomy

خودمختاری به معنای توانایی عامل برای عمل کردن بدون دخالت مستقیم انسان یا سیستم خارجی است.

سطوح مختلف خودمختاری

عامل‌ها می‌توانند سطوح مختلفی از خودمختاری داشته باشند:

سطح ۱: کاملاً وابسته

- عامل فقط دستورات را اجرا می‌کند
- هیچ تصمیمی خودش نمی‌گیرد
- مثل یک ماشین حساب

سطح ۲: نیمه خودمخت

- عامل برخی تصمیمات را خودش می‌گیرد
- در موارد مهم از انسان کمک می‌گیرد
- مثل سیستم‌های کمک راننده

سطح ۳: کاملاً خودمخت

- عامل تمام تصمیمات را خودش می‌گیرد
- بدون دخالت انسان عمل می‌کند
- مثل ربات‌های کاوشگر فضایی

✨ مثال: خودروهای خودران

```
سطح ۰ : راننده انسانی
- انسان همه کارها را انجام می‌دهد

سطح ۱ : کمک راننده
- سیستم هشدار می‌دهد
- انسان تصمیم می‌گیرد

سطح ۲: خودکار جزئی
- سیستم می‌تواند ترمز کند یا فرمان بچرخاند
- انسان باید نظارت کند

سطح ۳: خودکار شرطی
- سیستم در شرایط خاص (مثلاً اتوبان) خودش رانندگی می‌کند
- انسان باید آماده دخالت باشد

سطح ۴: خودکار بالا
- سیستم در اکثر شرایط خودش رانندگی می‌کند
- انسان می‌تواند کاملاً بی‌توجه باشد

سطح ۵: کاملاً خودکار
- سیستم در تمام شرایط خودش رانندگی می‌کند
- نیازی به راننده انسانی نیست
```

یادگیری و خودمختاری

خودمختاری ارتباط نزدیکی با یادگیری دارد:

- عامل خودمخت باید بتواند از تجربه یاد بگیرد
- باید بتواند با موقعیت‌های جدید کنار بیاید
- باید بتواند خودش را با محیط تطبیق دهد

ملاحظات اخلاقی (Ethics)

📖 Ethics in AI

اخلاق در هوش مصنوعی به مجموعه اصول و ارزش‌هایی گفته می‌شود که باید در طراحی و استفاده از عامل‌های هوشمند رعایت شوند.

چرا اخلاق مهم است؟

عامل‌های هوشمند می‌توانند:

- بر زندگی انسان‌ها تأثیر بگذارند
- تصمیماتی بگیرند که پیامدهای اجتماعی دارد
- به صورت ناخواسته به افراد آسیب برسانند
- تبعیض ایجاد کنند
- حریم خصوصی را نقض کنند

Performance Measure و اخلاق

هنگام تعریف Performance Measure باید به موارد زیر توجه کرد:

- **نه فقط سود خود عامل**: نباید فقط به سود یک عامل فکر کنیم
- **تأثیر بر دیگران**: باید تأثیر بر سایر عامل‌ها و انسان‌ها را در نظر بگیریم
- **پیامدهای بلندمدت**: نه فقط سود کوتاه‌مدت، بلکه پیامدهای بلندمدت را هم ببینیم
- **عدالت**: آیا عامل به صورت عادلانه با همه رفتار می‌کند؟

مثال: تاکسی‌درایور

```
Performance Measure نادرست:
- فقط سود راننده را در نظر بگیریم
- هر چه سریع‌تر برسد، بهتر
- نتیجه: رانندگی خطرناک، تصادف، آسیب به دیگران

Performance Measure اخلاقی:
- سود راننده
- ایمنی مسافر
- ایمنی سایر رانندگان و عابران
- رعایت قوانین
- مصرف سوخت و آلودگی محیط زیست
```

مسائل اخلاقی رایج

۱. تبعیض (Bias)

- عامل ممکن است بر اساس نژاد، جنسیت، سن و... تبعیض قائل شود
- این تبعیض معمولاً از داده‌های آموزشی می‌آید

۲. حریم خصوصی (Privacy)

- عامل چقدر اطلاعات جمع‌آوری می‌کند؟
- این اطلاعات چگونه استفاده می‌شوند؟

۳. شفافیت (Transparency)

- آیا می‌توان فهمید عامل چگونه تصمیم می‌گیرد؟
- آیا می‌توان تصمیمات را توجیه کرد؟

۴. مسئولیت (Accountability)

- اگر عامل اشتباه کند، چه کسی مسئول است؟
- چگونه می‌توان عامل را پاسخگو کرد؟

۵. ایمنی (Safety)

- آیا عامل می‌تواند به انسان‌ها آسیب برساند؟
- چگونه از سوءاستفاده جلوگیری کنیم؟

مثال: سیستم تشخیص پزشکی

```
مسائل اخلاقی:
- اگر سیستم اشتباه تشخیص دهد، چه کسی مسئول است؟
- آیا سیستم برای همه گروه‌های سنی/نژادی به یک اندازه دقیق است؟
- آیا اطلاعات پزشکی بیماران محفوظ است؟
- آیا پزشک می‌تواند تصمیم سیستم را توجیه کند؟
- آیا بیمار حق دارد بداند چرا این تشخیص داده شده؟
```

اصول طراحی اخلاقی

در طراحی عامل‌های هوشمند:

1. **شفافیت**: تصمیمات باید قابل توضیح باشند
2. **عدالت**: عامل نباید تبعیض قائل شود
3. **حریم خصوصی**: اطلاعات کاربران باید محفوظ باشد
4. **ایمنی**: عامل نباید به انسان‌ها آسیب برساند
5. **مسئولیت‌پذیری**: باید مشخص باشد چه کسی مسئول است
6. **کنترل انسانی**: انسان باید بتواند در صورت نیاز دخالت کند

خلاصه و جمع‌بندی

نکات کلیدی این جلسه

۱. ابعاد محیط:

- Observability: Fully vs Partially Observable
- Number of Agents: Single vs Multi Agent
- Determinism: Deterministic vs Stochastic
- Dynamics: Static vs Dynamic
- Discrete/Continuous
- Known/Unknown Physics

۲. نیازمندی‌های عامل:

- Partially Observable → Memory
- Stochastic → Probabilistic Reasoning
- Multi Agent → Opponent Modeling
- Dynamic → Fast Decision Making
- Unknown → Exploration & Learning

۳. مفاهیم پیشرفته:

- Reinforcement Learning
- Bounded Rationality
- Autonomy
- Ethics

۴. نکته مهم:

- هر محیط نیازمندی‌های خاص خود را دارد
- باید تعادل بین عملکرد و هزینه برقرار کرد
- ملاحظات اخلاقی را نباید فراموش کرد