

بسمه تعالی

هوش مصنوعی
جستجو در محیطهای پیچیده
نیمسال اول ۱۴۰۳-۱۴۰۲

دکتر مازیار پالهنک
آزمایشگاه هوش مصنوعی
دانشکده مهندسی برق و کامپیوتر
دانشگاه صنعتی اصفهان

یادآوری

- رفع محدودیتهای جستجوهای کلاسیک
- الگوریتمهای جستجوی محلی
- حالت فعلی را نگهدار - سعی کن آن را بهبود دهی
- جستجوی تپه نوردی
- تنوعها:
 - تپه نوردی تصادفی
 - تپه نوردی اولین انتخاب
 - تپه نوردی با باز شروع تصادفی
- سرد شدن شبیه سازی شده
- جستجوی پرتوی محلی
- الگوریتم ژنتیک

جستجوی محلی در فضای پیوسته

- الگوریتمهای بیان شده تاکنون بجز جستجوی تپه نوردی اولین انتخاب و سردشدن شبیه سازی شده توان برخورد با فضای حالت پیوسته را ندارند.
- در حالت پیوسته تعداد حالات تالی بسیار زیاد است.
- فرض کنید در مثال جهانگرد، می خواهیم سه فرودگاه جدید در ایران بسازیم
- می خواهیم مجموع مربعات فواصل شهرهای آن نقشه تا نزدیکترین فرودگاه حداقل باشد.

- فضای حالت مختصات این فرودگاهها
- $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3)$
- حرکت در این فضا همانند تغییر مکان فرودگاهها
- فرض C_i مجموعه شهرهائی که نزدیکترین فرودگاه به آنها در فضای حالت فعلی، فرودگاه i ام است.
- تابع هدف:

$$f(x_1, y_1, x_2, y_2, x_3, y_3) = \sum_{i=1}^3 \sum_{c \in C_i} (x_i - x_c)^2 + (y_i - y_c)^2 .$$

- چون مجموعه C_i وابسته به حالت است، تابع هدف بصورت محلی درست است.
- یک روش گسسته سازی
- هر بار فقط یک فرودگاه بتواند در راستای X یا Y به اندازه $\pm\delta$ تغییر کند.
- هر حالت دارای ۱۲ تالی
- حال امکان استفاده از هر یک از جستجوهای محلی
- یا سرد شدن شبیه سازی شده، یا تپه نوردی اولین انتخاب بدون گسسته سازی

گرادیان

- روش دیگر استفاده از گرادیان چشم انداز (تابع هدف)
- اگر تابع $f(x_1, x_2, \dots, x_n)$ را داشته باشیم.

$$\nabla f(x_1, x_2, \dots, x_n) = \frac{\partial f}{\partial x_1} e_1 + \frac{\partial f}{\partial x_2} e_2 + \dots + \frac{\partial f}{\partial x_n} e_n$$

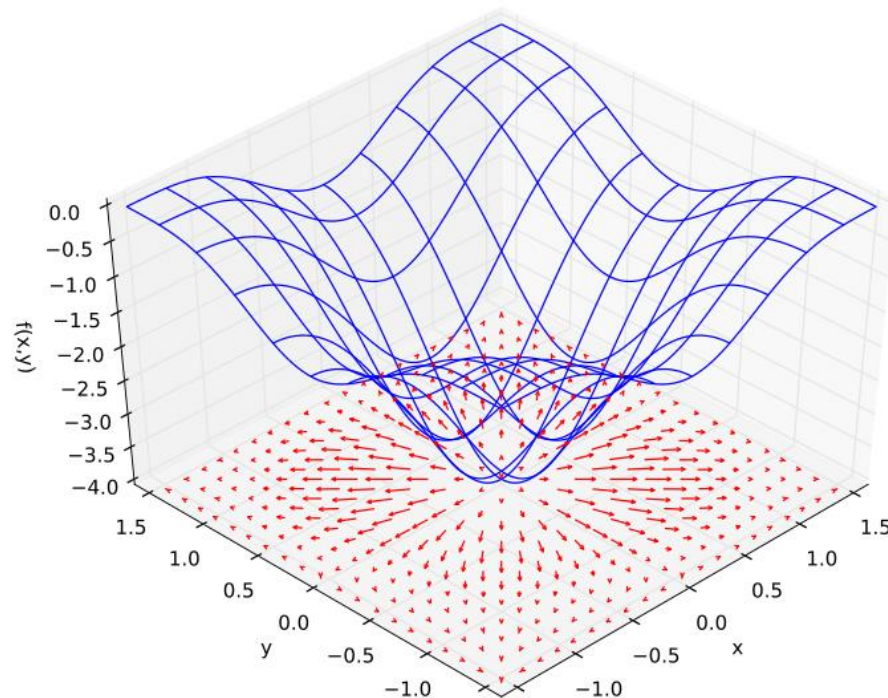
- که e_i ها بردارهای یکه هر یک از محورهای مختصات را نشان می دهند.

• بطور مثال اگر:

$$f(x, y, z) = 2x - 3y^3 + \sin(z)$$

$$\nabla f(x, y, z) = 2i - 9y^2 j + \cos(z)k$$

گرادیان



$$f(x, y) = -(\cos^2 x + \cos^2 y)^2$$

مازیار پالهنک

هوش مصنوعی - نیمسال اول ۱۴۰۲-۰۳

گرادیان

- گرادیان یک تابع، برداری که مقدار و جهت بیشترین افزایش تابع را نشان می دهد.
- برای مثال گفته شده:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$

- گاهی با حل $\nabla f=0$ می توان بیشینه را بدست آورد.
- به شرط آنکه معادله آن براحتی قابل حل باشد.
- بطور مثال اگر فقط یک فرودگاه می خواستیم بسازیم.
- در این حالت پاسخ میانگین مختصات همه شهرها بود.
- در بسیاری از موارد معادله بصورت بسته قابل حل نیست

■ روش دیگر، تغییر متغیرها به اندازه کمی در جهت گرادیان:

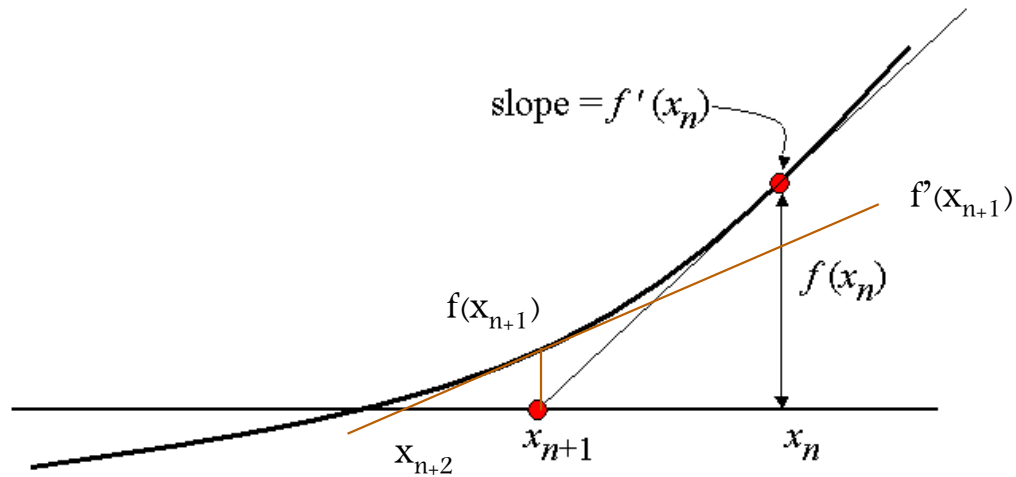
$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x}) ,$$

■ اگر مسئله کمینه سازی است، تغییر در جهت عکس گرادیان.

■ پارامتر α نرخ یادگیری

■ در صورتی که فرمول گرادیان در دسترس نباشد، از گرادیان تجربی استفاده می شود.

- در گرادیان تجربی، مقدار متغیرها اندکی کم و زیاد شده و بصورت تجربی مقداری گرادیان محاسبه می شود.
- روش دیگر، استفاده از روش نیوتن - رافسن
- روشی برای یافتن ریشه یک تابع



■ معادله خط مماس: $y = f'(x_n)(x - x_n) + f(x_n)$

■ محل برخورد با خط $y=0$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

<https://brilliant.org/wiki/newton-raphson-method/>

■ مجدداً معادله خط مماس در X_{n+1} محاسبه و X_{n+2} محاسبه می شود.

■ تکرار تا بدست آوردن ریشه

■ علاقمندیم ریشه ∇f را بدست آوریم، بنابراین بجای f ، ∇f را قرار می دهیم و خواهیم داشت:

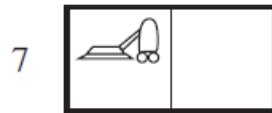
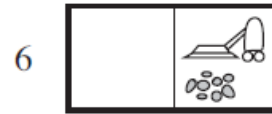
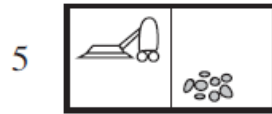
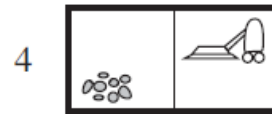
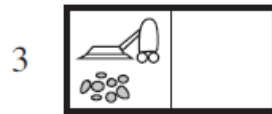
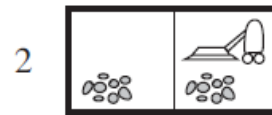
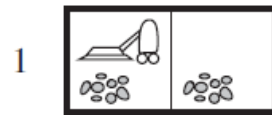
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x}) ;$$

■ $H_f(\mathbf{x})$ ماتریس هسین

$$H_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

جستجو با اعمال قطعی



■ محیط قطعی - مشاهده پذیر:

مسئله تک حالت

■ عامل دقیقاً می داند که در چه

حالتی خواهد بود.

■ حل یک دنباله

■ با شروع از ۵:

■ [راست، مکش]

جستجو با اعمال غیر قطعی

■ مثال ربات جاروی سرگردان:

■ محیط مشاهده پذیر-غیر قطعی

■ عمل مکش:

■ در خانه کثیف آن خانه را تمیز ولی گاهی خانه

همسایه را نیز تمیز می کند،

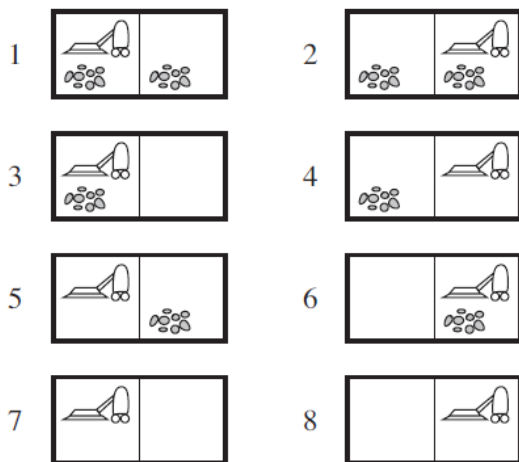
■ در خانه تمیز گاهی آشغال می ریزد.

■ تعمیم مدل انتقال در فصل ۳

■ تابع `Result()` یک مجموعه از حالات قابل

دستیابی ممکن باز می گرداند.

■ بطور مثال: مکش در حالت ۱ به مجموعه $\{۷و۵\}$ می رود.



جستجو با اعمال غیر قطعی

- اصلاح حل: بجای یک دنباله یک طرح شرطی (یا طرح اقتضائی یا راهبرد (استراتژی))
- بطور مثال با شروع از حالت ۱:
 $[Suck, \text{if } State = 5 \text{ then } [Right, Suck] \text{ else } []]$.
- حل مسائل همراه با عدم قطعیت شامل جملات اگر-آنگاه
- حل بصورت یک درخت بجای یک دنباله
- انتخاب عمل بر اساس اقتضاء بعد از عمل

درخت جستجوی AND-OR

- جستجو مجدداً با ایجاد یک درخت جستجو
- در حالت قطعی شاخه ها با اعمالی که عامل می تواند انجام دهد ایجاد می شد.
- به چنین رئوس ایجاد شده، **رئوس-یا** (OR nodes) می گوئیم.
- در حالت غیرقطعی باید همه رئوسی که ممکن است نتایج انجام عمل باشند نیز ایجاد شود.
- این رئوس، **رئوس-و** (AND nodes) گفته می شوند.

جستجو با اعمال غیر قطعی

- یک حل برای یک جستجو و-یا (AND-OR) یک زیر درخت از درخت جستجوی کامل است بطوری که:
- در هر برگ یک رأس هدف وجود دارد،
- در هر یک از رئوس یا در آن فقط یک عمل مشخص شده، و
- در هر یک از رئوس و در آن تمامی پیشامدها در نظر گرفته شده اند.

Figure 4.11

function AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*
return OR-SEARCH(*problem*, *problem*.INITIAL, [])

Figure 4.11

function AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*
 return OR-SEARCH(*problem*, *problem*.INITIAL, [])

function OR-SEARCH(*problem*, *state*, *path*) **returns** a conditional plan, or *failure*
 if *problem*.IS-GOAL(*state*) **then return** the empty plan
 if IS-CYCLE(*path*) **then return** *failure*
 for each *action* **in** *problem*.ACTIONS(*state*) **do**
 plan ← AND-SEARCH(*problem*, RESULTS(*state*, *action*), [*state*] + *path*)
 if *plan* ≠ *failure* **then return** [*action*] + *plan*
 return *failure*

چون عمق نخست است هر شاخه یا به جواب می رسد یا شکست می خورد

Figure 4.11

function AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*
 return OR-SEARCH(*problem*, *problem*.INITIAL, [])

function OR-SEARCH(*problem*, *state*, *path*) **returns** a conditional plan, or *failure*
 if *problem*.IS-GOAL(*state*) **then return** the empty plan
 if IS-CYCLE(*path*) **then return failure**
 for each *action* **in** *problem*.ACTIONS(*state*) **do**
 plan \leftarrow AND-SEARCH(*problem*, RESULTS(*state*, *action*), [*state*] + *path*)
 if *plan* \neq *failure* **then return** [*action*] + *plan*
 return failure

function AND-SEARCH(*problem*, *states*, *path*) **returns** a conditional plan, or *failure*
 for each *s_i* **in** *states* **do**
 plan_i \leftarrow OR-SEARCH(*problem*, *s_i*, *path*)
 if *plan_i* = *failure* **then return failure**
 return [if *s₁* **then** *plan₁* **else if** *s₂* **then** *plan₂* **else ... if** *s_{n-1}* **then** *plan_{n-1}* **else** *plan_n*]

An algorithm for searching AND-OR graphs generated by nondeterministic environments. A solution is a conditional plan that considers every nondeterministic outcome and makes a plan for each one.

خلاصه

- جستجوی پرتوی محلی
- جستجوی پرتوی محلی تصادفی
- الگوریتم ژنتیک
- جستجوی محلی در فضای پیوسته
- جستجو با اعمال غیرقطعی



- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائه شفاهی می باشد و به هیچ وجه یک جزوه درسی نیست و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.
- حضور فعال در کلاس دارای امتیاز است.