دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

درس هوش مصنوعی

استاد محمدحسین رهبان

تمرین دوم | جست‌وجوی محلی و جست‌وجو در فضای پیوسته | مصطفی قدیمی

سؤال ۱. Simulated Annealing

الگوریتم Simulated Annealing به این صورت عمل می‌کند که برای دماهای بالا، جست‌وجوی‌ش کاملا تصادفی است. هر چه دما را بیش‌تر کاهش بدهیم، فرآیند جست‌وجو بیش از پیش دقیق می‌شود. بنابراین، دو چیز در این فرآیند بسیار مهم است:

۱. شروع با دمای بالا (برای ارزیابی ویژگی‌های ناخالص تابع هدف) و تکامل آن به صورتی که به دمای پایین‌تر برسد.

۲. با توجه به طبیعت تصادفی بودن این الگوریتم، به‌طوری تعریف شده است که که احتمال حرکت رو به پایین (هر چند با مقدار اندک بزرگ‌تر از صفر) وجود دارد. این عامل سبب می‌شود تا احتمال گیر کردن الگوریتم در ماکسیمم‌های محلی را تا حدی از بین می‌برد. چون همیشه احتمال بیرون پریدن از آن وجود دارد.

- الف) اگر دما را به سرعت کاهش دهیم، باعث می‌شود که فرآیند بازپخت به خوبی صورت نگیرد. این عمل‌کرد باعث می‌شود تا به یک وضعیت suboptimal برسیم که می‌تواند یک ماکسیمم محلی یا یک سطح صاف باشد.
- ب) اگر دمای مثبت اولیه مقداری کوچک باشد، می‌تواند منجر به محدود شدن فضای مدل در نقطه‌ی شروع شود. اما اگر مقدار آن بزرگ باشد باعث افزایش تعداد random walkها و iterationها می‌شود.
- ج) اولین node که همان current است را برمی‌گرداند و الگوریتم به پایان می‌رسد.

سؤال ۲. Hill Climbing

- آ) 2^n
- ب) n
- ج) هدف: مجموعه‌ی همه‌ی Conjunction of Disjunctions مقدارش True شود. به‌طور دقیق‌تر برای هر State تعداد عبارت‌هایی که True (یا Satisfy) شده‌اند.
- د) در این مثال، اگر به هر کدام از پرانتزها (از چپ به راست) عددی از ۱ تا ۵ نسبت دهیم، مقداری که برای هر کدام از آن‌ها به دست می‌آید، به شرح زیر است:

$$1 \rightarrow False,\ 2 \rightarrow True,\ 3 \rightarrow True,\ 4 \rightarrow True,\ 5 \rightarrow False$$

با توجه به قسمت (ج)، هدف ما این است که مقدار همه‌ی آن‌ها را صحیح کنیم،.بنابراین به تصادف باید مقدار یکی از متغیرهای A یا D را برابر با True قرار دهیم تا به حالت بعدی برویم.

- ه) در این مثال، چون سه متغیر داریم که هر کدام دو حالت دارند، بنابراین اندازه‌ی فضای حالت برابر است با:

$$2^3 = 8$$

بهینه محلی به حالتی اطلاق می‌شود که از همسایه‌هایش بهتر باشد اما بهینه‌ی سراسری نباشد. بنابراین پس از بررسی حالت‌ها، حالت زیر یک بهینه محلی است:

$$A : False,\ B : True,\ C : True$$

سؤال ۳. Genetic Algorithm

مدل‌سازی: برای مدل کردن این مسئله با الگوریتم ژنتیک، یک رشته‌ی به طول n خواهیم داشت (که در آن n، تعداد رأس‌های گراف است). عدد «۱» در جایگاه iام، نشان‌دهنده‌ی حضور رأس i در مجموعه‌ی Vertex Cover و عدد «۰» به منزله‌ی عدم حضور آن است.

تابع fitness: تفاضل تعداد کل رأس‌ها از تعداد رأس‌هایی که عضو Vertex Cover نیستند و مقدار «۱» دارند.

تابع selection: همانند آن‌چه که در اسلاید شماره «۵» درس برای n-Queens داشتیم، چون در این‌جا با رشته‌ها کار می‌کنیم، یک عدد در محدوده‌ی ۱ تا $n-1$ انتخاب می‌کنیم و نام آن را x می‌گذاریم. با توجه به اعدادی (درصدهایی) که به کمک تابع fitness به دست آوردیم، دو به دو دسته‌بندی می‌کنیم.

تابع cross-over: همانند آن‌چه که در اسلاید شماره «۵» درس برای n-Queens داشتیم، از ۱ تا x کاراکتر از رشته‌ی اول را $x+1$ تا n رشته‌ی دوم را جابه‌جا می‌کنیم. دو رشته‌ی جدید از ترکیب به دست می‌آید.

تابع mutation: همانند آن‌چه که در اسلاید شماره «۵» درس برای n-Queens داشتیم، یکی از کاراکترها را تصادفی تغییر می‌دهیم.

سؤال ۴. **Local Search**

- آ) ابتدا یک زیرمجموعه تصادفی از S انتخاب می‌کنیم. مراحل زیر را حداکثر 2^n بار تکرار می‌کنیم:

  ۱. یک هم‌سایه‌ی تصادفی از زیرمجموعه‌ی فعلی پیدا کرده و نام آن را N می‌گذاریم.

  ۲. اگر هم‌سایه‌ی آن (N) باقی‌مانده‌ی کم‌تری داشت، مقدار آن را برابر با زیر مجموعه‌ی فعلی قرار می‌دهیم. به‌طور دقیق‌تر: $currentState = N$

  پی‌نوشت: زیرمجموعه $A \subseteq S$ هم‌سایه‌ی $B \subseteq S$ است، اگر یکی از شرایط زیر برقرار باشد:

  - بتوان یک یا دو عدد از A به B انتقال داد.
  - بتوان یک یا دو عدد از B به A انتقال داد.
  - بتوان یک عدد از A را با یک عدد از B جابه‌جا کرد.

  یک راه آسان برای تولید یک هم‌سایه‌ی تصادفی (B) از زیرمجموعه‌ی A به صورت زیر است.

  ۱. اعضای مجموعه S را به‌طور صعودی مرتب کنیم.

  ۲. در ابتدا B را یک کپی از A مقداردهی اولیه کنیم.

  ۳. دو شاخص تصادفی i و j را انتخاب می‌کنیم. به‌طوری که: $1 \leqslant i, j \leqslant n$

  ۴. اگر x_i در A بود، از B آن را حذف می‌کنیم. در غیر این‌صورت آن را به B اضافه می‌کنیم.

  ۵. اگر x_j در A بود، آن‌گاه با احتمال $\frac{1}{2}$ آن را از B حذف می‌کنیم. اگر x_j در A نبود، آن‌گاه با احتمال $\frac{1}{2}$ آن را به B اضافه می‌کنیم.

- ب) مدلم چنین حالتی را ندارد اما بلد نیستم اثبات کنم. :))))

سؤال ۵. Gradient Descent

- آ) نادرست. زیرا اگر ϵ عدد بزرگی باشد، ممکن است الگوریتم واگرا شود. همانند شکل مربوط به قسمت «ب» سوال ۶.
- ب) درست. به مقدار Learning Rate بستگی دارد. اگر بزرگ باشد، می‌تواند آن را از آن کمینه‌ی محلی خارج کند، اما چون مقدار گام‌ها در این قسمت بزرگ است، احتمال واگرا شدن بیش‌تر است. در صورتی که طول گام کوچک باشد، به دلیل کوچک شدن مقدار، احتمال هم‌گرا نشدن به x^* زیاد می‌شود.
- ج) درست. در کلاس اثبات شد.
- د) درست. چون الگوریتم هم‌گرا و تابع f محدب است، بنابراین به کمینه‌ی سراسری هم‌گرا می‌شود.

سؤال ۶. Gradient Descent

- آ) فرمول Gradient Descent به طور زیر است:

$$x^{(t+1)} = x^{(t)} - \alpha \nabla_x f$$

$$\nabla_x f = \frac{df}{dx} = \frac{a^T x}{dx} = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \frac{df}{dx_3} \\ \frac{df}{dx_4} \end{bmatrix} = \begin{bmatrix} 2(x_1 + 10x_2) + 3x_4^2 \\ 20(x_1 + 10x_2) + 15x_2^2 + 4(x_2 - 2x_3)^3 \\ -8(x_2 - 2x_3)^3 \\ 6x_1x_4 \end{bmatrix}, x^0 = (10, 5, 5, 4)$$

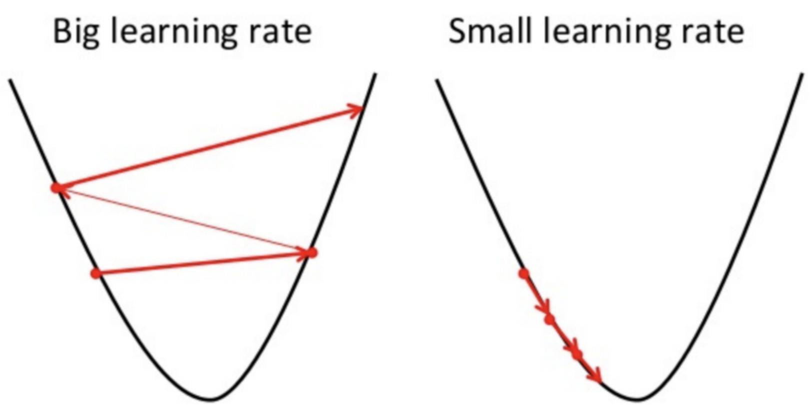
$$\rightarrow x^1 = \begin{bmatrix} 10 \\ 5 \\ 5 \\ 4 \end{bmatrix} - (0.1) \begin{bmatrix} 2(10 + 50) + 3(16) \\ 20(10 + 50) + 15(25) + 4(-125) \\ -8(-125) \\ 6(10)(4) \end{bmatrix} = \begin{bmatrix} -6.8 \\ -102.5 \\ -95 \\ -20 \end{bmatrix}$$

$$\rightarrow x^2 = \begin{bmatrix} -6.8 \\ -102.5 \\ -95 \\ -20 \end{bmatrix} - (0.1) \begin{bmatrix} 2(-6.8 - 1025) + 3(400) \\ 20(-6.8 - 1025) + 15(10506.25) + 4(-102.5 - 2(-95))^3 \\ -8(-102.5 - 2(-95))^3 \\ 6(-6.8)(-20) \end{bmatrix} = \begin{bmatrix} 79.56 \\ -281767.025 \\ 535842.5 \\ -101.6 \end{bmatrix}$$

- ب) هر چه مقدار learning rate کم‌تر باشد، برای پیدا کردن مقدار مینیمم قابل اعتمادتر است، اما از لحاظ بهینه‌سازی زمان بسیار زیادی طول می‌کشد تا به سمت مینیمم حرکت کند.

  اگر مقدار آن را بسیار بزرگ در نظر بگیریم، ممکن است جواب هم‌گرا نشود یا حتی واگرا شود.

  برای پیدا کردن مقدار مناسب برای learning rate پیشنهاد من این است که مقدار ثابت برای آن در نظر نگیریم. در ابتد که مقادیر با مقدار بهینه بسیار فاصله دارند، آن را بزرگ در نظر گرفته و رفته رفته آن را کم کنیم. به طور معمول مقادیر را در ابتدا ۰.۱ در نظر می‌گیرند و سپس به طور نمایی آن را کاهش می‌دهند (مثلا ۰.۰۱، ۰.۰۰۱ و...).



شکل ۱: تاثیر مقدار learning rate بر روی احتمال و سرعت رسیدن به جواب بهینه

- ج) در توابع غیرمحدب (non-convex) نمی‌توان مطمئن شد که به نقطه‌ی کمینه‌ی سراسری رسیده‌ایم یا خیر. زیرا علاوه‌بر عوامل دیگر بسیار بستگی به نقطه‌ی شروع دارد.

در توابع محدب (convex) با انتخاب مقدار مناسب (نه لزوما کوچک‌ترین مقدار) برای Learning Rate می‌توان مطمئن شد که الگوریتم نقطه‌ی کمینه‌ی سراسری را پیدا می‌کند. به طور دقیق‌تر، اگر اندازه‌ی گام‌ها بسیار بزرگ نباشد، تابع به یک مقدار با کمی خطا هم‌گرا می‌شود و چون تابع محدب است، کمینه‌ی سراسری خواهد بود. یعنی مقدار گرادیان در آن نقطه بسیار نزدیک به صفر خواهد بود.

سؤال ۷. Convex Functions

- آ) فرض کنید تابع f و تابع g محدب هستند. می‌خواهیم اثبات کنیم $h(x) = f(x) + g(x)$ نیز محدب است.
طبق تعریف داریم:

$$\forall x, y \in R, 0 \leqslant \alpha \leqslant 1:$$
$$f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha)f(y)$$
$$g(\alpha x + (1-\alpha)y) \leqslant \alpha g(x) + (1-\alpha)g(y)$$
$$h(x) = f(x) + g(x)$$
$$\to h = f(\alpha x + (1-\alpha)y) + g(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha)f(y) \ + \alpha g(x) + (1-\alpha)g(y)$$
$$\to h = (f+g)(\alpha x + (1-\alpha)y) \leqslant \alpha(f(x) + g(x)) + (1-\alpha)((f(x) + g(x))$$
$$h(\alpha x + (1-\alpha)y) \leqslant \alpha h(x) + (1-\alpha)h(y)$$

بنابراین ثابت می‌شود که جمع دو تابع محدب، محدب است. ■

- ب) فرض کنید $h(x) = Max\{f(x) + g(x)\}$ است.

$$h(\alpha x + (1-\alpha)y) = max(f(\alpha x + (1-\alpha)y),\ g(\alpha x + (1-\alpha)y))$$
$$h(\alpha x + (1-\alpha)y) \leqslant max(\alpha f(x) + (1-\alpha)f(y),\ \alpha g(x) + (1-\alpha)g(y))$$
$$h(\alpha x + (1-\alpha)y) \leqslant max(\alpha f(x),\ \alpha g(x)) + max((1-\alpha)f(y),\ (1-\alpha)g(y))$$
$$h(\alpha x + (1-\alpha)y) = \alpha h(x) + (1-\alpha)h(y)$$

پس ثابت شد که ماکسیمم دو تابع محدب نیز محدب است. ■

- ج)

(روش اول) ساده‌ترین راه برای اثبات این است که نشان دهیم حد زیر صعودی است. فرض شده است هر دو تابع f و g صعودی هستند.

$$(fg)'_+(x) = \lim_{h \to 0^+} \frac{(fg)(x+h) - (fg)(x)}{h}$$

پس داریم:

$$\begin{aligned}\frac{(fg)(x+h) - (fg)(x)}{h} &= \frac{(fg)(x+h) - f(x+h)g(x) + f(x+h)g(x) - (fg)(x)}{h}\\ &= f(x+h)\frac{g(x+h) - g(x)}{h} + g(x)\frac{f(x+h) - f(x)}{h}\\ &\to f(x)g'_+(x) + g(x)f'_+(x),\end{aligned}$$

با توجه به فرض بالا مبنی بر صعودی بودن هر تابع، پس تابع $(fg)'_+$ نیز صعودی است. بنابراین تابع fg محدب است. ■

(روش دوم) اگر تابع h را به صورت $h(x) = f(x)g(x)$ در نظر بگیریم، طبق تعریف تابع محدب داریم:

$$\begin{aligned}h(\alpha x + (1-\alpha)y) &= f(\alpha x + (1-\alpha)y)g(\alpha x + (1-\alpha)y)\\ &\leqslant [\alpha f(x) + (1-\alpha)f(y)][\alpha g(x) + (1-\alpha)g(y)]\\ &= \alpha^2 f(x)g(x) + \alpha(1-\alpha)[f(x)g(y) + f(y)g(x)] + (1-\alpha)^2 g(x)g(y)\\ &\leqslant \alpha^2 f(x)g(x) + (1-\alpha)^2 g(x)g(y) + \alpha(1-\alpha)[f(x)g(x) + f(y)g(y)]\\ &= \alpha f(x)g(x) + (1-\alpha)f(y)g(y) = \alpha h(x) + (1-\alpha)h(y)\end{aligned}$$

پس ثایت می‌شود که تابع h نیز محدب است. ■

- د) با توجه به روش اول حل قسمت «ج»، کافی است نشان دهیم تابع $\frac{f(x)}{g(x)}$ صعودی است. با توجه به قضایای مربوط به مشتق می‌دانیم که:

$$(\frac{f}{g})\prime(x) = \frac{f\prime(x)g(x) - f(x)g\prime(x)}{g(x)^2}$$

با توجه به این‌که مقدار مخرج مثبت، تابع f صعودی و تابع g نزولی هستند، نتیجه می‌گیریم که تابع $\frac{f(x)}{g(x)}$ نیز صعودی و محدب است. ■