تکلیف سری اول درس هوش مصنوعی

" هوالعلیم "

مریم سعیدپور ۹۶۲۹۳۷۴

سوال اول :

$X = abcdefgh$

$f(x) = (a+b)-(c+d)+(e+f)-(g+h)$

$x_1 = 65413532$, $x_2 = 87126601$, $x_3 = 23921285$, $x_4 = 41852094$

الف)

$f(x_1) = (6+5)-(4+1)+(3+5)-(3+2) = 9$

$f(x_2) = (8+7)-(1+2)+(6+6)-(0+1) = 23$

$f(x_3) = (2+3)-(9+2)+(1+2)-(8+5) = -16$

$f(x_4) = (4+1)-(8+5)+(2+0)-(9+4) = -19$

برحسب برازندگی

$\boxed{x_4 < x_3 < x_1 < x_2}$

$\Rightarrow$ برازندگی کل $= -3$

ب)

• single-point crossover : دو برازنده ترین ترکیب شوند .

871	26601	→	y_1 = 871	13532

x_2 = 871 | 26601

x_1 = 654 | 13532

y_1 = 871 | 13532

y_2 = 654 | 26601

• two-point crossover : دوم و سوم ترکیب شوند .

x_2 = 871 | 26 | 601

x_3 = 239 | 21 | 285

y_3 = 871 | 21 | 601

y_4 = 239 | 26 | 285

• Uniform crossover : اول و سوم ترکیب شوند
(برای هر بیت ، سکه انداختم)

x_1 = 6|5|4|1|3|5|3|2

x_3 = 2|3|9|2|1|2|8|5

y_5 = 6|5|4|1|3|2|8|5

y_6 = 2|3|9|2|1|5|3|2

پ)

$f(y_1) = (8+7)-(1+1)+(3+5)-(3+2) = 16$

$f(y_2) = (6+5)-(4+2)+(6+6)-(0+1) = 16$

$f(y_3) = (8+7)-(1+2)+(1+6)-(0+1) = 18$

$f(y_4) = (2+3)-(9+2)+(6+2)-(8+5) = -11$ ×

$f(y_5) = (6+5)-(4+1)+(3+2)-(8+5) = -2$

$f(y_6) = (2+3)-(9+2)+(1+5)-(3+2) = -5$ ×

$\Rightarrow$ برازندگی کل $= 48$

بله برازندگی کل جمعیت جدید (48) نسبت به جمعیت اولیه (3-) افزایش چشمگیری داشته است.

ت)

$\boxed{\max(f(x)) = (9+9)-(0+0)+(9+9)-(0+0) = 36}$

$\boxed{X_{\text{بهینه}} = 99009900}$

ث) از 2-swap میتوان استفاده کرد. همچنین میتوان Uniform-mutation (که برای حالت real-value بود) را به گونه ای تعریف کرد که $\{1,2,3,\dots,9,0\} = A$ و حالا یک موضع به تصادف انتخاب شده و یک مقدار از دامنه ی A انتخاب شده و در موضع منتخب جایگزین رقم فعلی میشود .

ج) خیر . در single-point crossover قرار است یک برش داشته و چهار قطعه ایجاد شده را طبق تعریف، به نحوی جایگزین یکدیگر کنیم . حالا یک نگاه به ژن اول (a) جمعیت اولیه کافیست تا بفهمیم اصلا در این ژن ، رقم 9 نداشتیم و حالا هر چقدر هم این single-point Crossover را اجرا کنیم ، محال است در ژن a ، 9 ظاهر شود پس محال است به جواب بهینه یعنی 99009900 برسیم و نقطه mutation لازم است!

سوال دوم:

الف) الگوریتم شبیه‌سازی ذوب فلزات با $T=0$: $e^{\frac{\Delta E}{T}} \to 0$ در نتیجه کاملاً حریصانه داریم رفتار می‌کنیم. مثل الگوریتم HC ورژن steepest descent/ascent.

ب) الگوریتم شبیه‌سازی ذوب فلزات با $T=\infty$: $e^{\frac{\Delta E}{T}} \to 1$ در نتیجه کاملاً تصادفی داریم رفتار می‌کنیم. مثل الگوریتم random walk.

پ) الگوریتم جستجوی پرتوی محلی با $K=1$: همان HC است ورژن steepest descent/ascent

ت) الگوریتم جستجوی پرتوی محلی با $K=\infty$: این دیگه عملاً داره کل فضای حالت را جستجو می‌کند :: مثل BFS (سطح به سطح جلو، همه همسایه‌ها را می‌سازه و بعد دوباره برای تمام نودهای فعلی، تمام همسایه‌ها ... الی آخر.)

ث) الگوریتم ژنتیک وقتی که هر نسل فقط شامل یک نفر باشد: چون فقط یک نفر در هر نسل داریم اصلاً crossover معنا ندارد و الگوریتم با جهش ادامه پیدا می‌کند و چون جهش یک حرکت تصادفی است و عملاً انتخابی برای نسل بهتر یا بدتر نداریم، پس عملاً نشستیم به امید جهش (یک حرکت تصادفی) ← میشه گفت شبیه الگوریتم random walk میشه.

سوال سوم:

الف) با الگوریتم HC

← هدف: یافتن مسیر با کمترین هزینه.

$A6 \to B3 \to G2 \to G1 \to F0$

در این مسیر ارائه شده، هزینه برابر با 12 که یک Global Optima هست و بهینه‌ترین جواب ممکن.

← هدف: یافتن مسیر با بیشترین پاداش:

$A_6 \xrightarrow{2} B_3 \xrightarrow{2} C_3 \xrightarrow{2} D_3 \xrightarrow{3} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

$D_3 \xrightarrow{3} G_2 \xrightarrow{1} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

از D_3 به طور تصادفی یک مسیر را پیش می‌گیرد. اگر فرض کنیم شاخه پایین را انتخاب کند پس:

$A_6 \xrightarrow{2} B_3 \xrightarrow{2} C_3 \xrightarrow{2} D_3 \xrightarrow{3} G_2 \xrightarrow{1} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

که در این مسیر پاداش برابر با 21 است و یک local Optima است. بهترین پاداش $+\infty$ است. (اگر از گره E_1 با یال صفر به I_4 می‌رفت و در حلقه‌ای شامل نودها $(E_1, I_4, H_5, G_2, G_1)$ گیر می‌افتاد!)

ب) با الگوریتم TS

← هدف: یافتن مسیر با کمترین هزینه. دقیقاً مشابه مثال قبل است

$A6 \to B3 \to G2 \to G1 \to F0$ در Global Optima متوقف می‌شود.

← هدف: یافتن مسیر با بیشترین پاداش. این حالت هم مشابه قبل است. یکی از دو مسیر فوق به تصادف انتخاب می‌شود. فقط احتمال اتفاق اولویتی که گفتیم از بین میره کاملاً. در local Optima متوقف نمی‌شود.

پ) مکانیزم محلی یا سراسری؟ در مورد مشخص شد.

ت) الگوریتم برای ارضای هر دو شرط به صورت همزمان: باید دقیقاً تابع هدف مشخص بشود. بین این دو trade off وجود دارد و شاید مجبور باشیم اولویت‌دهی کنیم. به هر حال باید تابع هدف مشخص باشد؛ ولی اگر منظور این است که مسیری همزمان بهترین باشد از لحاظ هزینه‌ی کم و پاداش زیاد، وجود ندارد. بدیهی است که کم‌هزینه‌ترین مسیر (هزینه 12) یک Optima سراسری بود

← ادامه قسمت (ت) سوال سوم: در حالی که بهترین مسیر از نظر پاداش، حاوی یک لوپ است و پاداش در این مسیر بهینه برابر است با $2+2+2+3+5+(0+2+4+1+5)n+6 = +\infty$.

اما اگر مثلاً تابع هدف را به صورت: $\frac{\text{پاداش}}{\text{هزینه}} = f(x)$ معرفی کنم که در این صورت مسئله maximization است.

و کافیست با الگوریتم های local search و تابع هدف $f(x)$ که قرار است max شود، مسئله را حل کنیم.

یا مثلاً تابع هدف را به صورت: $f(x) = \text{پاداش} - \text{هزینه}$ معرفی کنیم و هدف max کردن $f(x)$ باشد.

در هر حال برای تابع هدف $\frac{\text{پاداش}}{\text{هزینه}}$ ، HC و TS هر دو یک جواب می دهند:

$A_6 \to B_3 \to C_3 \to D_3 \to G_1 \to F_0$

برای تابع هدف دوم (هزینه - پاداش)، باز هم HC و TS یک جواب می دهند:

(حالتهایی که مسیر دو شاخه می شود، هر دو تقسیم می کنیم و می رویم.)

$A_6 \to B_3 \to C_3 \to D_3 \to G_1 \to F_0$

$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \hookrightarrow E_1 \to F_0$

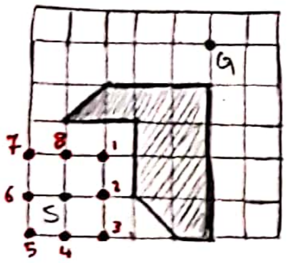
$\quad\quad \hookrightarrow G_2 \to G_1 \to F_0$

$\quad\quad\quad\quad\quad\quad\quad \hookrightarrow E_1 \to F_0$

سوال چهارم:

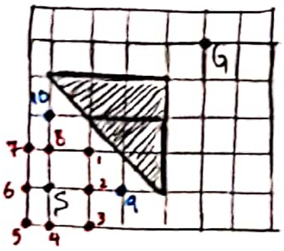
← الف) چگونگی کارکرد الگوریتم HC: همانطور که با تعریفی که از همسایگی برای مسئله در نظر می گیریم، هر بار می آید همسایه ها را پیدا کرده و آن که به مقصد (G) نزدیک تر است را انتخاب می کند. مثلاً اگر منظور از همسایگی، آن هایی باشد که با فاصله ی یک از نقطه فعلی هستند، 8 همسایه برای S پیدا می کنیم که از این بین آن که به G نزدیک تر است را انتخاب می کنیم و به آن نقطه می رویم.

← ب) مثال برای گیر کردن در بهینه محلی با موانع غیر محدب:



در مثال روبرو یک مانع غیر محدب داریم. همسایگی را هم نقاط با فاصله ی یک از نقطه فعلی تعریف کردم. می بینیم که 8 همسایه داریم که از بین این هشت تا، همسایه شماره 1 به G نزدیک تر است و اگر به آن برویم بعد 3 همسایه داریم که همین شماره های 8 و 2 و S خواهند بود و هیچ کدام ارتقادهنده نیستند و در نتیجه الگوریتم در این local optima گیر می افتد.

← پ) گیر افتادن با موانع محدب؟

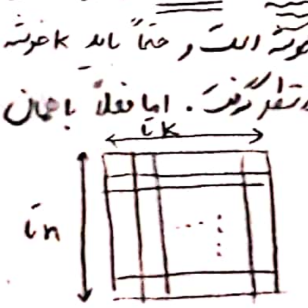


در مثال روبرو یک مانع محدب داریم. همسایگی را هم نقاط با فاصله ی یک از نقطه ی فعلی تعریف کرده ام. می بینیم که 8 همسایه داریم که از این بین، همسایه شماره 1 به G نزدیک تر است و اگر به آن برویم بعد 5 همسایه داریم (9، 2، S، 8، 10) که ارتقادهنده نیستند و الگوریتم همین جا متوقف می شود.

← ت) آیا SA کمک کننده است برای فرار از ها؟ از آنجایی که این الگوریتم در مواضعی که همسایه ارتقادهنده نداریم، به صورت تصادفی عمل می کند، به ما امکان خارج شدن از local optima را می دهد تا به این ترتیب باز هم شانس رسیدن به بهینه سراسری را داشته باشیم.

سوال پنجم:

← الف) Representation: یک ماتریس باینری $n \times k$ (n راس ها و k خوشه ها) به شرطی که حداقل یک "یک" در هر ستون و دقیقا یک "یک" در هر سطر باشد. (هر راس فقط متعلق به یک خوشه است و حتماً باید k خوشه داشته باشیم) می توان شرط ها را برداشت و جریمه برای نقض آن در نظر گرفت. اما فعلاً با همان شرط ها ادامه می دهیم.



ادامه‌ی سوال پنجم:

ب) تعداد کل اعضای فضای جستجو: اول با K راس، در هر خوشه یک راس می‌گذاریم تا مطمئن شویم دقیقاً K خوشه داریم. حالا از $(n-K)$ راس باقی‌مانده، هر راس می‌تواند در یکی از K خوشه دلخواه باشد. لذا برای این راس‌های باقیمانده K^{n-K} حالت داریم. (تا اینجا $\binom{n}{K}K^{n-K}$). حالا از بین این حالات باید مواردی که دو سریال در خوشه‌های متفاوت باشند را کم کنیم (فرض کنیم ۱۸۷ یک سریال باشد، یکبار ۱۸ در خوشه یک و ۷ در خوشه دو است در حالت دیگر ۷ در خوشه دو و ۱۸ در خوشه یک است ولی در دو حالت از آنجایی که دسته‌ها حائز اهمیت نیستند یکسان محسوب می‌شوند پس باید این موارد از حالات کلی کم شود) خلاصه تعداد کل فضای جستجو: $\frac{\binom{n}{K}K^{n-K}}{2}$

پ) تعریف همسایگی: یه نوعی همسایگی جدیدی تعریف می‌کنیم به این شکل که:

خوشه‌ی راس را عوض کنیم به این شرط که تعداد خوشه‌ها از K کمتر نشود (اگر خوشه‌ای تنها یک راس داشت، این راس بلاک می‌شود و حق تغییر خوشه ندارد.)

در مورد representation گفته شده، یک سطر را (یعنی یک راس را) انتخاب کرده و اول چک می‌کنیم تک راس در خوشه نباشد (در ستون مربوط اش اگر تنها یک مورد یک نباید به آن راس دست بزنیم.) و سپس خوشه مربوط آن را عوض می‌کنیم اگر راس بلاک شده نبود.

ت) ماکزیمم و مینیمم تعداد همسایه‌ها:

* کمینه تعداد همسایگی: شرایطی را فرض می‌کنیم که $(K-1)$ خوشه هر کدام با یک راس داریم و یک خوشه با $(n-K+1)$ راس. در این شرایط فقط از بین $(n-K+1)$ راس می‌توان یک راس را برای تغییر خوشه انتخاب کنیم پس کلاً: $(n-K+1)\times(K-1)$ همسایگی داریم.
* بیشینه تعداد همسایگی: شرایطی که می‌توانیم هر راس دلخواهی را تغییر خوشه دهیم، مطلوب را می‌دهد. پس در کل تعداد بیشینه همسایگی برابر است با: $n\times(K-1)$.

ث) در این خصوص (البته به راهنمایی TA) در مورد الگوریتم‌های Graph clustering تحقیق کردم و متوجه شدم این مسئله، یک مسئله‌ی NP-Complete هست و الگوریتم‌های Exact ندارند. همچنین مسئله‌ی این سوال هم بسیار مشابه Graph clustering است و از آن هم الگوریتم Exact ندارد.

سوال ششم

کد و گزارش جزئی از روند تهیه که در فایل آپلود شده در سامانه موجود است.