به نام خدا

تمرین اول پیاده سازی

درس هوش مصنوعی

احمد زعفرانی

97105985

بهار 98-99

راهنمای اجرای کد:

- ابتدای بازه ی تغییرات مجهول ها، انتهای این بازه و طول گام جابجایی مجهولات را به ترتیب و در یک خط به برنامه بدهید (هر کدام بصورت عدد صحیح یا اعشاری).
- فایل ورودی برنامه باید با نام input.txt در همان پوشه ای که کد در آن قرار دارد ذخیره شود.
- در صورتی که تعداد ضرایب یک معادله (تعداد اعدادی که در یکی از خطوط فایل ورودی وجود دارند و با "،" از هم جدا شده اند) از N+1 کمتر باشند، برنامه با یک پیغام خطا به پایان می رسد.

گزارش کار:

1. تابع ارزیاب را اینگونه تعریف می کنیم:

اگر a_1 تا a_n مجهولات ما باشند، مقدار پیشنهادی الگوریتم در هر مرحله، یک بردار با n مولفه است. حال پاسخ پیشنهادی الگوریتم را در هر مرحله در معادله i ام جایگذاری می کنیم و عدد ثابت سمت راست معادله را از آن کم می کنیم و حاصل را x_i می نامیم. بنابراین اگر تعداد معادله های ورودی m معادله باشد، $f(x)$ را اینگونه تعریف می کنیم:

$$f(x) = -\sum_{i=0}^{m} |x_i|$$

از این تابع هم به عنوان "value" یک حالت در جست و جوی تپه نوردی، و هم به عنوان "E (energy)" یک حالت در الگوریتم شبیه سازی ذوب فلزات بهره می جوییم (همانطور که مستحضرید، هر چقدر قدر مطلق $f(x)$ به صفر نزدیکتر باشد، در وضعیت مطلوبتری هستیم؛ بنابراین یک منفی در این تابع ضرب می کنیم تا مطابق تعریفمان از توابع ارزیاب شود).

2. اولین چالش در پیاده سازی، کار کردن با اعداد اعشاری بود. به این منظور از کلاس decimal که بصورت پیش فرض در پکیج های زبان پایتون تعریف شده است. با کمک این کلاس اعداد ممیز شناور را به اعداد ممیز ثابت تبدیل کردم و دقت محاسبات را نیز تا 6 رقم اعشار کاهش دادم.

3. برای تولید نقطه ی شروع الگوریتم، مختصات را بصورت اعداد تصادفی در بازه ای که برنامه در ابتدا دریافت می کند تولید می کنم. همچنین طول گام، از حاصل ضرب طول گام ورودی در یک عدد تصادفی بین 0 تا 1 بدست می آید و تا 1 رقم اعشار گرد می شود. برای هر نقطه نیز 2N همسایه تولید می کنم.

4. زمان اجرای کد که در خروجی اعلام می شود، به محض دریافت ورودی ها از کنسول شروع به محاسبه می شود؛ این زمان شامل زمان خواندن ورودی ها از فایل، مجموع زمان اجرای الگوریتم ها روی مسئله به تعداد معین و مدت زمانی است که طول می کشد تا همسایه های هر نقطه تولید شوند و برازش آنها محاسبه شود. همچنین معیاری برای خطای تابع ارزیاب تعریف می کنیم:

X = max{ |interval_start| , |interval_end| }

مجموع قدر مطلق تمام اعداد درون فایل ورودی، ضرب در X . درصد خطای این مقدار را 100، و درصد خطای جواب معادله (نقطه ای که تابع ارزیاب را صفر می کند) را صفر تعریف می کنیم.

بنابراین درصد خطای هر نقطه ای مانند A از این رابطه بدست می آید:

$$\frac{|f(A)|}{x} * 100$$

5. برای الگوریتم شبیه سازی ذوب فلزات، چالش اول تعیین تابع دما است. برای این کار ابتدا یک حد بالا برای بیشینه ی گام هایی که ممکن است در الگوریتم شبیه سازی ذوب فلزات برداشته شوند را اینگونه تعریف کردم:

Max_steps_sas = (interval_end – interval_start) / step_length

که step_length همان عددی است که در ابتدای برنامه به عنوان طول گام از کاربر دریافت می شود.

بنابراین تابع دما اینگونه تعریف می شود:

Temperature = (Max_steps_sas – تعداد گام هایی که تاکنون برداشته ایم) * N

همچنین برای پیاده سازی این بخش از الگوریتم که عنوان می دارد:"اگر همسایه ای که بصورت تصادفی انتخاب شده است انرژی کمتری نسبت به حالت فعلی داشت، آن را با احتمال p انتخاب کن" ، عددی تصادفی بین صفر و یک تولید می کنم؛ اگر این عدد تصادفی از p کوچکتر بود، این همسایه انتخاب می شود، در غیر اینصورت همسایه دیگری برای حالت فعلی تولید می کنم.

6. پس از تکمیل نسخه ی اولیه ی کد، مشاهدات اولیه حاکی از پیروزی قاطع جست و جوی تپه توردی بر الگوریتم شبیه سازی ذوب فلزات بودند. نتیجه ی زیر، حاصل اولین اجرای ورژن اول کد روی فایلی است که به عنوان مثال همراه pdf سوالات ارسال شده بود:

please enter start of interval, end of interval and step length
-100 100 1
Hill Climbing Search ran 10 times. best answer started from a point with coordinates:
[-88.134, -94.009, 29.24, -20.5, 7.927, -33.02, 66.848, 35.712, -71.743', -90.829]
and fitness : -1595.22

with step length : 0.1

and now a local maximum found with coordinates :

[30.366, 104.591, -31.96, -97.8, -15.673, -33.02, 116.048, 28.612, -71.743, 87.771]
and fitness : -655.765
in 3.945474 seconds and the algorithm found the answer in 7157 steps.
****
Simulated Annealing Search ran 10 times. best answer started from a point with coordinates :
[-85.927, 56.705, 84.532, -60.09, -19.919, 8.887, 47.713, 13.133, -53.269, 32.321]
and fitness : -1031.85
with step length : 0.5 and now the temperature reach's zero, after : 2000 steps.
the answer found in 0.066177 seconds a point with coordinates :
[-78.927, 50.205, 92.532, -61.59, -17.919, 10.387, 60.213, 11.133, -50.769, 47.821]
and fitness : -952.506

اجرای کد 7.923069 ثانیه طول کشیده بود و درصد خطای بهترین جواب (تپه نوردی) 0.67 درصد محاسبه شده بود. اجراهای بعدی کد با همین ورودی ها مشمئز کننده نیز بودند! چراکه الگوریتم شبیه سازی ذوب فلزات، برای چند بار نقاطی را به عنوان جواب اعلام کرد که برازندگی شان حتی از برازندگی نقطه ی اولیه ای که این الگوریتم کارش را با آن آغاز می کرد، کمتر بود!

لکن نمی توان از این نکته هم غافل شد که زمان اجرای این الگوریتم به طور متوسط بین $\frac{1}{30}$ ام تا $\frac{1}{50}$ ام زمان اجرای الگوریتم جست و جوی تپه نوردی است، هر چند تعداد گام های این الگوریتم بطور متوسط 3 برابر گام های الگوریتم دیگر است.

7. در آخرین گام، کمی به کارهای آماری روی آوردم! برنامه ای با نام play.py ایجاد کردم که تقریبا همان کار کد اصلی را می کرد؛ با این تفاوت که پس از ایجاد یک نقطه ی شروع تصادفی و یک طول گام تصادفی، هر دو الگوریتم را روی این ورودی ها 100 بار اجرا می کرد و خروجی را در دو فایل مجزا می نوشت. سپس به کمک نرم افزار اکسل، جدول زیر از اجرای چندین باره ی این برنامه استخراج شد.

هر 3 پارامتر جدول، میانگین نتیجه ی 100 بار اجرا شدن الگوریتم ها هستند. همچنین برای تولید معادله از برنامه ای دیگر که به تعداد تصادفی، ضرایبی تصادفی تولید می کرد استفاده کردم.

بازه تغییرات مجهولات	تعداد مجهولات	جست و جوی تپه نوردی			شبیه سازی ذوب فلزات		
		تعداد گام	زمان	بهبود برازش	تعداد گام	زمان	بهبود برازش
-100 تا 100	10	1100	0.599167	407.4035	2000	0.067131	1.57581
-1000 تا 1000	10	6939	4.003837	4128.26	20000	6.99216	20.6751
-400 تا 200	5	240	0.020223	195093.6	300	0.005235	146035.7
-4000 تا 2000	5	3424	0.296358	2040000	3000	0.049451	1340000
-10تا 10	40	323	11.3926	259886.5	800	0.358343	186397.5

8. نتایجی جالبی از جدول بالا می توان بدست آورد:

- اگر کمیت برای ما مهم است، بدون شک جست و جوی تپه توردی عملکرد بهتری خواهد داشت. (تا کنون به مثالی بر نخورده ام که الگوریتم شبیه سازی ذوب فلزات تابع ارزیاب را بیشتر از جست و جوی تپه نوردی بهبود ببخشد)
- اما سرعت بالا مشخصه و برتری اصلی الگوریتم شبیه سازی ذوب فلزات است.
- تاثیر افزایش تعداد مجهولات و حتی افزایش تعداد معادلات، اثر بیشتری بر افزایش زمان اجرای هر دو الگوریتم، نسبت به بزرگ کردن بازه ی تغییر محهولات دارد.
- عملکرد تصادفی الگوریتم شبیه سازی ذوب فلزات در جدول بالا کاملا مشهود است؛ اما اگر مشکل هزینه ی محاسبه، زمان یا حافظه داریم، معقول است امیدوار باشیم با چند بار اجرای شبیه سازی ذوب فلزات، می توان پاسخی یافت که "تقریبا" عملکردی به خوبی جست و جوی تپه نوردی داشته باشد.

پایان