

3/25/2019

گزارش تمرین عملی دوم

هوش مصنوعی

Mohammad Reza Ghamkhar

95106494

الگوریتم ژنتیک:

Fitness function:

دو تا تابع مختلف برای این هدف نوشتیم:

تابع اول یا fitness_function:

این تابع با استفاده از کتاب قانونی که میگیره بخشی از جامعه را تا عمقی به اندازه depth شبیه سازی میکنه و در نهایت تفاوت تعداد افراد خواب با بیدار را محاسبه میکند.

سپس یک مرحله دیگه پیشرفت جامعه را شبیه سازی میکند و این بار تفاوت تعداد افراد بیدار با خواب را اندازه گیری میکند و این دو عدد را باهم جمع میزند و به عنوان خروجی باز میگرداند.

نکات حائز اهمیت:

۱: برای شبیه سازی پیشرفت جامعه فقط "بخشی" از آن برای این کار انتخاب میشود تا از سرعت برنامه کاسته نشود.

۲: برای تعداد مراحل شبیه سازی از متغیر depth استفاده میکنیم که در ابتدا ۱۰ است و به مرور زمان زیادتر میشود تا بهتر بتواند کتاب های مناسب و مناسب تر را از هم تشخیص دهد.

۳: بعد از شبیه سازی اول باز هم یک مرحله شبیه سازی میکنیم تا علاوه بر اهمیت یکسان شدن رنگ ها اهمیت نوسان کردن هم به برنامه بفهمانیم.

تابع دوم یا fitness_function2:

بعد از بررسی ناکارآمدی تابع اول حدس زدم که با صرف استفاده از شمردن نمی توان مقدار خوبی برای fitness بدست آورد برای همین سعی کردم ناحیه محور به سطح سیاره یا ماتریس نگاه کنم.

در این تابع برای هر نقطه تعداد خانه های همسایه بیدار و خواب (به جز خانه پایین که تاثیری ندارد) را تحت عنوان situation matrix ذخیر میکنم.

حال شباهت خانه های کنار هم را با تفریق تعداد همسایه های بیدار با خواب ذخیر میکنم.

سپس پیشرفت جامعه را چند مرحله شبیه سازی میکنم و دوباره با محاسبه situation matrix شباهت خانه های نزدیک به هم را بررسی میکنم.

و براساس بهتر شدن شباهت `sync_value` را مقدار دهی میکنم.

اما `Sync_value` به برنامه اهمیت یکرنگ شدن را میگوید که این باعث خوابیدن یا بیدار شدن همه جامعه میشود و برای ما بیدار شدن سپس خواب یا به عبارتی چشمک زدن هم مهم است برای این کار `situation matrix` را در نظرم میگیریم و جامعه را یک مرحله شبیه سازی میکنیم حال می گوییم هرخانه باید رنگش مخالف رنگ اکثریت همسایه هایش در مرحله قبل باشد و مقدار آن را در `toggle_value` ذخیره میکنیم که بیانگر اهمیت چشمک زدن یا خواب و بیدار شدن هر نفر است.

جمع وزن دار این دو عدد را به عنوان خروجی `fitness function` بر میگردانیم.

Cross_over:

در این تابع دو کتاب قانون را می گیریم و از یک قسمت رندوم آن ها را تکه کرده و با هم ترکیب میکنیم همانند مسئله `n` وزیر.

Mutate:

برای این کار هم دو تابع در نظر گرفتیم که در تابع اول هر کتاب مقداری احتمال داشت که دچار جهش شود و اگر میشد به بند شانس آن کتاب قانون عوض میشد.

در تابع دوم هر بند کتاب مستقلاً شانس جهش پیدا کردن دارد و شانس آن برابر مقدار متغیر `mutation_rate` است.

در این برنامه از تابع دوم برای جهش ژنتیکی استفاده شده است.

نکات تکمیلی:

۱: فضای نمونه را ۱۴ در نظر گرفتیم یعنی در هر مرحله ۱۴ کتاب قانون داریم و روی آن ها تابع های مختلف را صدا میزنیم.

۲: شانس جهش ۰,۵ درصد در نظر گرفته شده.

۳: عمق بررسی برای تابع `fitness` در حالت اولیه ۱۰ هست و هر ۲۰ مرحله در ۱,۱ ضرب میشود.

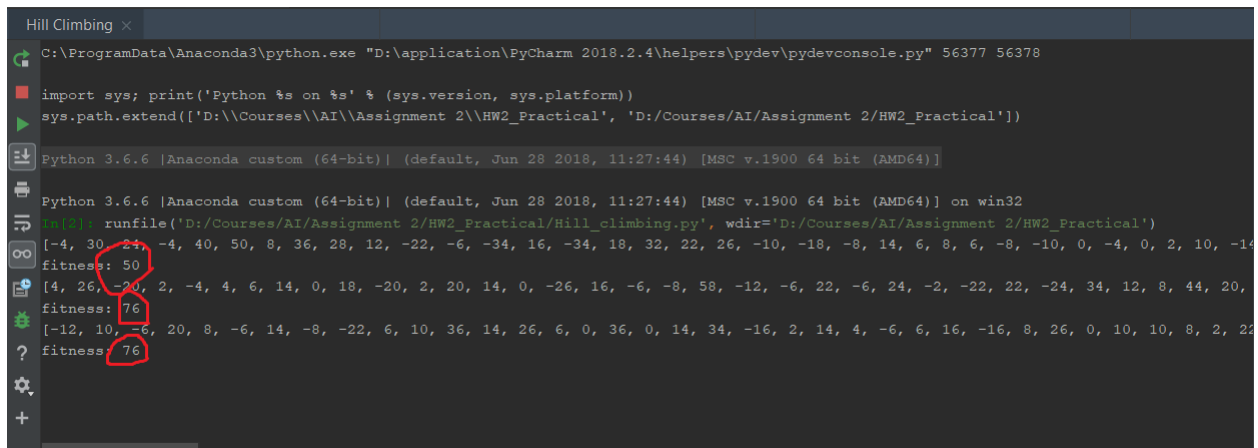
۴: همه این مقادیر تجربی بدست آمدند.

Hill Climbing:

صورت سوال خواسته که از hill climbing ساده استفاده کنیم، پس در هر مرحله همه همسایه ها را میسازیم (کتاب هایی که فقط در یک بند با کتاب ما تفاوت دارند) و fitness شان را که مشابه تابع اول الگوریتم ژنتیک قسمت قبل بدست می آید را محاسبه میکنیم.

سپس بهترین همسایه را، اگر از کتاب ما نیز بهتر باشد، انتخاب میکنیم و به آن میرویم.

ولی مشاهده میشود که بعد از حدود ۲۳ مرحله دچار local maxima می شویم و همسایه بهتری برای رفتن وجود ندارد و الگوریتم ما ازونجایی که طبق صورت سوال hill climbing ساده است گیر میکند و پیشرفتی نمی کند.



```
Hill Climbing x
C:\ProgramData\Anaconda3\python.exe "D:\application\PyCharm 2018.2.4\helpers\pydev\pydevconsole.py" 56377 56378

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\Courses\AI\Assignment 2\HW2_Practical', 'D:/Courses/AI/Assignment 2/HW2_Practical'])

Python 3.6.6 [Anaconda custom (64-bit)] (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)]
Python 3.6.6 [Anaconda custom (64-bit)] (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32
In [5]: runfile('D:/Courses/AI/Assignment 2/HW2_Practical/Hill_climbing.py', wdir='D:/Courses/AI/Assignment 2/HW2_Practical')
[-4, 30, 24, -4, 40, 50, 8, 36, 28, 12, -22, -6, -34, 16, -34, 18, 32, 22, 26, -10, -18, -8, 14, 6, 8, 6, -8, -10, 0, -4, 0, 2, 10, -14]
fitness: 50
[4, 26, -20, 2, -4, 4, 6, 14, 0, 18, -20, 2, 20, 14, 0, -26, 16, -6, -8, 58, -12, -6, 22, -6, 24, -2, -22, 22, -24, 34, 12, 8, 44, 20,
fitness: 76
[-12, 10, -6, 20, 8, -6, 14, -8, -22, 6, 10, 36, 14, 26, 6, 0, 36, 0, 14, 34, -16, 2, 14, 4, -6, 6, 16, -16, 8, 26, 0, 10, 10, 8, 2, 22]
fitness: 76
```