به نام خدا

معین شیردل

۸۱۰۱۹۷۵۳۵

گزارش پروژه دوم هوش مصنوعی

الگوریتم های ژنتیک

فاز اول:

ساختار کلی برنامه به این صورت است که هر راه حل که به عنوان یک کروموزوم شناخته می‌شود، دارای تعدادی ژن است که در این الگوریتم، تعداد ژن ها همان تعداد گیت هایی است که باید در انتها خروجی بدهیم. در ابتدای برنامه به کمک کتاب‌خانه csv فایل دارای جدول درستی گیت های مورد نظر سوال ورودی گرفته می شود و تعدادی پارامتر نظیر تعداد گیت های راه حل و ماکسیمم مقدار fitness با توجه به تعداد ستون ها و سطر های آن برداشت می‌شود. این فایل csv به یک لیست تبدیل می‌شود (چون سرعت کار با لیست به مراتب بالاتر از Data Frame های کتاب‌خانه pandas بود) و برای به دست آوردن مقدار fitness (سازگاری) که در ادامه توضیح داده می‌شود، به کار می‌آید. آرایه total_population کروموزوم هایی که جمعیت فعلی ما هستند را در خود نگه می دارد و این جمعیت، هر کدام از نوع یک object به نام Chromosome هستند که در خود مقدار fitness آن کروموزوم و لیست ژن های آن را در خود نگه می‌دارد. ژن ها در این الگوریتم، تعدادی رشته هستند که نشانگر نام گیت هستند. پس دامنه ی متغیر هایی که می توان به هر ژن نسبت داد دارای ۶ عضو است که همان ۶ گیت موجود در صورت پروژه هستند.

فاز دوم:

در این قسمت در همان ابتدای برنامه پس از load شدن جدول درستی درون لیست، تعدادی جمعیت اولیه به صورت کاملا رندوم به وجود می آوریم. تعداد کروموزوم های این جمعیت، به عنوان یک hyper parameter به نام POPULATION در ابتدای برنامه تعریف شده است که مقدار ۵۰ به آن داده شده است. یعنی در شروع کار، ۵۰ کروموزوم که هر کدام دارای تعدادی ژن (که به طور تصادفی و بدون جهت‌گیری تعیین می شوند) هستند، درون یک لیست از کروموزوم ها ریخته می شوند. این جمعیت اولیه ماست. برای هر کروموزوم، در همان لحظه ای که object مربوط به آن تشکیل می شود، مقدار fitness یا معیار سازگاری آن نیز محاسبه می شود.

فاز سوم:

معیار سازگاری در این مسئله، تعداد پاسخ های درست و منطبق بر ستون خروجی فایل csv داده شده به ازای هر ردیف از ورودی ها تعیین شده است. در همان هنگامی که ژن های کروموزوم های ما، یعنی گیت های راه حل ایجاد شده ی ما تعیین می‌شوند، معیار سازگاری آن نیز محاسبه می شود به این شکل که تمام ردیف های ورودی ممکن به این سری از گیت ها داده می‌شود و پاسخ خروجی آن ها با خروجی نوشته شده در همان ردیف از فایل csv مقایسه می شود. تعداد خروجی های صحیح، میزان خوب بودن آن راه حل یا سازگاری آن می باشد.

فاز چهارم:

الگوریتم به این شکل پیاده سازی شده است که فراوانی جمعیت در هر نسل تغییر نمی‌کند. در ابتدا تمام اعضای جمعیتمان را بر اساس fitness آن ها مرتب میکنیم. کروموزوم های ابتدای لیست، سازگاری بیشتری خواهند داشت و راه حل های بهتری دارند و به جواب نهایی نزدیک تر هستند. ۵۰ عضو نسل بعدی را به این شکل می‌سازیم:

- ۱۰ عضو ابتدایی نسل قبلی که بیشترین سازگاری را دارند وارد نسل بعدی می شوند.
- ۲۰ عضو ابتدایی نسل قبل را یک بار crossover و حاصل آن را mutate (با احتمال پایین) می کنیم که این فرایند ۲۰ عضو دیگر به ما می دهد.
- ۱۰ عضو سوم (۲۱ تا ۳۰ ام به ترتیب fitness) را به طور جدا mutate (با احتمال بالا) می کنیم و ۱۰ عضو جدید را وارد نسل بعدی می کنیم.
- همان ۱۰ عضو سوم را، این بار crossover می کنیم و ۱۰ عضو جدید را وارد نسل بعدی می کنیم.

به این شکل ۵۰ عضو جدید نسل بعدی به وجود می آیند و از آنجایی که از ۲۰ عضو آخر نسل قبل استفاده ای نکردیم، آن ها دور ریخته می‌شوند.

عملیات mutation: این عملیات به این صورت انجام می شود که هر ژن از یک کروموزوم در حال جهش را به یک احتمال به خصوص تغییر می دهیم. طی این عملیات ممکن است کروموزوم تمام ژن هایش تغییر کند یا اصلا هیچ تغییری نکند. احتمال جهش در دو حالتی که جهش انجام می دهیم متفاوت است که مقدار دقیق آن ها و علت تفاوت در آنها در ادامه توضیح شرح داده شده است. از هر ژن بعد جهش، یک ژن حاصل می شود.

عملیات crossover: به تابع crossover در برنامه، لیستی ازکروموزوم به عنوان کروموزوم های والد داده می‌شود. به طور تصادفی دو تا از والد ها را در هر مرحله (از بین والد های باقی مانده) انتخاب می‌کنیم. این دو والد قرار است که با هم ترکیب شوند و به جای آنها دو کروموزوم جدید ایجاد شوند. یک نقطه به طور تصادفی در این کروموزوم ها تعیین می شود که از آن نقطه این عملیات صورت بگیرد. برای اینکه این عملیات کروموزوم را بیشتر متحول کند، به شکلی طراحی شده است که از نقاط گوشه ای آن crossover صورت نگیرد چون در این حالت تعداد کمی از ژن ها در والد ها تغییر می کنند. پس از مشخص شدن این نقطه one-point crossover انجام می شود و دو فرزند از هر دو والد ایجاد می شوند.

فاز پنجم:

۱) در نهایت جواب مطلوب باید به ازای تمامی ورودی ها، جواب منطبق بر آنچه ردر جدول آورده شده است به ما بدهد. پس تعداد جواب های درست و منطبق بر جدول معیار خوبی برای مطلوب بودن کروموزوم و نزدیک بودن آن به راه حل درست، می‌باشد.

۲) روش انتخاب افراد منتخب برای نسل بعدی در بالا توضیح داده شده است. ما فقط از ۳۰ کروموزوم با سازگاری کمتر استفاده می‌کنیم و ۲۰ کروموزوم بعدی را حذف می‌کنیم. علت این امر این است که هر چه سازگاری کمتر

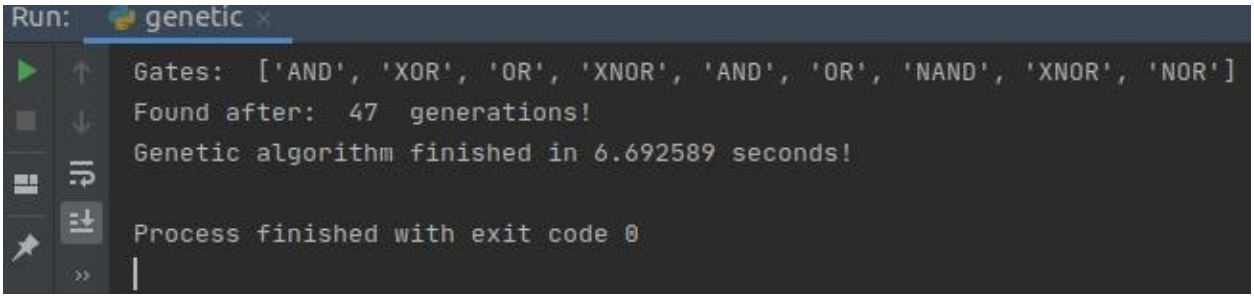
باشد راه حل دورتر از پاسخ درست است و در صورت حذف آنها و کار کردن با کروموزوم های سازگار تر سرعت بیشتری برای رسیدن به جواب خواهیم داشت. بین ۳۰ کروموزوم برتر هم، توجه بیشتری به ۱۰ تای اول شده و بیشتر تغییرات روی آنها صورت گرفته چون آنها به پاسخ بسیار نزدیک هستند و نیاز به تغییرات کمی برای رسیدن به جواب دارند. پس حالت های مختلف تغییرات را روی آنها آزمایش می‌کنیم.

۳) mutation می تواند یک کروموزوم را دگرگون کند و تغییر زیادی در سازگاری آن به وجود آورد. برای این تابع در برنامه ۲ احتمال مختلف در نظر گرفته شده است. برای ۲۰ عنصر اول که هم جهش و هم crossover روی آن ها رخ می‌دهد، احتمال پایین تری در نظر گرفته شده چون احتمال بالای جهش، ممکن است کروموزوم های نزدیک به جواب را از جواب دور کند. این احتمال در یک hyper parameter به نام SINGLE_GENE_MUTATION_LOW_POSSIBILITY مقدار دهی شده و مقدار آن ۱۵درصد است که احتمال جهش یافتن یک ژن از یکی از کروموزوم های crossover شده ی برتر است.

دیگر احتمال جهش مربوط به جهش یافتن کروموزوم های ۲۰ تا ۳۰ است که احتمال جهش هر ژن در آن کروموزوم ها ۴۰ درصد (SINGLE_GENE_MUTATION_HIGH_POSSIBILITY) تعیین شده است. بیش از احتمال قبلی است چون این ها نیاز به جهش و تغییر بیشتری دارند.

احتمال crossover نیز توضیح داده شده است که برای بعضی از کروموزوم ها انجام می شود و اینکه کدام دو کروموزوم با هم crossover کنند به طور شانسی و نقطه آن نیز به طور شانسی مشخص می شود.

۴) دلیل مشکلات موجود در برنامه و اینکه جواب هایی که می گیریم این است که پایه و اساس این الگوریتم بر اساس شانس است و در بسیاری از مواقع ممکن است خوش شانس باشیم و در زودترین موقع ممکن به جواب برسیم و جواب های پیشنهادی ما به سرعت بهتر شوند یا برعکس و هردوی این حالات ممکن است در شرایط مشابه با هم رخ بدهند.



هایپر پارامتر های دیده شده در تصویر زیر نیز بعضی در بالا توضیح داده شده اند. متغیر های TO_BE_MUTATED_FIRST و TO_BE_MUTATED_LAST نیز به طور مثال بازه های جمعیت که جهش (با احتمال بالا) روی آنها رخ می‌دهد را نشان می‌دهند.

```
GATES = ["AND", "OR", "XOR", "NAND", "NOR", "XNOR"]
SURVIVING_CHROMOSOMES_NUM = 10
MUT_AND_CROSS_FROM_FIRST = 20
TO_BE_CROSSED_OVER_FIRST = 20
TO_BE_CROSSED_OVER_LAST = 30
TO_BE_MUTATED_FIRST = 20
TO_BE_MUTATED_LAST = 30
SINGLE_GENE_MUTATION_HIGH_POSSIBILITY = 0.4
SINGLE_GENE_MUTATION_LOW_POSSIBILITY = 0.15
POPULATION = 50
```