باسمه تعالی

گزارش تمرین کدی هوش مصنوعی

عرفان رفیعی اسکویی - 98243027

مقدمه

در این پروژه هدف پیدا کردن پاسخ یک عبارت منطقی در قالب CNF می‌باشد. تعداد ۴۲۹ عبارت با سه متغیر در فایل ورودی وجود دارد. تعداد کل متغیرها برابر با ۱۰۰ است. فایل ورودی در قالب DIMACS می‌باشد که نوع استاندارد ذخیره‌سازی قالب CNF می‌باشد. برای خواندن این فایل از کتابخانه‌ی python-sat استفاده شده است. لازم به ذکر است نیازی به استفاده از این کتابخانه نبود، صرفا جهت آشنایی بیشتر از آن استفاده شده است.

برای نصب این کتابخانه ابتدا مراحل زیر را طی میکنیم :

- Open `File > Settings > Project` from the PyCharm menu.
- Select your current project.
- Click the `Python Interpreter` tab within your project tab.
- Click the small `+` symbol to add a new library to the project.
- Now type in the library to be installed, for example Pandas, and click `Install Package`.
- Wait for the installation to terminate and close all popup windows.

سپس در قسمت package کتابخانه python-sat را سرچ کرده و نصب میکنیم.

پس از نصب این کتابخانه قالب CNF با دستور زیر در برنامه استفاده می‌شود:

```
from pysat.formula import CNF
```

با استفاده از این کتابخانه یک شی CNF ساخته می‌شود و سپس به گزاره‌های آن به راحتی دسترسی پیدا می‌کنیم.

```
lines = np.loadtxt("database.cnf",dtype=int, delimiter="  ", unpack=False)
cnf = CNF()
for line in lines:
    cnf.append(line[:-1].tolist())
```

برای دسترسی به گزاره‌ها از عبارت cnf.clauses و برای دسترسی به تعداد متغیرها (۱۰۰) بیت از عبارت cnf.nv استفاده می‌کنیم.

بررسی خروجی یک عبارت منطقی

یک عبارت منطقی با n متغیر می‌تواند ۲ به توان n ورودی داشته باشد و به ازای هر ورودی تعدادی از گزاره‌ها درست و تعداد غلط می‌شوند. حتی در صورت وجود یک گزاره غلط در قالب CNF نتیجه نهایی غلط است. اندازه‌گیری تعداد گزاره‌های درست که معیار کارکرد الگوریتم‌ها است توسط تابع check_satisfation اندازه گیری می‌شود.

این تابع دو ورودی دارد:

- گزاره‌ها: لیستی از گزاره‌های مربوط به عبارت منطقی؛ در این برنامه یک لیست ۴۲۹ تایی از گزاره‌های ۳ تایی.
- حالت: یکی از ۲ به توان n حالتی که ورودی‌ها می‌توانند داشته باشند. برای راحتی کار، در این برنامه ورودی به شکل یک آرایه صد تایی است. هر عنصر آرایه در ایندکس i می‌تواند عبارت i یا i- باشد. که اولی یعنی خود متغیر و دومی یعنی نقیض متغیر استفاده شده است. به طور مثال عبارت زیر یک ورودی برای حالت است:

[-1, -2, 3, 4, 5, -6, -7, -8, -9, -10, -11, -12, -13, -14, 15, 16, -17, 18, -19, 20, 21, -22, 23, 24, -25, 26, 27, 28, -29, -30, 31, -32, -33, -34, -35, -36, 37, 38, 39, 40, 41, 42, 43, 44, -45, -46, -47, -48, -49, 50, 51, 52, -53, -54, 55, 56, 57, -58, 59, -60, -61, -62, 63, 64, -65, 66, 67, 68, -69, 70, 71, 72, 73, -74, -75, -76, -77, -78, 79, -80, 81, 82, 83, 84, 85, -86, -87, -88, -89, -90, 91, -92, 93, -94, 95, -96, 97, 98, -99, -100]

random_solution می‌تواند یک حالت با n متغیر تصادفی ایجاد نماید.

پیاده‌سازی الگوریتم ژنتیک

الگوریتم ژنتیک به صورت توابع مجزا پیاده‌سازی شده است. این توابع عبارتند از:

- population_sort
- population_cross
- clone

- mutate
- fitness

هر یک از توابع در تابع اصلی genetic فراخوانی می‌شوند. ورودی تابع ژنتیک به شکل زیر است:

- Clauses: گزاره‌های مربوط به عبارت منطقی
- nbL: تعداد متغیرهای مربوط به عبارت منطقی
- k: جمعیت اولیه
- nbIter: تعداد تکرار
- rate: نرخ جهش (بین صفر و یک)

فراخوانی این تابع به شکل زیر است:

```
genetic(cnf.clauses, cnf.nv, 1000, 10000000000,0.20)
```

در نهایت در بهترین حالت به دقت ۹۹ درصد می‌رسیم.

پیاده‌سازی الگوریتم شبیه‌سازی حرارتی

الگوریتم شبیه‌سازی حرارتی به صورت توابع مجزا پیاده‌سازی شده است. این توابع عبارتند از:

- disturb
- next_temperature

تابع اول با هدف ایجاد یک ورودی جدید استفاده می‌شود که به صورت تصادفی تولید می‌شود. تابع دوم با استفاده از فرمول

$$temrature = 1 - k\ /k_{max}$$

مقدار حرارت را به روز می‌کند. مقدار k برابر تکرار فعلی و مقدار k_max معادل کل تکرارها می‌باشد.

در نهایت تابع simulated_annealing به صورت زیر عمل می‌کند:

- یک راه‌حل تصادفی می‌سازد؛ با استفاده از تابع random_solution
- با استفاده از check_satisfation امتیاز این راه‌حل را حساب می‌کند (اگر راه‌حل صحیح باشد یعنی ۴۲۹ گزاره درست هستند، در غیر این صورت عددی کمتر از ۴۲۹ به دست می‌آید).
- وارد یک حلقه می‌شود که به تعداد n_iterations تکرار می‌شود. (برای محاسبه دما n_iterations همان k_max است).
- به ازای هر تکرار یک راه‌حل تصادفی تولید می‌شود و امتیاز آن محاسبه می‌شود. در صورتی که امتیاز از راه‌حل قبلی بیشتر باشد با احتمال وابسته به دما این راه حل جایگزین می‌شود:

```
if delta <= 0 or random() < np.exp(-delta/temperature)
```

- در نهایت با این الگوریتم به دقت حدود ۹۵ درصد می‌رسیم. یعنی در حدود ۴۱۰ گزاره درست از ۴۲۹ گزاره.