Institute for Advanced Studies
in Basic Sciences
GavaZang, Zanjan, Iran

Rastrigin Metaheuristic Optimization

Prof. Majid Ramezani

Report for Project <2>

Faeze Ahmadi

December 2025

بهینه‌سازی تابع Rastrigin با استفاده از الگوریتم‌های Metaheuristic

کد کامل این پروژه و فایل‌های مربوطه در ریپازیتوری گیت‌هاب قابل دسترسی هست:

https://github.com/Faeze-Ahmadi/rastrigin-metaheuristic-optimization

این پروژه اصلا چی هست و چرا نوشته شده؟

این پروژه با هدف آشنایی عملی با الگوریتم‌های فراابتکاری در درس هوش مصنوعی نوشته شده است. ایده اصلی پروژه این است که نشان بدهد وقتی با یک مسئله بهینه‌سازی سخت و پر از مینیمم‌های محلی طرف هستیم، روش‌های ساده و سرراست معمولا جواب خوبی نمی‌دهند و لازم است سراغ الگوریتم‌هایی برویم که بتوانند از تله‌های محلی فرار کنند و به جواب‌های بهتر نزدیک شوند.

برای این کار، از یک تابع معروف به نام تابع Rastrigin استفاده شده است. این تابع عمدا طوری طراحی شده که سطحش پر از پستی و بلندی باشد و تعداد زیادی نقطه کمینه محلی داشته باشد. به همین دلیل، می‌شود به‌خوبی عملکرد الگوریتم‌های مختلف را روی آن مقایسه کرد و دید کدام روش‌ها واقعا توانایی رسیدن به کمینه سراسری را دارند.

در این پروژه چند الگوریتم مختلف پیاده‌سازی شده‌اند؛ از روش‌های ساده‌تر مثل Hill Climbing گرفته تا روش‌های پیشرفته‌تر مثل Simulated Annealing و Particle Swarm Optimization. علاوه بر پیدا کردن جواب نهایی، تمرکز اصلی پروژه روی این بوده که مسیر حرکت الگوریتم‌ها هم قابل مشاهده باشد، تا بشود به‌صورت بصری دید هر الگوریتم چطور در فضای مسئله حرکت می‌کند و چگونه تصمیم می‌گیرد.

هدف نهایی این پروژه فقط رسیدن به یک عدد نیست، بلکه فهمیدن رفتار الگوریتم‌ها، تفاوت نگاه آن‌ها به مسئله و درک شهودی از مفهوم بهینه‌سازی در هوش مصنوعی است. به همین دلیل، بخش تصویرسازی و انیمیشن نقش مهمی در پروژه دارد.

ساختار کلی پروژه

پروژه به شکلی سازمان‌دهی شده که هر بخش وظیفه مشخص خودش را داشته باشد. فایل‌هایی که مربوط به خود الگوریتم‌ها هستند از فایل‌هایی که مربوط به نمایش نتایج و ذخیره‌سازی خروجی‌ها هستند جدا شده‌اند. این کار باعث می‌شود هم خوانایی پروژه بالا برود و هم اگر بخواهیم بعدا الگوریتم جدیدی اضافه کنیم یا نمایش‌ها را تغییر بدهیم، مجبور نشویم همه‌چیز را به‌هم بریزیم.

```
rastrigin-metaheuristic-optimization/
├ src/
| ├ rastrigin.py # Rastrigin function + 2D landscape plot
| ├ hill_climbing.py # Hill Climbing + Random Restart
| ├ simulated_annealing.py # Simulated Annealing + SA with restarts
| ├ pso.py # Particle Swarm Optimization (PSO)
| |
| ├ animate_sa_light.py # Low-memory SA convergence GIF (recommended)
| ├ animate_sa_heavy.py # High-quality SA animation (MP4 / heavy)
| ├ animate_pso_light.py # Low-memory PSO convergence GIF
| |
| └ utils.py # Shared utilities (bounds, clamp, stats runner)
|
├ visualizations/
| ├ simulated_annealing_path.png # Final SA path on Rastrigin landscape
| ├ sa_convergence.gif # Simulated Annealing convergence animation
| └ pso_convergence.gif # PSO convergence animation
|
├ results/
| └ sa_best.txt # Saved best objective value (SA)
|
├ report/
| ├ report.md # Short academic-style report
| └ .gitkeep # Keeps folder tracked in Git
|
├ README.md
├ requirements.txt
└ LICENSE
```

تابع Rastrigin چیست و چرا برای این پروژه انتخاب شده؟

تابع Rastrigin یکی از معروف‌ترین توابع آزمایشی در دنیای بهینه‌سازی است. این تابع عمدا طوری طراحی شده که حل کردنش سخت باشد، نه به این خاطر که محاسباتش پیچیده است، بلکه چون سطحش پر از دام و تله است. اگر از بالا به آن نگاه کنیم، شبیه یک زمین پر از موج، پستی و بلندی و چاله‌های ریز و درشت است که تقریبا همه‌جایش شبیه به کمینه به نظر می‌رسد، اما فقط یک نقطه واقعا بهترین است.

ویژگی اصلی تابع Rastrigin این است که تعداد بسیار زیادی کمینه محلی دارد. یعنی اگر الگوریتمی فقط اطراف خودش را نگاه کند، خیلی سریع به یک نقطه می‌رسد که از نظر محلی بهترین است و دیگر جلوتر نمی‌رود، حتی اگر جواب نهایی‌اش اصلا خوب نباشد. به همین دلیل، این تابع تبدیل شده به یک معیار استاندارد برای تست الگوریتم‌های بهینه‌سازی.

از نظر مفهومی، Rastrigin ترکیبی از یک بخش ساده و یک بخش نوسانی است. بخش ساده باعث می‌شود هرچه از مرکز دورتر می‌شویم، مقدار تابع بزرگ‌تر شود، اما بخش نوسانی باعث می‌شود روی این سطح کلی، موج‌های ریز و درشت ایجاد شود. همین موج‌ها هستند که کمینه‌های محلی را می‌سازند و الگوریتم‌ها را به دام می‌اندازند.

کمینه سراسری این تابع در مرکز فضا قرار دارد. یعنی بهترین جواب دقیقا در نقطه‌ای است که همه متغیرها صفر هستند. نکته جالب اینجاست که رسیدن به این نقطه از نظر محاسباتی ساده است، اما از نظر مسیر رسیدن بسیار سخت است، چون الگوریتم باید بارها تصمیم بگیرد از چاله‌های به‌ظاهر خوب خارج شود و به مسیرهای نامطمئن‌تر برود.

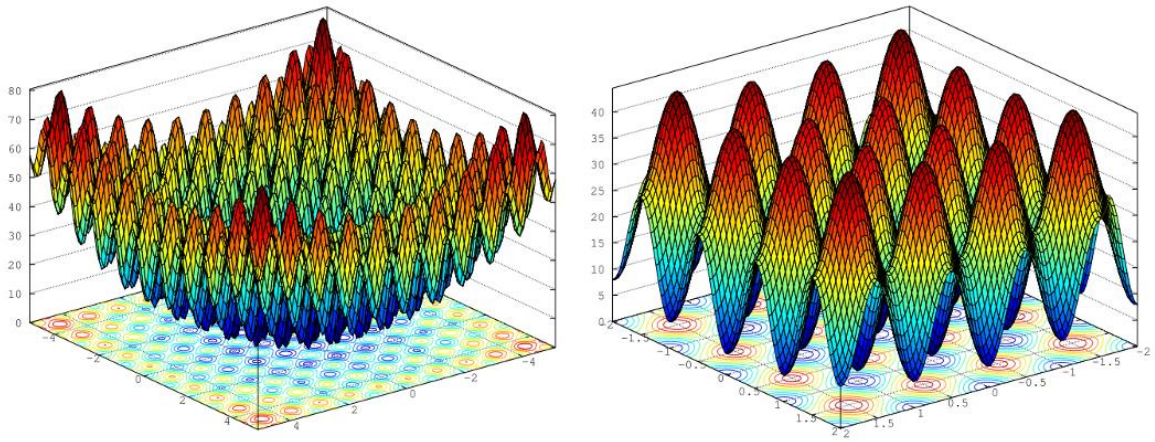
دلیل انتخاب Rastrigin برای این پروژه دقیقا همین ویژگی‌هاست. این تابع به‌خوبی نشان می‌دهد کدام الگوریتم‌ها فقط ظاهر مسئله را می‌بینند و کدام‌ها واقعا می‌توانند فضای مسئله را بگردند. وقتی مسیر حرکت

الگوریتم‌ها روی این تابع رسم می‌شود، می‌توان به‌صورت بصری دید که چرا بعضی روش‌ها گیر می‌افتند و بعضی دیگر با وجود مسیرهای پیچیده، به جواب درست نزدیک می‌شوند.

در این پروژه، Rastrigin نقش یک زمین تمرینی سخت را دارد؛ زمینی که اگر الگوریتم بتواند روی آن خوب عمل کند، می‌شود به توانایی‌اش در مسائل واقعی هم اعتماد بیشتری داشت. به همین دلیل است که این تابع هنوز هم بعد از سال‌ها یکی از محبوب‌ترین ابزارها برای ارزیابی الگوریتم‌های بهینه‌سازی محسوب می‌شود.

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$$

Dimensions:	n
Domain:	$-5.12 \leq x_i \leq 5.12$
Global Optimum:	$f(x) = 0.0$ at $x = (0.0, 0.0, \ldots, 0.0)$
Operator:	RastriginEvaluator
Charts:	



(a) [-5.12, 5.12] (b) [-2.0, 2.0]

Figure 7: Rastrigin function plots.

توضیح فایل مربوط به تابع Rastrigin

یکی از فایل‌های اصلی پروژه فایلی است که تابع Rastrigin را تعریف می‌کند. در این فایل، خود تابع هدف پیاده‌سازی شده؛ یعنی همان تابعی که الگوریتم‌ها تلاش می‌کنند مقدارش را کمینه کنند. علاوه بر خود تابع، یک

بخش مربوط به رسم نمای دوبعدی این تابع هم وجود دارد تا بتوان سطح تابع را دید و فهمید چرا این مسئله تا این حد سخت است.

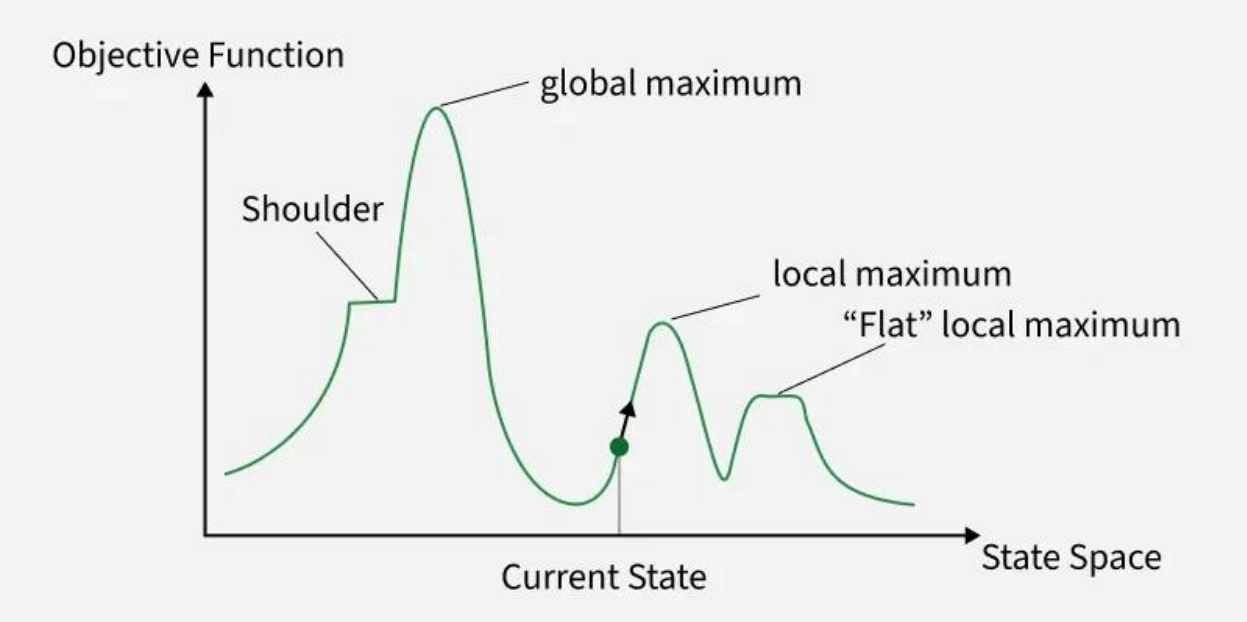
این فایل عملا نقش زمین بازی را دارد. همه الگوریتم‌ها روی همین زمین حرکت می‌کنند و اگر این تابع نبود، مقایسه و تحلیل رفتار الگوریتم‌ها معنایی نداشت. رسم دوبعدی تابع کمک می‌کند حتی بدون نگاه کردن به کد، فقط با دیدن تصویر بفهمیم چرا الگوریتم‌ها ممکن است گیر بیفتند یا مسیرهای عجیب‌وغریب طی کنند.

الگوریتم Hill Climbing چیست و چطور کار می‌کند؟

Hill Climbing ساده‌ترین الگوریتمی است که می‌شود برای بهینه‌سازی تصور کرد. ایده‌اش دقیقا شبیه این است که روی یک زمین پستی‌وبلندی‌دار ایستاده باشیم و بخواهیم به پایین‌ترین نقطه برسیم. الگوریتم از یک نقطه شروع حرکت می‌کند و در هر مرحله فقط به اطراف خودش نگاه می‌کند و می‌گوید: «آیا جایی هست که از اینجا بهتر باشد؟» اگر پیدا کند، به همان سمت می‌رود و این کار را ادامه می‌دهد.

مسئله اصلی Hill Climbing این است که هیچ دید کلی نسبت به کل زمین ندارد. یعنی اگر به نقطه‌ای برسد که اطرافش همه بدتر از خودش باشند، تصور می‌کند بهترین جای ممکن را پیدا کرده، در حالی که شاید آن نقطه فقط یک کمینه محلی باشد و پایین‌تر از آن، جای خیلی بهتری وجود داشته باشد. به همین دلیل، این الگوریتم خیلی زود گیر می‌افتد.

در این پروژه، Hill Climbing بیشتر برای این استفاده شده که نشان بدهد چرا روش‌های ساده برای توابعی مثل Rastrigin جواب خوبی نمی‌دهند. دیدن شکست‌های این الگوریتم کمک می‌کند بفهمیم چرا به الگوریتم‌های پیشرفته‌تر نیاز داریم.



توضیح فایل مربوط به الگوریتم Hill Climbing

در فایل مربوط به Hill Climbing، یکی از ساده‌ترین الگوریتم‌های بهینه‌سازی پیاده‌سازی شده است. این الگوریتم از یک نقطه شروع حرکت می‌کند و در هر مرحله سعی می‌کند به یک نقطه بهتر در اطراف خودش برود. اگر جایی برسد که هیچ همسایه بهتری نداشته باشد، متوقف می‌شود.

در این پروژه از نسخه‌ای استفاده شده که چند بار از نقاط شروع تصادفی اجرا می‌شود تا شانس رسیدن به جواب بهتر کمی بیشتر شود. با این حال، این فایل به‌خوبی نشان می‌دهد که چرا چنین روش‌هایی برای توابعی مثل Rastrigin چندان مناسب نیستند و خیلی زود در مینیمم‌های محلی گیر می‌افتند. این بخش بیشتر جنبه آموزشی دارد و نقش مقایسه‌ای مهمی در پروژه بازی می‌کند.

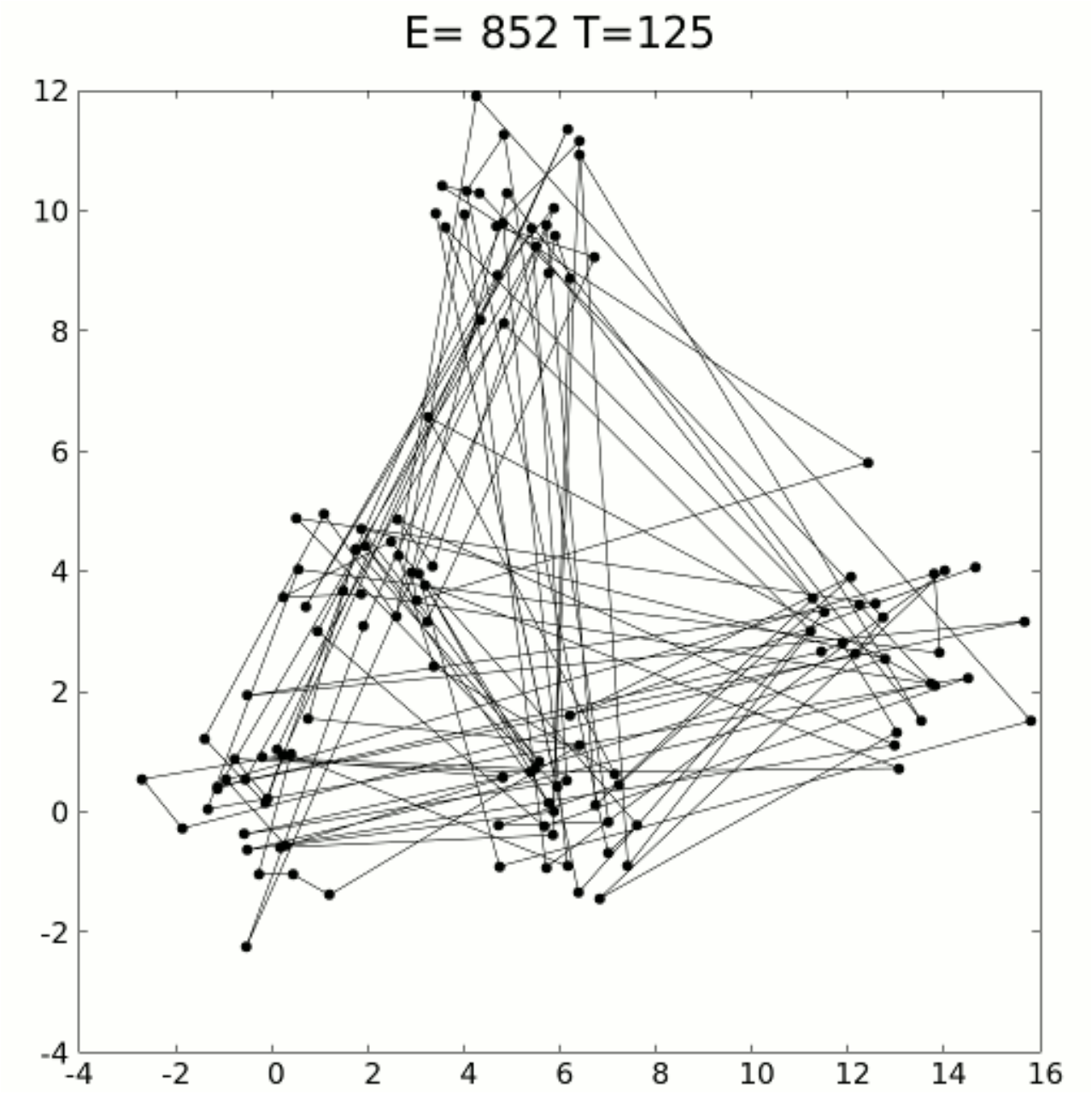
الگوریتم Simulated Annealing چیست و چرا مهم است؟

Simulated Annealing از یک ایده فیزیکی الهام گرفته شده؛ فرایند سرد شدن فلزات. وقتی فلز داغ است، اتم‌ها آزادی حرکت زیادی دارند و می‌توانند حالت‌های نامنظم را تجربه کنند. هرچه فلز سردتر می‌شود، این آزادی کمتر می‌شود و ساختار منظم‌تری شکل می‌گیرد.

در این الگوریتم هم دقیقا همین اتفاق می‌افتد. در ابتدای کار، الگوریتم اجازه دارد حتی به جواب‌های بدتر هم حرکت کند. این کار عمدا انجام می‌شود تا از مینیمم‌های محلی فرار کند. هرچه زمان می‌گذرد، این آزادی کمتر می‌شود و الگوریتم سخت‌گیرتر می‌شود و فقط حرکت‌های بهتر یا نزدیک به بهتر را قبول می‌کند.

فرمول اصلی Simulated Annealing بر اساس مقایسه کیفیت جواب فعلی و جواب جدید است. اگر جواب جدید بهتر باشد، بدون چون‌وچرا قبول می‌شود. اگر بدتر باشد، با یک احتمال مشخص که به دما بستگی دارد ممکن است پذیرفته شود. این احتمال هرچه دما پایین‌تر می‌آید کمتر و کمتر می‌شود.

دلیل اینکه Simulated Annealing به درد ما می‌خورد این است که دقیقا برای مسئله‌هایی طراحی شده که پر از مینیمم محلی هستند. تابع Rastrigin یکی از بهترین مثال‌ها برای این حالت است و این الگوریتم نشان می‌دهد که چگونه می‌شود با کمی «ریسک کنترل‌شده» به جواب‌های خیلی بهتری رسید.


E= 852 T=125
12
10
8
6
4
2
0
-2
-4
-4
-2
0
2
4
6
8
10
12
14
16

توضیح فایل مربوط به Simulated Annealing

Simulated Annealing یکی از مهم‌ترین بخش‌های پروژه است. در فایل مربوط به این الگوریتم، روشی پیاده‌سازی شده که برخلاف Hill Climbing، گاهی اجازه می‌دهد به جواب‌های بدتر هم حرکت کند. این رفتار باعث می‌شود الگوریتم بتواند از مینیمم‌های محلی خارج شود و فضای مسئله را بهتر بگردد.

در این فایل، علاوه بر منطق اصلی الگوریتم، به محدود کردن فضای جستجو و تنظیم اندازه حرکت‌ها هم توجه شده است تا الگوریتم رفتار واقع‌بینانه‌تری داشته باشد. همچنین امکان اجرای الگوریتم از چند نقطه شروع مختلف وجود دارد تا نتیجه نهایی قابل اعتمادتر شود.

این فایل عملا نشان می‌دهد که چگونه یک ایده ساده، یعنی پذیرش موقت جواب‌های بدتر، می‌تواند تفاوت بزرگی در کیفیت جواب نهایی ایجاد کند.

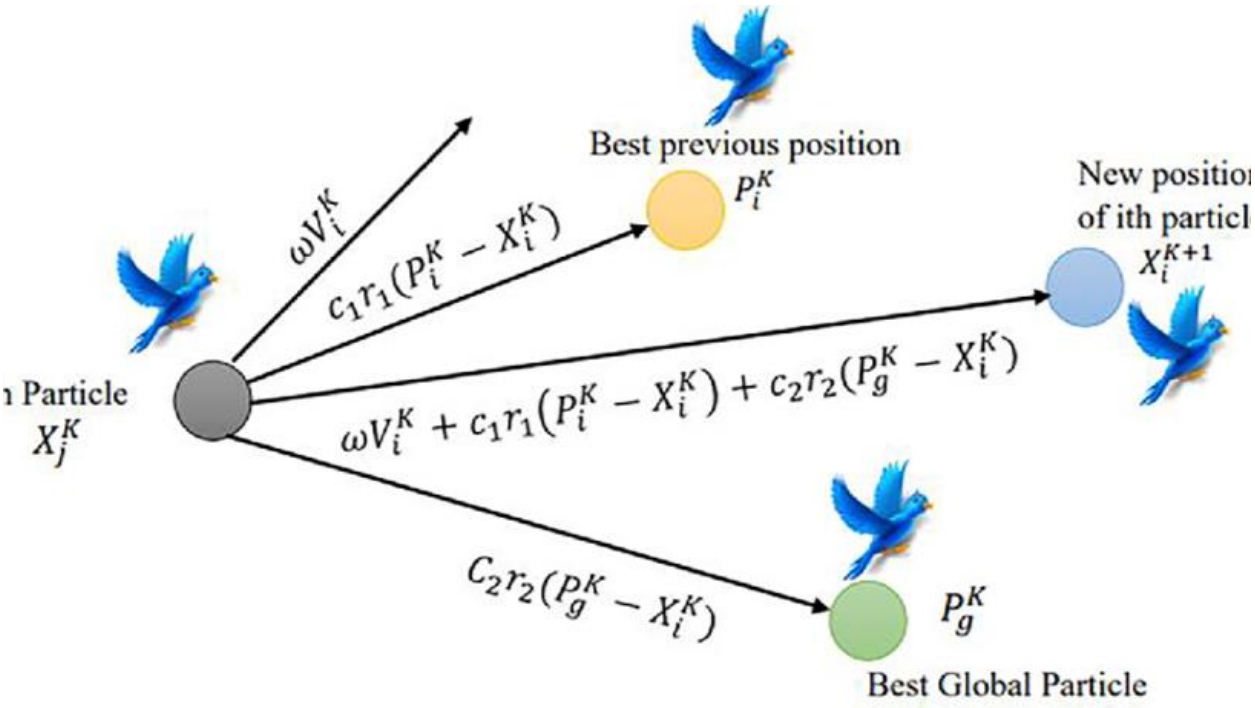
الگوریتم Particle Swarm Optimization (PSO) چیست؟

PSO کاملا نگاه متفاوتی به مسئله دارد. به‌جای اینکه یک نقطه در فضا حرکت کند، این الگوریتم با یک جمع از ذرات کار می‌کند. هر ذره یک جواب ممکن است و همه ذرات هم‌زمان در فضای مسئله حرکت می‌کنند.

هر ذره دو چیز را همیشه به خاطر می‌سپارد: بهترین جایی که خودش تا الان دیده و بهترین جایی که کل جمع ذرات پیدا کرده‌اند. حرکت ذره ترکیبی از این دو عامل است؛ یعنی هم تجربه شخصی خودش روی تصمیمش تأثیر می‌گذارد و هم تجربه جمع.

فرمول حرکت ذرات شامل سه بخش اصلی است. یک بخش مربوط به سرعت قبلی ذره است که باعث می‌شود حرکتش خیلی ناگهانی تغییر نکند. یک بخش مربوط به تمایل ذره برای برگشتن به بهترین تجربه شخصی خودش است و بخش دیگر مربوط به کشش به سمت بهترین جواب جمعی است. ترکیب این سه عامل باعث می‌شود ذرات هم فضا را بگردند و هم کم‌کم دور جواب‌های خوب جمع شوند.

PSO برای این پروژه خیلی ارزشمند است چون رفتار آن کاملا قابل مشاهده است. وقتی انیمیشن آن را می‌بینیم، به‌وضوح مشخص است که ذرات ابتدا پراکنده‌اند و بعد از مدتی شروع می‌کنند به همگرا شدن به سمت یک ناحیه خاص. این رفتار جمعی باعث می‌شود الگوریتم بدون نیاز به منطق‌های پیچیده تصادفی، به جواب‌های خیلی خوبی برسد.



توضیح فایل مربوط به الگوریتم Particle Swarm Optimization

در فایل مربوط به Particle Swarm Optimization، یک الگوریتم جمعی پیاده‌سازی شده است. در این روش به‌جای اینکه فقط یک نقطه در فضا حرکت کند، چندین ذره به‌طور هم‌زمان حرکت می‌کنند. هر ذره هم تجربه شخصی خودش را در نظر می‌گیرد و هم به بهترین تجربه‌ای که کل جمع پیدا کرده توجه می‌کند.

این فایل نشان می‌دهد که چگونه همکاری غیرمستقیم بین ذرات می‌تواند به پیدا کردن جواب‌های خوب منجر شود. رفتار این الگوریتم از نظر بصری بسیار جذاب است، چون می‌شود دید که ذرات کم‌کم دور یک ناحیه خاص جمع می‌شوند و به سمت کمینه سراسری همگرا می‌شوند.

Particle Swarm Optimization در این پروژه نقش مهمی در مقایسه الگوریتم‌ها دارد و به‌خوبی نشان می‌دهد که نگاه جمعی چه تفاوتی با نگاه تک‌نقطه‌ای دارد.

چرا این الگوریتم‌ها برای این پروژه انتخاب شده‌اند؟

این سه الگوریتم کنار هم یک داستان کامل را تعریف می‌کنند. Hill Climbing نشان می‌دهد نگاه کاملا محلی چقدر محدود است. Simulated Annealing نشان می‌دهد با اضافه کردن کمی شانس و انعطاف، چطور می‌شود از این محدودیت عبور کرد. PSO هم نشان می‌دهد که همکاری و نگاه جمعی چقدر می‌تواند قدرتمند باشد.

ترکیب این سه روش کمک می‌کند مفهوم بهینه‌سازی در هوش مصنوعی نه فقط به‌صورت تئوری، بلکه به‌صورت شهودی و قابل لمس درک شود. دیدن مسیر حرکت الگوریتم‌ها روی یک تابع پیچیده، درک عمیق‌تری از تصمیم‌گیری الگوریتم‌ها می‌دهد که با خواندن فرمول‌ها به‌تنهایی به دست نمی‌آید.

توضیح فایل‌های مربوط به تصویرسازی و انیمیشن

برای اینکه رفتار الگوریتم‌ها فقط در حد عدد باقی نماند، فایل‌هایی جداگانه برای ساخت انیمیشن در نظر گرفته شده‌اند. در این فایل‌ها، مسیر حرکت الگوریتم‌ها به‌صورت تدریجی روی سطح تابع نمایش داده می‌شود و خروجی به شکل فایل تصویری متحرک ذخیره می‌شود.

نسخه‌های سبک این انیمیشن‌ها طوری طراحی شده‌اند که مصرف حافظه کمی داشته باشند و روی سیستم‌های معمولی هم بدون مشکل اجرا شوند. این فایل‌ها کمک می‌کنند رفتار الگوریتم‌ها به‌صورت شهودی درک شود و صرفا به نتیجه نهایی نگاه نکنیم.

فایل ابزارهای کمکی

در فایل ابزارهای کمکی، یک‌سری توابع عمومی قرار داده شده که در بخش‌های مختلف پروژه استفاده می‌شوند. این توابع برای محدود کردن مقادیر، مدیریت تکرار آزمایش‌ها و محاسبه آمار نتایج به کار می‌روند. وجود این فایل باعث شده کدهای اصلی ساده‌تر و خواناتر باشند و از تکرار جلوگیری شود.

این پروژه در نهایت چه خروجی به ما می‌دهد؟

یکی از سؤال‌هایی که ممکن است برای خواننده پیش بیاید این است که بعد از اجرای این پروژه دقیقا باید دنبال چه چیزی بگردد. آیا هدف فقط رسیدن به یک عدد خاص است یا چیز دیگری هم اهمیت دارد؟ واقعیت این است که خروجی این پروژه فقط یک مقدار نهایی نیست. در کنار عددی که نشان می‌دهد الگوریتم به چه مقدار بهینه‌ای رسیده، مسیر حرکت الگوریتم هم اهمیت زیادی دارد.

در این پروژه، هم خروجی عددی داریم که کیفیت جواب نهایی را نشان می‌دهد و هم خروجی تصویری که مسیر حرکت الگوریتم را روی سطح تابع نمایش می‌دهد. این تصاویر و انیمیشن‌ها کمک می‌کنند بفهمیم الگوریتم چطور تصمیم می‌گیرد، کجا گیر می‌افتد، از کجا فرار می‌کند و چطور به جواب نهایی نزدیک می‌شود. به‌عبارت دیگر، پروژه فقط به جواب نگاه نمی‌کند، بلکه به «روند رسیدن به جواب» هم توجه دارد.

هدف پروژه مقایسه الگوریتم‌هاست یا نشان دادن رفتار آن‌ها؟

نکته مهمی که لازم است شفاف گفته شود این است که این پروژه قرار نیست اعلام کند کدام الگوریتم بهترین است. هدف اصلی، مقایسه خشک و عددی نیست. بلکه هدف این است که رفتار الگوریتم‌ها در مواجهه با یک مسئله سخت دیده و درک شود.

هر الگوریتم نگاه خاص خودش را به مسئله دارد. بعضی کاملا محلی تصمیم می‌گیرند، بعضی با شانس و ریسک جلو می‌روند و بعضی به‌صورت جمعی فکر می‌کنند. این پروژه سعی می‌کند این تفاوت نگاه‌ها را نشان بدهد، نه اینکه صرفا یک برنده معرفی کند. اگر خواننده با این دید پروژه را بخواند، برداشت دقیق‌تر و عمیق‌تری خواهد داشت.

نقش تصادفی بودن در این الگوریتم‌ها چیست؟

یکی از چیزهایی که ممکن است خواننده را سردرگم کند این است که چرا هر بار اجرای پروژه ممکن است نتیجه یا مسیر کمی متفاوت داشته باشد. دلیلش این است که الگوریتم‌های استفاده‌شده در این پروژه ذاتا تصادفی هستند. این تصادفی بودن نه‌تنها ضعف نیست، بلکه یکی از نقاط قوت اصلی این الگوریتم‌هاست.

تصادفی بودن باعث می‌شود الگوریتم‌ها اسیر یک مسیر خاص نشوند و بتوانند فضای مسئله را بهتر بگردند. در واقع اگر همه‌چیز کاملا قابل پیش‌بینی و ثابت بود، الگوریتم‌ها خیلی زود در مینیمم‌های محلی گیر می‌افتادند. بنابراین تغییرپذیری نتایج نشانه عملکرد بد نیست، بلکه بخشی طبیعی از رفتار این روش‌هاست.

چرا در این پروژه روی تصویرسازی و انیمیشن تاکید شده است؟

در بسیاری از پروژه‌های بهینه‌سازی، فقط عدد نهایی گزارش می‌شود. اما در این پروژه عمدا تمرکز زیادی روی تصویرسازی گذاشته شده است. دلیلش این است که دیدن رفتار الگوریتم‌ها، مخصوصا برای یادگیری، بسیار ارزشمندتر از دیدن یک عدد خشک است.

وقتی مسیر حرکت یک الگوریتم روی تابع رسم می‌شود یا حرکت ذرات در PSO دیده می‌شود، درک مفاهیمی مثل مینیمم محلی، فرار از تله‌ها و همگرایی خیلی راحت‌تر می‌شود. این تصویرسازی‌ها کمک می‌کنند مفاهیم انتزاعی هوش مصنوعی به چیزی قابل لمس و قابل فهم تبدیل شوند.

این پروژه چه محدودیت‌هایی دارد؟

این پروژه با هدف آموزشی نوشته شده است، نه برای استفاده صنعتی یا تحقیقاتی در مقیاس بزرگ. به همین دلیل، مسئله به‌صورت دوبعدی در نظر گرفته شده تا امکان تصویرسازی و درک بهتر فراهم شود. همچنین تنظیم پارامترها به شکلی انجام شده که رفتار الگوریتم‌ها قابل مشاهده باشد، نه اینکه حتما بهترین تنظیم ممکن باشند.

الگوریتم‌ها در این پروژه ساده‌سازی شده‌اند تا خوانایی و فهم آن‌ها اولویت داشته باشد. در دنیای واقعی، نسخه‌های پیچیده‌تر و بهینه‌تری از این روش‌ها وجود دارد، اما برای هدف این پروژه، تمرکز روی فهم مفهومی و شهودی بوده است.

اگر کسی بخواهد این پروژه را اجرا کند، چه انتظاری باید داشته باشد؟

اجرای این پروژه معمولا بدون مشکل خاصی انجام می‌شود، اما بعضی بخش‌ها مثل ساخت انیمیشن ممکن است زمان‌بر باشند یا منابع بیشتری مصرف کنند. به همین دلیل نسخه‌های سبک‌تر برای انیمیشن‌ها در نظر گرفته شده‌اند تا روی سیستم‌های معمولی هم قابل اجرا باشند.

همچنین باید انتظار داشت که اجرای پروژه در دفعات مختلف، دقیقا مسیر یکسانی تولید نکند. این رفتار طبیعی است و بخشی از ماهیت الگوریتم‌های استفاده‌شده محسوب می‌شود.

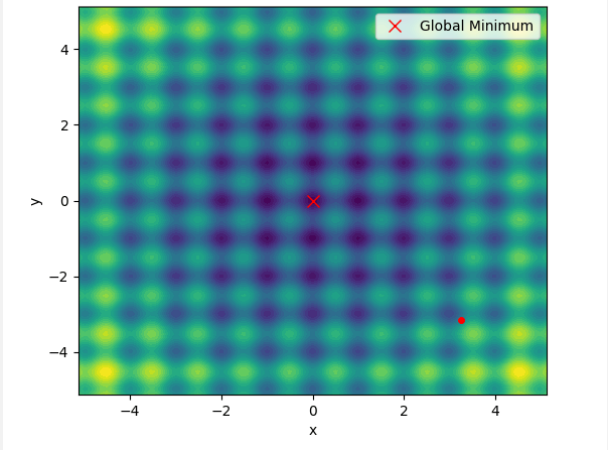
جمع‌بندی نهایی

این پروژه فقط تمرینی برای نوشتن کد نبوده، بلکه تلاشی برای درک عمیق‌تر مفهوم بهینه‌سازی در هوش مصنوعی است. کار با این الگوریتم‌ها کمک می‌کند بفهمیم چرا بعضی مسائل ساده به نظر می‌رسند اما حل کردنشان سخت است، و چرا گاهی باید به الگوریتم‌ها اجازه داد اشتباه کنند تا در نهایت تصمیم‌های بهتری بگیرند.

ترکیب پیاده‌سازی، تصویرسازی و توضیح مفهومی باعث شده این پروژه هم از نظر آموزشی ارزشمند باشد و هم پایه‌ای مناسب برای توسعه‌های بعدی در زمینه الگوریتم‌های بهینه‌سازی فراهم کند.

خروجی پروژه:

SA



PSO