بسمه تعالی

گزارش تکلیف دوم مبانی هوش محاسباتی

محمدباقر عابدی‌سقّا – ۹۵۳۱۹۰۴

آذر ۱۳۹۸

بازنمایی مسئله:

در این مسئله، تمام نقاط ورودی در لیستی به نام Pointsذخیره می‌شوند. پس از آن لیستی از کروموزوم‌ها طبق قالب تعریف پروژه ساخته می‌شود. اندازه لیست به اندازه طول ورودی است. در هر کروموزوم ۲ ژن وجود دارد که ضرایب خط z را معلوم می‌کنند. هر z به کمک نقطه متناظر آن در آرایه points محاسبه می‌شود. هر کروموزم متغیری برای ذخیره احراف معیار خود دارد (score).

Chromosome.py

این کلاس برای نگهداری اطلاعات هر کروموزوم، شامل ضرایب و انحراف معیار است. تابع evaluate برای محاسبه شایستگی(انحراف معیار) و تابع normalize برای نرمال سازی نقاط پس از هر جهش استفاده می‌شوند.

Plot.py

در این کلاس نمودارهای نهایی رسم خواهند شد. نقاط + خاکستری نقاط ورودی، نقاط حلقه بنفش، داده‌پاسخ مسئله و خط قرمز رنگ، خط متناظر با بهترین نقطه در مسئله است.

File_handler.py

در ابتدای اجرا برنامه نقاط ورودی را به صورت لیستی از جفت ضرایب برمی‌گرداند. [x,y]

Es.py

الگوریتم تکاملی در این کلاس اجرا می‌شود. مراحل آن را در زیر بررسی می‌کنیم.

تولید جمعیت اولیه: به تعداد نقاط ورودی به مسئله، کروموزوم ساخته و آن‌ها را در لیست pop ذخیره می‌کنیم.

مراحل پایین در حلقه‌ای و به تعداد نسل انجام می‌شوند.

انتخاب والدین: والدین جدیدی به صورت تصادفی به شکل جفت و به تعداد لامبدا انتخاب می‌شوند. نتیجه آن لیستی از جفت والدین خواهد شد.

کراس اور: والدین انتخاب شده با احتمال crossover_probability که از پارامتر های مسئله است، با هم کراس اور می‌شوند. به این صورت که به شکل تصادفی ضرایب اول یا دوم والد اول به عنوان ضریب اول فرزند و ضریب دوم هم به صورت تصادفی از واالد دوم انتخاب می‌شود. اگر احتمال کراس اور در محدوده crossover_probaility وجو نداشت، کراس اور صورت نمی‌گیرد. نتیجه این کار، لیستی از فرزندان جدید است.

جهش: جمعیت این نسل و فرزندان تولید شده با استفاده از تابع نرمال گوسی جهش می‌یابند. به این صورت که ضرایب هر کدام با مقدار noise که خروجی تابع نرمال است جمع می‌شود. پارامتر های sigma, mutation_rate قابل کنترل است و احتمال جهش و قدم جهش را تغییر می‌دهد.

شایستگی: شایستگی جمعیت جهش یافته که شامل جمعیت کنونی و فرزندان است، محاسبه می‌شود. شایستگی رابطه مستقیم با انجراف معیار هر کروموزوم از داده‌های ورودی دارد.

انتخاب: روش انتخاب شده Mu + Lambda خواهد بود. به این صورت که هر دو گروه جمعیت کنونی و فرزندان در یک لیست بر حسب شایستگی سورت شده و سپس، به اندازه نصف Mu از بالای لیست و نیمی دیگر از پایین لیست برگردانده می‌شود. لیست جدید همان جمعیت نسل بعد است.

نکات:

- در هر نسل، شایستگی بهترین کروموزوم، بدترین آن و میانگین شایستگی چاپ می‌شود.
- شرط خاتمه الگوریتم تکاملی پایان یافتن تعداد نسل خواهد بود. در نسل‌های آخر شاهد همگرایی شایستگی میانگین هستیم.

تاثیر پارامترها:

پارامترهای مسئله به صورت زیر هستند:

```
number_of_generationsتعداد نسل
Lambda_coefficient ضریب لامبدا که در میو ضرب شده و لامبدا را تعیین می‌کند
crossover_probabilityاحتمال ترکیب
mutation_rateاحتمال جهش
sigma بازه جهش

file_numشماره فایل دیتاست
min_val کمترین مقدار ضریب
max_val بیشترین مقدار ضریب
```

- به دلیل حجم محاسبات، تعداد نسل ۱۰۰ در نظر گرفته شده است.
- اگر احتمال کراس اور را افزایش دهیم مسئله دیرتر همگرا می‌شود و اگر احتمال آن را پایین بیاوریم تغییر نسل‌ها نسبت به هم کم شده، یا هرگز به جواب نمی‌رسید و یا ممکن است باعث همگرایی زودرس شود. (مینیمم محلی گیر کند)
- اگر تعداد فرزندان هر نسل کمتر شود حرکتی صورت نگرفته و ممکن است به جواب نرسیم. (جستجوی خوبی نداریم.
- اگر احتمال جهش بالا باشد حرکت بیشتری صورت می‌گیرد و فضای بیشتری جستجو می‌شود.
- اگر قدم های جهش زیاد شوند. حرکت مشابه رندوم واک می‌شود و اگر کم باشد، بازه جستجو کم می‌شود.

←جستجوی بهینه با مقدار متعادلی از ضرایب بالا انجام می‌گردد.

ضرایب در نظر گرفته شده برای دیتاست‌ها به شکل زیر است:

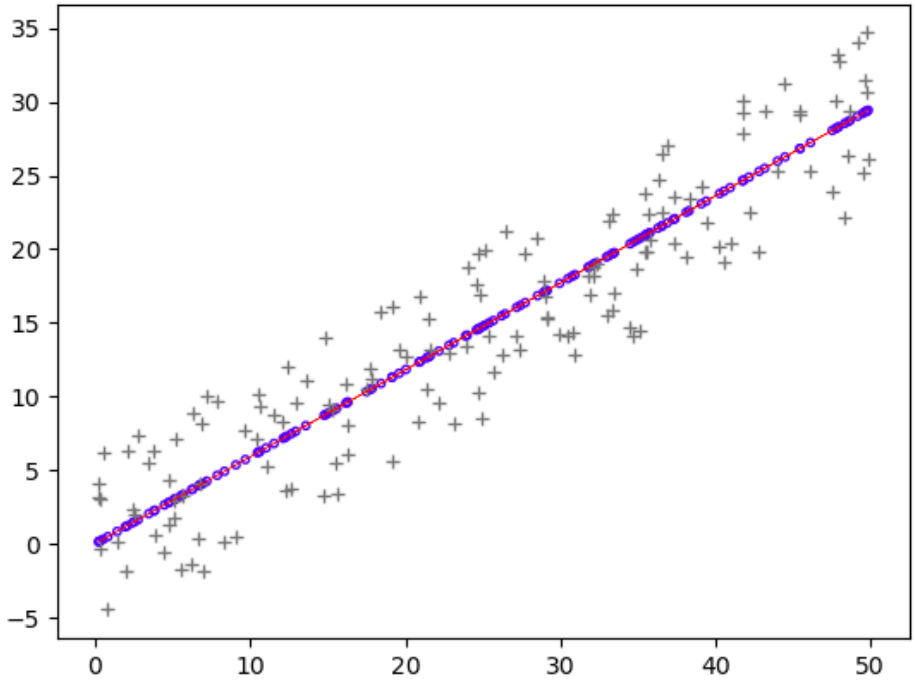
```
# problem parameters
number_of_generations = 100
Lambda_coefficient = .8
crossover_probability = .5
mutation_rate = .7
sigma = 5

file_num = '1'
min_val = 1
max_val = 10
```

نتایج آزمایش در صفحه بعد رسم شده است

Dataset1

best score: 17.2685 | worst score: 3.5858 | average: 14.4653



Dataset 2

best score: 29.9801| worst score: 5.2646| average 24.7450