به نام خدا

دانشکده مهندسی کامپیوتر

هوش مصنوعی و سیستم های خبره

پروژه سوم (برنامه نویسی ژنتیک)

دکتر آرش عبدی

پاییز ۱۴۰۲

امیرحسین سلیمی

شماره دانشجویی : 400521432

نحوه دقیق نگاشت مسئله بر درخت و محدودیت های احتمالی :

1. **ساختار درخت**
   - هر درخت در جمعیت بیانگر یک عبارت ریاضی است.
   - گره های درخت می توانند عملگر یا عملوند باشند.

2. **Initialazation**
   - جمعیت اولیه درختان به صورت تصادفی ایجاد می شود. عمق درختان توسط پارامتر max_depth کنترل می شود که حداکثر تعداد سطوح را در یک درخت تعیین می کند.

3. **ارزیابی fitness**
   - تناسب یک درخت با مقایسه پیش‌بینی‌های آن در برابر مقادیر هدف با استفاده از میانگین مربعات خطا ارزیابی می‌شود.
   - هدف تکامل درخت‌هایی است که عباراتی را تولید می‌کنند که دقیقاً با رابطه اساسی در داده‌ها مطابقت دارند.

4. **Selection**
   - درختان بر اساس امتیاز تناسب برای تولید مثل انتخاب می شوند. هرچقد فیت تر احتمال انتخاب بیشتر می شود.

5. **Crossover**
   - شامل تبادل زیردرخت بین دو درخت والد برای ایجاد دو فرزند است. این امکان تبادل بلوک های سازنده عبارات مختلف را فراهم می کند.

6. **Mutation**
   - جهش تغییرات تصادفی را در یک درخت ایجاد می کند، مانند جایگزینی یک زیردرخت با یک زیردرخت جدید که به طور تصادفی تولید شده است.

7. **Tree measurement**
   - متد Tree.measure عبارت ریاضی ارائه شده توسط یک درخت را با مجموعه ای از مقادیر ورودی ارزیابی می کند.

8. **Termination critera**
   - این الگوریتم پس از تعداد مشخصی از نسل ها (پارامتر نسل ها) یا زمانی که درختی با تناسب صفر (تناسب کامل) پیدا شد، پایان می یابد.

کلاس Learn :

این کلاس با هدف تکامل عبارات ریاضی که می توانند یک تابع هدف را بر اساس داده های ورودی تقریب بزنند طراحی شده.

ویژگی ها:

- **dimensions**: تعداد بعدها در ورودی
- **max_depth**: حداکثر عمق مجاز برای درخت های عبارت تکامل یافته
- **population_size**: تعداد افراد (درختان) در جمعیت
- **generations**: تعداد نسل هایی که الگوریتم برای آنها جمعیت را تکامل می دهد.
- **mutation_rate**: احتمال جهش در طول فرآیند تکامل.

متدها:

Init_pop : با ایجاد مجموعه ای از درختان تصادفی با استفاده از متد Tree.make_random_tree، جمعیت را راه اندازی می کند.

Measure_fitness : تناسب هر درخت در جمعیت را با مقایسه پیش‌بینی‌های آن در برابر مقادیر هدف ارزیابی می‌کند. تناسب با استفاده از میانگین مربعات خطا اندازه گیری می شود.

Selection : افراد (درختان) را از جمعیت بر اساس نمرات تناسب آنها برای تولید مثل انتخاب می کند.

Crossover : انجام crossover (recombination) بین دو درخت والد برای ایجاد دو درخت فرزند. crossover شامل تبادل subtree ها است.

Mutation : جهش های تصادفی را به درخت معرفی می کند. یک گره تصادفی در درخت انتخاب می کند و آن را با یک زیردرخت جدید که به طور تصادفی تولید می شود جایگزین می کند.

Fit : حلقه آموزشی اصلی الگوریتم برنامه ریزی ژنتیک. این به طور مکرر جمعیت را از طریق انتخاب، crossover و جهش تکامل می دهد. همچنین تکامل بهترین پیش‌بینی‌های درخت را ترسیم می‌کند.

توضیحات پارت a بخش c, d , e, f, i

متد init_pop از Tree.make_random_tree برای ایجاد درخت های تصادفی با حداکثر عمق مشخص شده استفاده می کند.

والدین بر اساس بالاترین تناسب انتخاب می شوند.

Crossover به این صورت است که به صورت رندوم 2 زیر درخت انتخاب و باهم جا به جا شده و 2 بچه نسل بعد تولید میشود.

Mutation به این صورت است که اگر مقدار mse متوسط فعلی از قبلی بدتر باشد مقدار mutation rate به مقذار خودش (تا سقف 0.5) افزایش یافته و سپس در آخر هر مرحله بهترین درخت از لحاظ تناسب انتخاب شده.(در صورتی که تناسب برابر 0 شود برنامه زودتر خاتمه پیدا میکند)

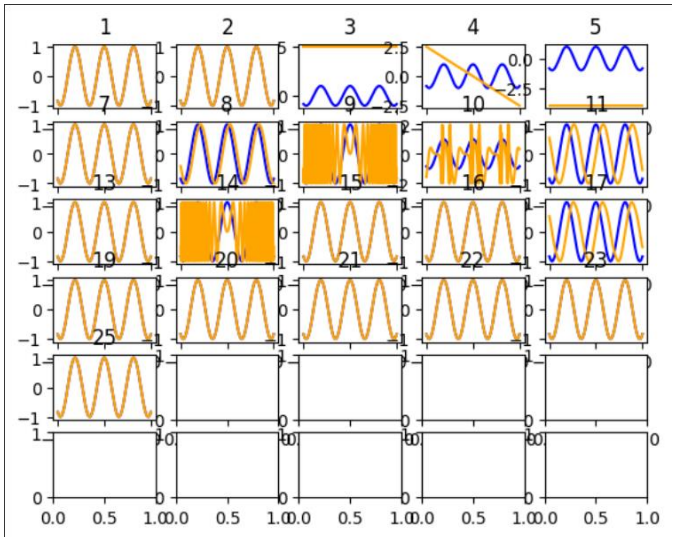
نتایج آزمایش های انجام شده :

Cos(x):

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 24.97115214963063
generation = 8 =>, best fitness : 24.97115214963063
generation = 9 =>, best fitness : 24.97115214963063
generation = 10 =>, best fitness : 24.97115214963063
generation = 11 =>, best fitness : 24.97115214963063
generation = 13 =>, best fitness : 1.2133376469638264
generation = 14 =>, best fitness : 1.2133376469638264
generation = 15 =>, best fitness : 1.2133376469638264
generation = 16 =>, best fitness : 1.2133376469638264
generation = 17 =>, best fitness : 1.2133376469638264
generation = 19 =>, best fitness : 0.8942226720679576
generation = 20 =>, best fitness : 0.8942226720679576
generation = 21 =>, best fitness : 0.8942226720679576
generation = 22 =>, best fitness : 0.8942226720679576
generation = 23 =>, best fitness : 0.8942226720679576
generation = 25 =>, best fitness : 0.5230018023893986
generation = 26 =>, best fitness : 0.5230018023893986
generation = 27 =>, best fitness : 0.5230018023893986
generation = 28 =>, best fitness : 0.5230018023893986
generation = 29 =>, best fitness : 0.5230018023893986
generation = 31 =>, best fitness : 0.0
```

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 33.92287528478778
generation = 8 =>, best fitness : 33.92287528478778
generation = 9 =>, best fitness : 33.92287528478778
generation = 10 =>, best fitness : 33.92287528478778
generation = 11 =>, best fitness : 33.92287528478778
generation = 13 =>, best fitness : 1.0292493175058348
generation = 14 =>, best fitness : 1.0292493175058348
generation = 15 =>, best fitness : 1.0292493175058348
generation = 16 =>, best fitness : 1.0292493175058348
generation = 17 =>, best fitness : 1.0292493175058348
generation = 19 =>, best fitness : 0.9930607109872042
generation = 20 =>, best fitness : 0.9930607109872042
generation = 21 =>, best fitness : 0.9930607109872042
generation = 22 =>, best fitness : 0.9930607109872042
generation = 23 =>, best fitness : 0.9930607109872042
generation = 25 =>, best fitness : 0.0
```

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 33.92306853578947
generation = 8 =>, best fitness : 33.92306853578947
generation = 9 =>, best fitness : 33.92306853578947
generation = 10 =>, best fitness : 33.92306853578947
generation = 11 =>, best fitness : 33.92306853578947
generation = 13 =>, best fitness : 1.256623908659075
generation = 14 =>, best fitness : 1.256623908659075
generation = 15 =>, best fitness : 1.256623908659075
generation = 16 =>, best fitness : 1.256623908659075
generation = 17 =>, best fitness : 1.256623908659075
generation = 19 =>, best fitness : 0.9999999999999999
generation = 20 =>, best fitness : 0.9999999999999999
generation = 21 =>, best fitness : 0.9999999999999999
generation = 22 =>, best fitness : 0.9999999999999999
generation = 23 =>, best fitness : 0.9999999999999999
generation = 25 =>, best fitness : 0.0
```

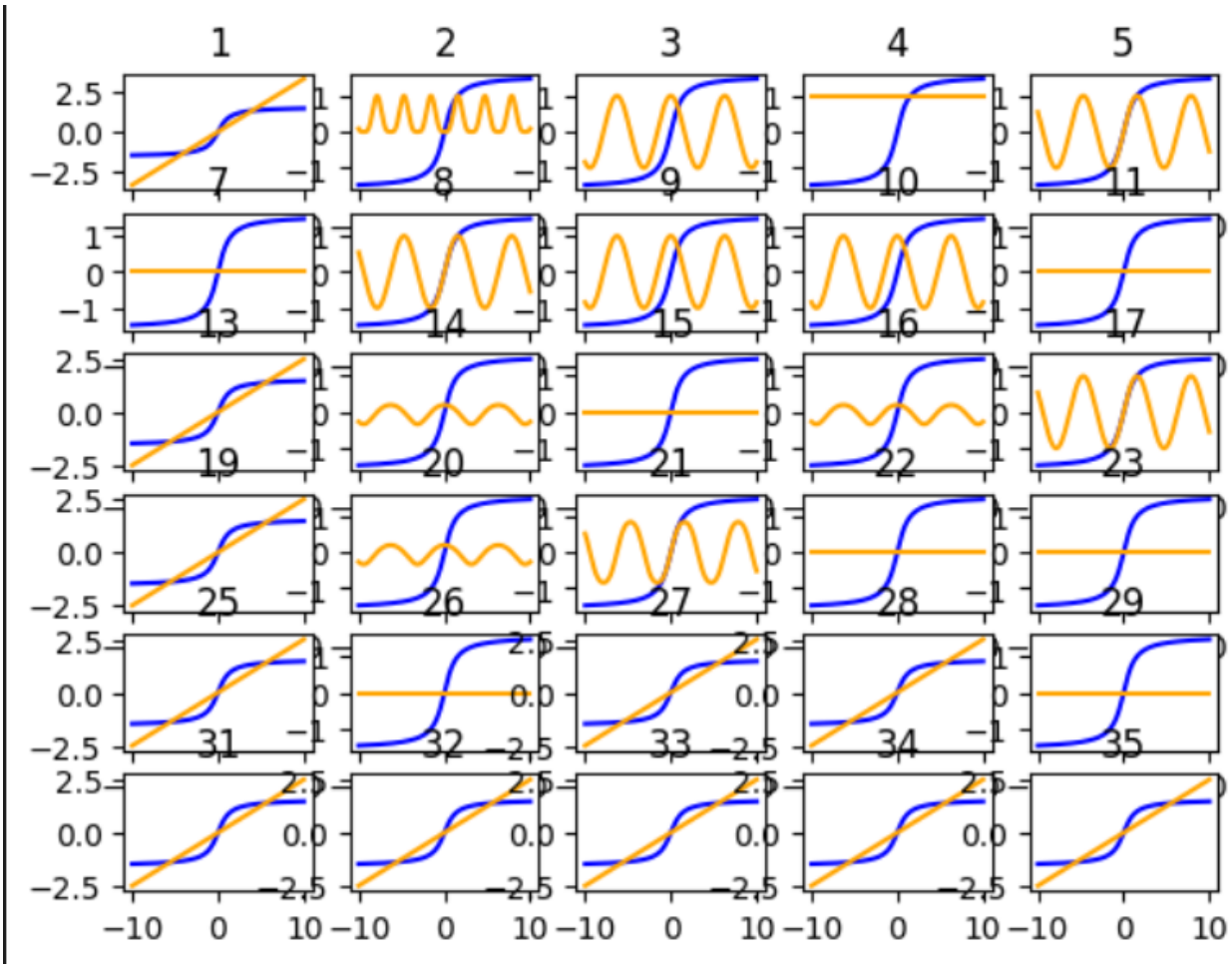


Arctan(x)

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 21.163751860523334
generation = 8 =>, best fitness : 21.163751860523334
generation = 9 =>, best fitness : 21.163751860523334
generation = 10 =>, best fitness : 21.163751860523334
generation = 11 =>, best fitness : 21.163751860523334
generation = 13 =>, best fitness : 2.160651082017888
generation = 14 =>, best fitness : 2.160651082017888
generation = 15 =>, best fitness : 2.160651082017888
generation = 16 =>, best fitness : 2.160651082017888
generation = 17 =>, best fitness : 2.160651082017888
generation = 19 =>, best fitness : 1.7548940224106142
generation = 20 =>, best fitness : 1.7548940224106142
generation = 21 =>, best fitness : 1.7548940224106142
generation = 22 =>, best fitness : 1.7548940224106142
generation = 23 =>, best fitness : 1.7548940224106142
generation = 25 =>, best fitness : 1.7548940224106142
generation = 26 =>, best fitness : 1.7548940224106142
generation = 27 =>, best fitness : 1.7548940224106142
generation = 28 =>, best fitness : 1.7548940224106142
generation = 29 =>, best fitness : 1.7548940224106142
...
generation = 33 =>, best fitness : 1.6541796696362305
generation = 34 =>, best fitness : 1.6541796696362305
generation = 35 =>, best fitness : 1.6541796696362305
```

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 21.163751860523334
generation = 8 =>, best fitness : 21.163751860523334
generation = 9 =>, best fitness : 21.163751860523334
generation = 10 =>, best fitness : 21.163751860523334
generation = 11 =>, best fitness : 21.163751860523334
generation = 13 =>, best fitness : 2.160651082017888
generation = 14 =>, best fitness : 2.160651082017888
generation = 15 =>, best fitness : 2.160651082017888
generation = 16 =>, best fitness : 2.160651082017888
generation = 17 =>, best fitness : 2.160651082017888
generation = 19 =>, best fitness : 1.7548940224106142
generation = 20 =>, best fitness : 1.7548940224106142
generation = 21 =>, best fitness : 1.7548940224106142
generation = 22 =>, best fitness : 1.7548940224106142
generation = 23 =>, best fitness : 1.7548940224106142
generation = 25 =>, best fitness : 0.7241130880600881
generation = 26 =>, best fitness : 0.7241130880600881
generation = 27 =>, best fitness : 0.7241130880600881
generation = 28 =>, best fitness : 0.7241130880600881
generation = 29 =>, best fitness : 0.7241130880600881
...
generation = 33 =>, best fitness : 0.7241130880600881
generation = 34 =>, best fitness : 0.7241130880600881
generation = 35 =>, best fitness : 0.7241130880600881
```

```
generation = 1 =>, best fitness : inf
generation = 2 =>, best fitness : inf
generation = 3 =>, best fitness : inf
generation = 4 =>, best fitness : inf
generation = 5 =>, best fitness : inf
generation = 7 =>, best fitness : 21.163751860523334
generation = 8 =>, best fitness : 21.163751860523334
generation = 9 =>, best fitness : 21.163751860523334
generation = 10 =>, best fitness : 21.163751860523334
generation = 11 =>, best fitness : 21.163751860523334
generation = 13 =>, best fitness : 2.160651082017888
generation = 14 =>, best fitness : 2.160651082017888
generation = 15 =>, best fitness : 2.160651082017888
generation = 16 =>, best fitness : 2.160651082017888
generation = 17 =>, best fitness : 2.160651082017888
generation = 19 =>, best fitness : 2.1328672456468603
generation = 20 =>, best fitness : 2.1328672456468603
generation = 21 =>, best fitness : 2.1328672456468603
generation = 22 =>, best fitness : 2.1328672456468603
generation = 23 =>, best fitness : 2.1328672456468603
generation = 25 =>, best fitness : 1.7548940224106142
generation = 26 =>, best fitness : 1.7548940224106142
generation = 27 =>, best fitness : 1.7548940224106142
generation = 28 =>, best fitness : 1.7548940224106142
generation = 29 =>, best fitness : 1.7548940224106142
...
generation = 33 =>, best fitness : 1.6772143483438218
generation = 34 =>, best fitness : 1.6772143483438218
generation = 35 =>, best fitness : 1.6772143483438218
```



همانطور که در نتایج میبینم توابع ساده تر و داری ضوابط کمتر بهتر تناسب پیدا میکنند.

جمع بندی ژنتیک :

ژنتیک به خوبی در حل مسائل پیچیده و چندبعدی عمل می‌کنند. این الگوریتم‌ها معمولاً می‌توانند در جستجوی فضاهای جستجوی پیچیده به بهینه‌سازی بپردازند.

الگوریتم ژنتیک می‌تواند با مسائلی که تابع هدف آن‌ها غیرخطی یا غیردسته‌ای است، به خوبی کار کند. این الگوریتم می‌تواند به صورت همزمان در جهات مختلف جستجو کند.

الگوریتم‌های ژنتیک به دلیل ویژگی‌های انعطاف‌پذیری و تطبیق آن‌ها با مسائل مختلف، می‌توانند برای حل انواع مسائل بهینه‌سازی مورد استفاده قرار گیرند. ویژگی‌هایی چون توانایی تطبیق جمعیت با محیط، تولید تنوع جمعیت از طریق عملیات متنوع ژنتیک، و امکان بهبود جواب‌ها از طریق جهش، این الگوریتم‌ها را به یک ابزار مؤثر در بهینه‌سازی تبدیل کرده است.