تقریب توابع ریاضی

به کمک الگوریتم ژنتیک

# پروژه درس هوش مصنوعی

گردآورنده: میرنیما قاسمیان

استاد درس: دکتر آرش عبدی

**هدف پروژه**: ارزیابی میزان تسلط به مفاهیم پایه الگوریتم های تکاملی (در اینجا برنامه نویسی ژنتیک)

**پیاده سازی:**

ابتدا تعدادی ورودی را در نظر گرفته و خروجی آنها را با جایگذاری در تابع اصلی به دست می آوریم. سپس

عملگر هایی که در این پیاده سازی مورد استفاده قرار می گیرند را معرفی میکنیم مانند :

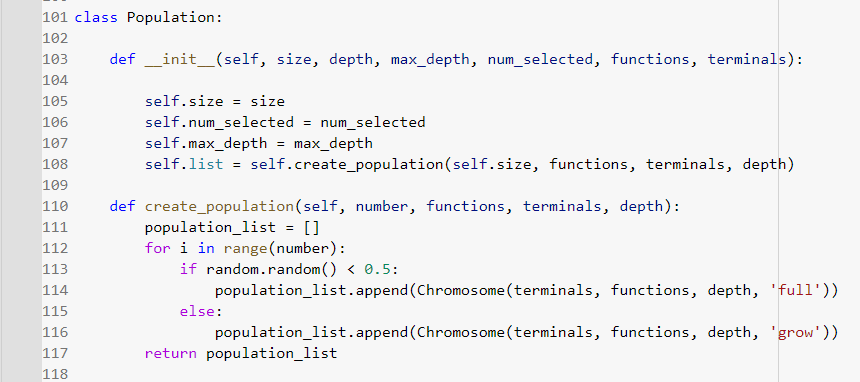
عملگر های دو عملوندی: “^”,“-“,”+”,”\*”,”/”عملگرهای تک عملوندی: "cos"،"sin"

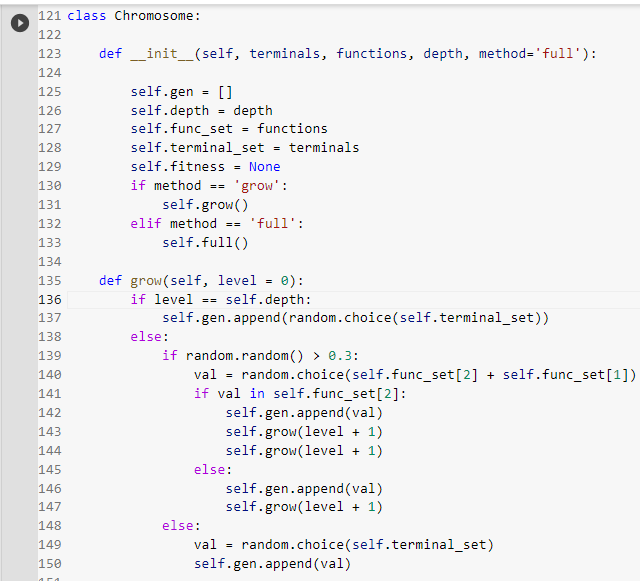
به صورت کلی سعی داریم با آموزش تعدادی از نقاط، تابع اولیه را مجدد به دست آوریم و پیش بینی کنیم ضابطه ی اولیه چه بوده است!

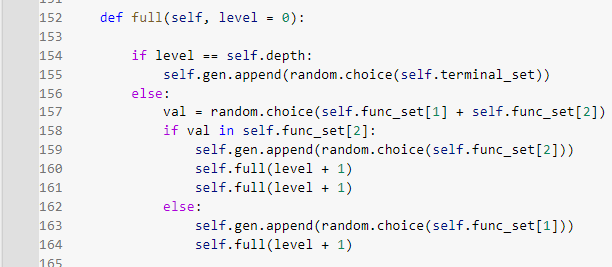
• به عنوان نخستین گام باید جمعیت اولیه(population) را تولید کنیم.

جمعیت تولید شده شامل n تا کروموزوم میباشد که تعداد n به عنوان ورودی Population Constructor داده می شود.

هر کروموزوم تخمینی از حالت اولیه مسئله می باشد(در واقع هرکدام یک درخت اولیه برای پیشبرد مسئله میباشد) که در ادامه به هر کروموزوم یک مقدار شایستگی تخصیص می دهیم (fitness)

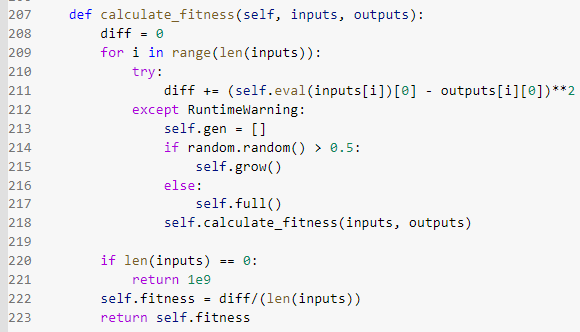




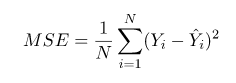


هر کدام از کروموزوم ها نمایی از genotype بوده که پیمایش preorder traversal از درخت آزمایشی میباشد! بعد از ساخت جمعیت اولیه بخش اصلی الگوریتم در کلاس Algorithm پیاده سازی شده است که مسئول آموزش نقاط(train)، ترکیب کروموزوم ها(crossover)، عمل جهش(Mutate) و ساخت نسل جدید بر اساس شایستگی می باشد.

ابتدا توسط تابع calculate\_fitness شایستگی هر کدام از کروموزوم ها را تعیین کرده و عددی به آن نسبت می دهیم.

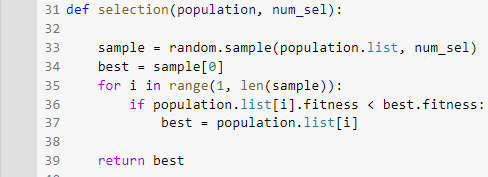


* برای تعیین مقدار fitness همانگونه که در تصویر بالا مشخص است از الگوریتم Mean Squared Error (MSE) استفاده می کنیم که همان میانگین مربعات تفاضل مقدار تخمین زده شده از مقدار اصلی میباشد.

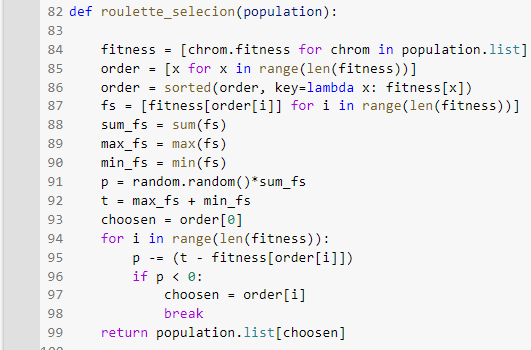


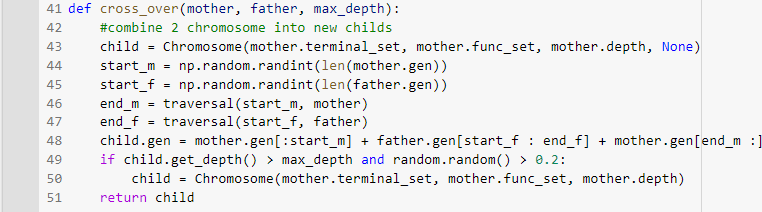
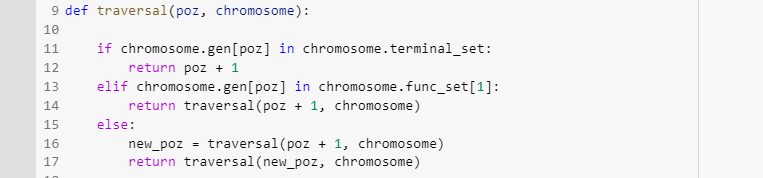
با توجه به این تعریف روند الگوریتم باید بگونه ای باشد که به مرور زمان به مقدار fitness کمتری برای کروموزوم های تخمین زده شده برسیم.

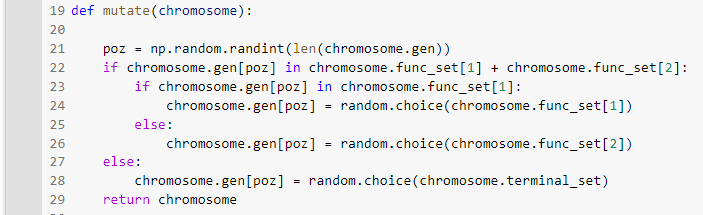
روند آموزش نقاط به تعداد متغیر iterations انجام می شود که در ابتدای برنامه مقدار آن را مشخص میکنیم. در هر iterate با استفاده از تابع selection دو مورد از کروموزوم ها با بهترین شایستگی را به صورت تصادفی انتخاب میکنیم.



* برای دقت بیشتر الگوریتم در این مرحله بهتر است از تابع roulette\_selection استفاده کنیم تا کروموزوم های با شایستگی بیشتر و متنوع تری به دست آورده و در نتیجه عملیات crossover موفق تری داشته باشیم!!





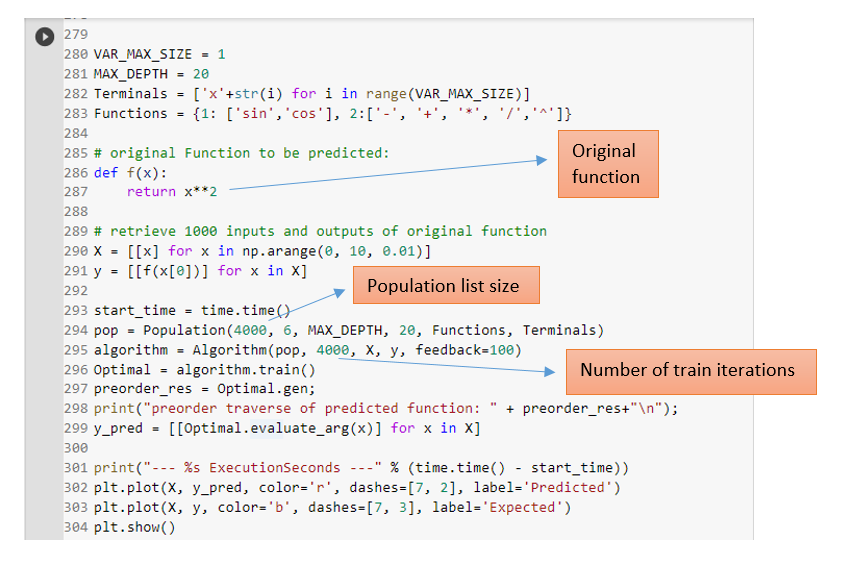


کاملا به صورت تصادفی بخشی از کروموزوم های پدر و مادر را انتخاب کرده و آن را با یکدیگر جایگزین میکنیم تا نسل جدید ایجاد شود.

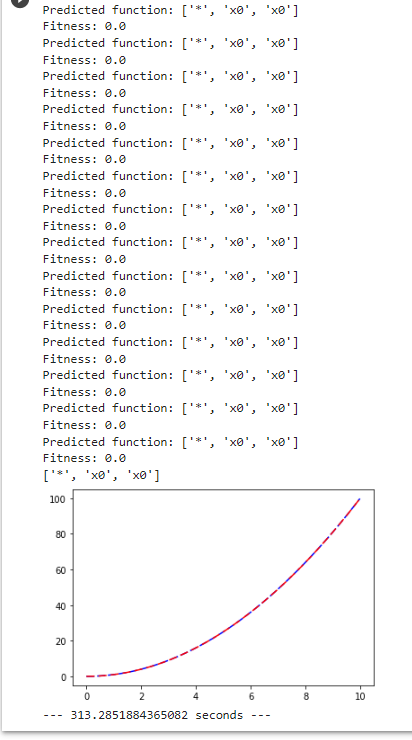
در تابع mutate یکی از ایندکس های کروموزوم را تغییر می دهیم.

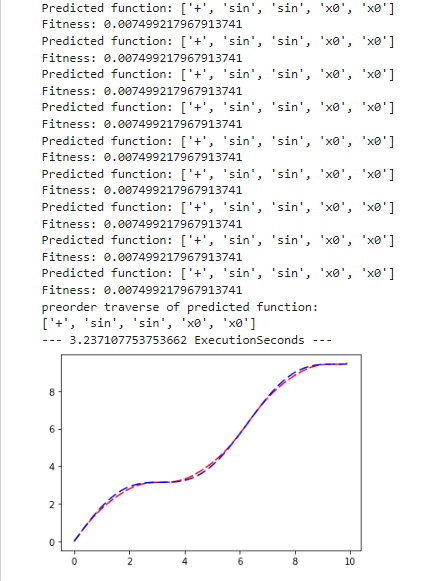
* در گام آخر بعد از تعداد train هایی که اول برنامه مشخص کرده بودیم، باید بهترین کروموزوم به دست آمده (از نظر تابع شایستگی) را نمایش دهیم.

در نمودار نهایی خط قرمز تابع پیش بینی شده و خط آبی تابع اولیه می باشد.

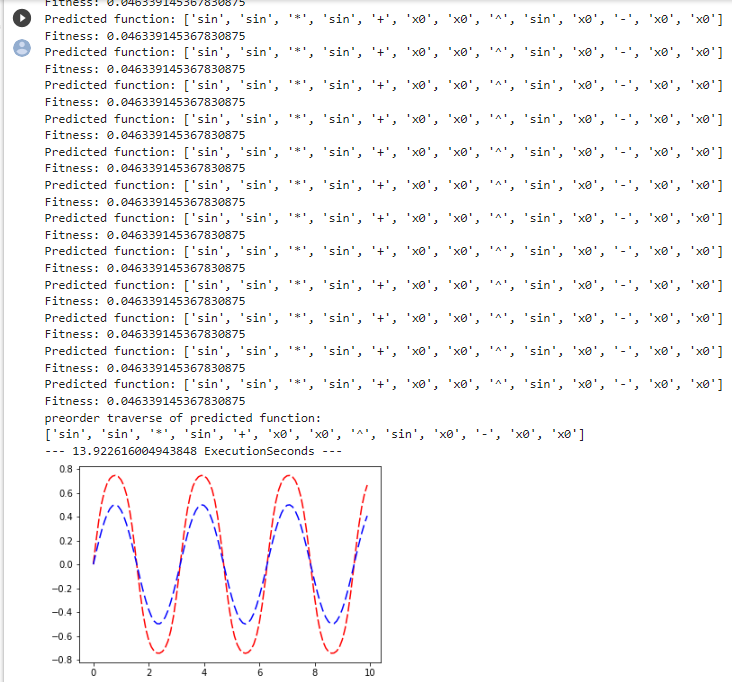
**خروجی**: به عنوان خروجی مقدار شایستگی هر کروموزوم به همراه پیمایش pre-order درخت آن آورده شده و روی نمودار نیز تابع پیش بینی شده با رنگ قرمز نمایش داده شده است. رنگ آبی تابع اولیه(اصلی) می باشد. در آخرین خط نیز زمان اجرای الگوریتم بر حسب ثانیه آورده شده.

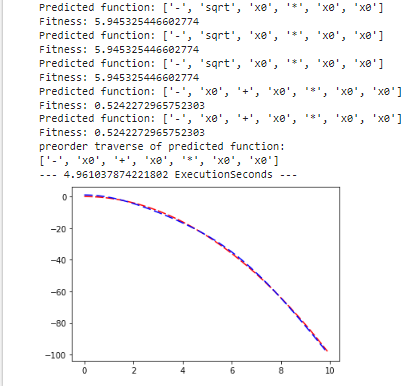
در تصاویر زیر چند مثال مختلف بررسی شده و تابع اولیه هر تصویر کنار آن ذکر شده است:





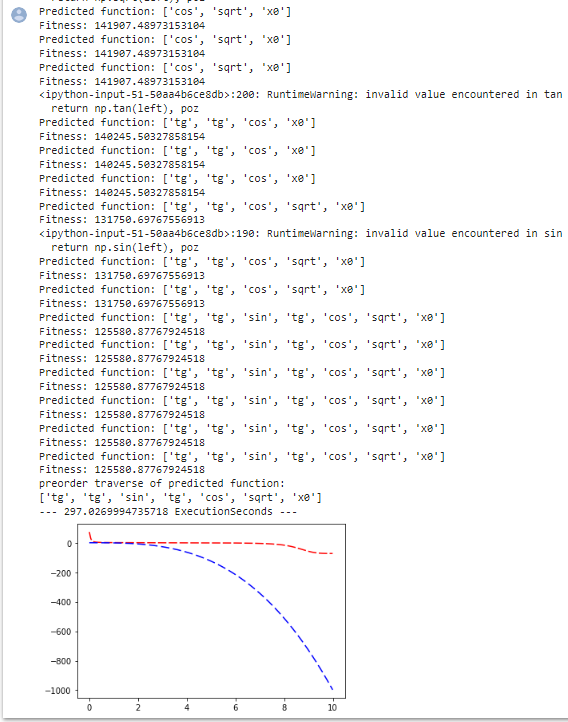


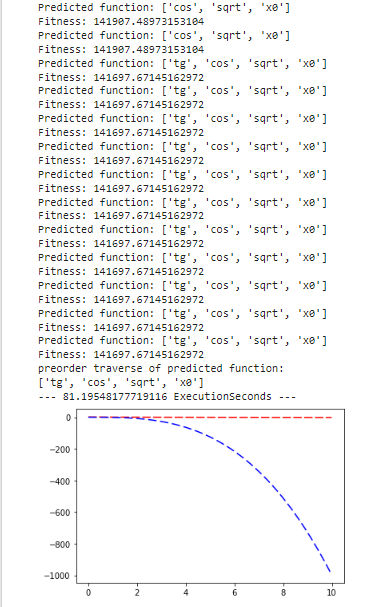






در مثال های قبل فقط عملگر های ابتدایی برای تخمین استفاده شده بود، در تصویر بعدی ln, tan, abs,sqrt نیز اضافه شده اند.





نکته قابل توجه در اشکال خروجی اینست که توابع پیش بینی شده کاملا وابسته به تعداد مجموعه ی جمعیت اولیه و تعداد دفعات train می باشد و در صورت افزایش این مقادیر توابع رفته رفته به واقعیت نزدیک تر میشوند و شایستگی بهتری پیدا می کنند.

به عنوان مثال در شکل سمت راست تعداد train های نقاط اولیه 2 برابر شده و اعضای جمعیت اولیه نیز بیشتر شدند تا در نتیجه به جواب بهتری با fitness کمتر برسیم.

* جمع بندی:

بنابراین به کمک الگوریتم ژنتیک در کنار عملیاتی مانند (MSE, roulette selection,…) توانستیم با داشتن اطلاعاتی از تابع اولیه، تابعی نظیر آن را تخمین بزنیم و ارزش گذاری کنیم.