浙大城市学院

ZHEJIANG UNIVERSITY CITY COLLEGE

遗传算法

——TSP及符号回归的python实现

课程名称：人工智能导论

学　　号：31801341

姓　　名：童峻涛

专业班级：软件工程1802

所在学院：计算机与计算科学院

报告日期：2021年01月02日

一、选题背景简介

1. 算法简介

“进化”（Evolution），是人类历史上伟大的发现之一。适者生存，不适者淘汰，达尔文的进化理论让我们见识到自己是怎么来的。而当计算机的程序也能进化，淘汰劣质的程序，从而得到最优化的解，是多么的神奇。

在这里，遗传算法(genetic algorithm, GA)就是进化算法的一个分支，其基本原理是仿效生物界中的“物竞天择，适者生存”的演化法则，将达尔文的进化理论搬进了计算机。

遗传算法是把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及变异等运算来交换种群中染色体的信息，最终生成符合优化目标的染色体。

2. 名词解释

在遗传算法中，染色体对应的是数据或数组，通常是由一维的串结构数据来表示，串上各个位置对应基因的取值。

基因组成的串就是染色体(chromosome)，或者称为基因型个体(individual)。

一定数量的个体组成了群体(population)。

群体中的个体数目称为群体大小(population size)，也成为群体规模。

而各个个体对环境的适应程度叫适应度(fitness)。

3. 应用领域

因为在求解较为复杂的组合优化问题时，相对一些常规的优化算法，通常能够较快地获得较好的优化结果。遗传算法已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。

二、遗传算法基本步骤

1. 一般步骤

（1）编码

GA 在进行搜索之前先将解空间的解数据表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合便构成了不同的点。

（2）初始群体的生成

随机产生 N 个初始串结构数据，每个串结构数据称为一个个体，N 个个体构成了一个群体。GA 以这 N 个串结构数据作为初始点开始进化。

（3）适应度评估

适应度表明个体或解的优劣性。不同的问题，适应度函数的定义方式也不同。

（4）选择

选择的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应度强的个体为下一代贡献一个或多个后代的概率大。选择体现了达尔文的适者生存原则。

（5）交叉

交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。

（6）变异

变异首先在群体中随机选择一个个体，对于选中的个体以一定的概率随机地改变串结构数据中某个串的的值。同生物界一样，GA 中变异发生的概率很低，通常取值很小。

2. 特点

（1） 遗传算法从问题解的串集开始搜索，而不是从单个解开始，这是与传统优化算法的极大区别。传统优化算法从单个初始值迭代求最优解，容易误入局部最优解。遗传算法从串集开始搜索，覆盖面大，利于全局择优。

（2） 遗传算法同时处理群体中的多个个体，即对搜索空间中的多个解进行评估，减少了陷入局部最优解的风险，同时算法本身易于实现并行化。

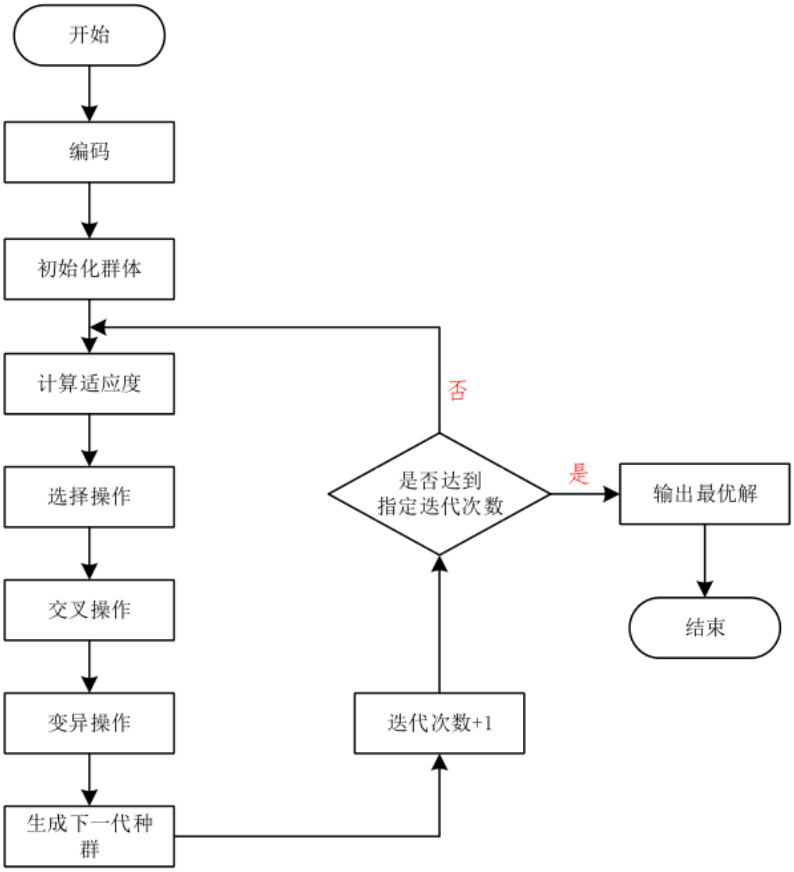
（3） 遗传算法基本上不用搜索空间的知识或其它辅助信息，而仅用适应度函数值来评估个体，在此基础上进行遗传操作。适应度函数不仅不受连续可微的约束，而且其定义域可以任意设定，使得遗传算法的应用范围大大扩展。

（4） 遗传算法不是采用确定性规则，而是采用概率的变迁规则来指导他的搜索方向。

（5） 遗传算法具有自组织、自适应和自学习性，利用进化过程获得的信息自行组织搜索时，适应度大的个体具有较高的生存概率，并获得更适应环境的基因结构。

（6） 算法本身也可以采用动态自适应技术，在进化过程中自动调整算法控制参数和编码精度，比如使用模糊自适应法。

3. 算法流程图



三、所选问题简介

1. 旅行商问题概述

旅行商问题(Traveling Saleman Problem，TSP)是指单一的旅行商需要到 n 个城市去推销商品，要求从某一城市出发，经过 n-1 个城市，旅行商在途径的 n-1 个城市能且仅能经过一次，再回到出发城市，使得旅行商的路程最短。

遗传算法的思想为优胜劣汰，通过交叉操作、变异操作获得多样性的解，在下一代中保留目标函数最优的解，并通过轮盘赌方式选择下一代个体，不断循环迭代，从而不断优化问题的解。

在旅行社问题中采用的编码方式为染色体均为互不相同的一组整数排列，所以遗传算法的交叉操作、变异操作、选择操作可不发生变化，在变异采用了随机交换两个城市的顺序。

2. 基于遗传算法的符号回归问题概述

符号回归，是相对于“数值”回归(Numerical Regression)而言的，即统计中的

回归分析。回归分析通常要首先假设问题所服从的函数的形式及其参数，然后根据数据求得最符合的一组参数，从而最终确定函数。

符号回归是遗传编程最早的一类应用之一。符号回归的运算符集合(terminal set)主要由运算符（比如+、-、*、/、sin、cos、log…），随机数和变量组成。在这个问题中，演化的程序就是一串运算符。所希望最终得到的程序，，能够尽量符合给定的数据集。符号回归完全没有假定函数的形式，实际上符号回归包括遗传编程，甚至整个计算智能（Computational Intelligence）和演化计算（Evolutionary computing）领域的目标就是全自动地发现知识、模式和规律。

四、旅行商问题实现分析

1. 定义全局参数

```
# DNA size 城市的个数
N_CITIES = 20
# 交叉概率
CROSS_RATE = 0.1
# 变异概率
MUTATE_RATE = 0.02
# 样本库
POP_SIZE = 500
# 500 次迭代
N_GENERATIONS = 500
```

2. DNA 编码

我们可以尝试对每一个城市有一个 ID, 那经历的城市顺序就是按 ID 排序。比如说商人要经过 3 个城市, 我们就有以下六种方式：

（1）0-1-2

（2）0-2-1

（3）1-0-2

（4）1-2-0

（5）2-0-1

（6）2-1-0

这 6 种排列方式每一种排列方式我们就能把它作为一种 DNA 序列, python 中用 numpy 产生这种 DNA 序列的方式很简单。

```
import numpy as np
np.random.permutation(3)
# array([1, 2, 0])
```

3. 适应度函数的确定

在计算适应度的时候，我们只需要将几个城市连成线，计算总路径的长度即可。根据路径的长度判断，总路径越短越优秀。

使用如下的计算公式得到适应度：

```
1.  fitness = 1/total_distance
```

考虑到使适应度区分效果更加明显，使用 exp 函数将数值扩大，以至于在轮盘赌策略时，尽可能多停留在优质基因区间。

将公式改进得到：

```
1.  fitness = np.exp(self.DNA_size * 2 / total_distance)
```

4. 交叉和变异

对于基因的交叉，一般的交叉规则如下：

```
1.  p1=[0,1,2,3] #baba
2.  p2=[3,2,1,0] #mama
3.  cp=[m,b,m,b] #m:mama  b:baba
4.  c1=[3,1,1,3] #child
```

但是在旅行社问题中，这意味着商人从 3 号城市前往 1 号城市后继续停留在了 1 号城市，最终会到 3 号城市，另外的 0 号城市和 2 号城市并没有到达。

显然不行，所以对于交叉和变异在该问题中要换一种策略。

```
1.  p1=[0,1,2,3] #baba
2.  cp=[ ,b, ,b] #from baba
3.  c1=[1,3, , ] #to be continued
4.  p2=[3,2,1,0] #mama
5.  cp=[m, ,m, ] #from mama
6.  c1=[1,3,2,0] #child
```

此时先选择来自对应位置父序列的城市序号，排列在子序列的前面位置，后将未访问的城市序号按照其在母序列中的顺序填入子序列中得到完整的子序列。这样就能避免存在为访问城市的问题。

```
1.  # 交叉
2.      def crossover(self, parent, pop):
3.          # 概率交叉，先确定排列父亲的城市，找出与父亲不同的城市，然后按照顺序排列，
    组合得到新的序列
4.          if np.random.rand() < self.cross_rate:
5.              i_ = np.random.randint(0, self.pop_size, size=1)
6.              # 选择交叉的点
```

```
            cross_points = np.random.randint(0, 2, self.DNA_size).astype(np.bool)
            # 找到与父亲不同的城市
            keep_city = parent[~cross_points]
            # 得到交叉的城市
            swap_city = pop[i_, np.isin(pop[i_].ravel(), keep_city, invert=True)]
            # 保留原有的母本
            parent[:] = np.concatenate((keep_city, swap_city))
        return parent
```

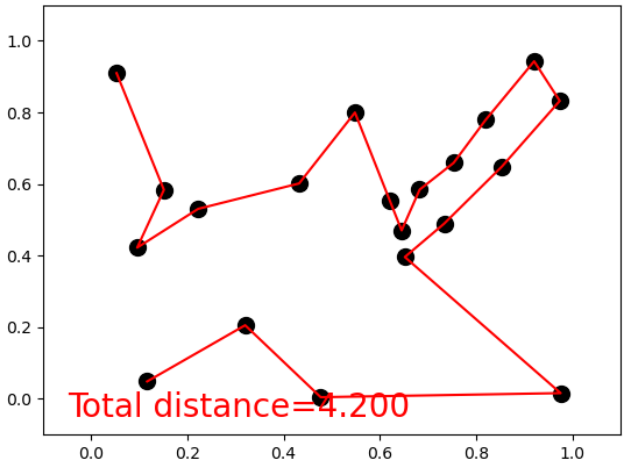
在变异阶段，采取的措施是随机将两个城市序号交换，以防止产生重复或者不存在的编号。

```
# 变异
    def mutate(self, child):
        # 概率交换，随机选择两个数位，进行交换
        for point in range(self.DNA_size):
            if np.random.rand() < self.mutate_rate:
                swap_point = np.random.randint(0, self.DNA_size)
                swapA, swapB = child[point], child[swap_point]
                child[point], child[swap_point] = swapB, swapA
        return child
```

5. 结果分析

程序最开始会生成城市的点，通过不断地交叉变异，选择不同的路径搭配，来搜索最短的距离。

每次均在 100-200 代之间趋于稳定，最终结果并一定是最优的，是短时间内大概最优的一个选择，可以通过调节代数以及变异率等进一步得到更优结果。



```
Gen: 101 | best fit: 10783.81
Gen: 102 | best fit: 11174.58
Gen: 103 | best fit: 11174.58
Gen: 104 | best fit: 11174.58
Gen: 105 | best fit: 13679.56
Gen: 106 | best fit: 11882.84
Gen: 107 | best fit: 11882.84
Gen: 108 | best fit: 11882.84
Gen: 109 | best fit: 11882.84
Gen: 110 | best fit: 11882.84
Gen: 111 | best fit: 11882.84
Gen: 112 | best fit: 13679.56
Gen: 113 | best fit: 13679.56
Gen: 114 | best fit: 13679.56
Gen: 115 | best fit: 13679.56
Gen: 116 | best fit: 13679.56
Gen: 117 | best fit: 13679.56
```

五、基于遗传算法的符号回归问题实现分析

1. 浅析 gplearn

gplearn 是目前 Python 内最成熟的符号回归算法实现，作为一种一种监督学习方法，符号回归（symbolic regression）试图发现某种隐藏的数学公式，以此利用特征变量预测目标变量。

符号回归的具体实现方式是遗传算法（genetic algorithm）。一开始，一群天真而未经历选择的公式会被随机生成。此后的每一代中，最“合适”的公式们将被选中。随着迭代次数的增长，它们不断繁殖、变异、进化，从而不断逼近数据分布的真相。

2. gplearn 中 SymbolicRegressor 的参数介绍

名称	说明
population_size	种群规模
metric	字符串，目标函数(损失函数)(默认值=‘MAE’(平均绝对误差))，此外还包括 gplearn 提供的 mse 等，也可以自定义。
generations	需要进化的代数
stopping_criteria	浮点数，停止条件
verbose = 1	为输出进度条记录
init_method	**1）grow 决策：**公式树从根节点开始生长。在每一个子节点，gplearn 会从所有常数、变量和函数中随机选取一个元素。如果它是常数或者变量，那么这个节点会停止生长，成为一个叶节点。如果它是函数，那么它的两个子节点将继续生长。用 grow 策略生长得到的公式树往往不对

	称，而且普遍会比用户设置的最大深度浅一些；在变量的数量远大于函数的数量时，这种情况更明显。 **2）full 决策：**除了最后一层外，其他所有层的所有节点都是内部节点——它们都只是随机选择的函数，而不是变量或者常数。最后一层的叶节点则是随机选择的变量和常数。用 full 策略得到的公式树必然是 perfect binary tree。 **3）half and half 决策：**一半的公式树用 grow 策略生成，另一半用 full 策略生成。
function_set	字符串, 用于符号回归的函数，包括 gplearn 原始提供以及自定义
parsimony_coefficient	简约系数，浮点数或 “auto”, 可选 (默认值=0.001)用于惩罚过于复杂的公式。 简约系数往往由实践验证决定。如果过于吝啬（简约系数太大），那么所有的公式树都会缩小到只剩一个变量或常数；如果过于慷慨（简约系数太小），公式树将严重膨胀。
random_state	为了保证程序每次运行都分割一样的训练集和测试集。否则，同样的算法模型在不同的训练集和测试集上的效果不一样。
n_jobs	整数，可选(默认值=1)用于设置并行计算的操作
max_samples	浮点数, 可选 (默认值=1.0)从样本中抽取的用于评估每个树(成员)的百分比

3. 利用 gplearn 进行特征工程

（1）简要描述

gplearn 这个库提供对于解决邻域的先验知识缺少问题的思路，随机化生成大量特征组合方式，解决了没有先验知识，手工生成特征费时费力的问题。

通过遗传算法进行特征组合的迭代，而且这种迭代是有监督的迭代，存留的特征和标签相关性是也来越高的，大量低相关特征组合会在迭代中被淘汰掉，用决策树算法做个类比的话，我们自己组合特征然后筛选，好比是后剪枝过程，遗传算法进行的则是预剪枝的方式。

根据领域的先验知识，对如金额特征、日期特征进行比值操作生成一些特征，这些特征经常能够提升验证集和测试集的分数，在模型中重要程度也很高。

（2）论文描述

论文选自 1994 年在《Statistics and Computing》中 John Koza 所著的《Genetic programming as a means for programming computers by natural selection》一文，其主要思想是通过基于遗传算法的符号回归来由计算机得到一个适合的计算公式，来对相关数据进行预测。

《Statistics and Computing》最新 CiteScore 值为 2.37（引文计数（2018）/文

献（2015——2017)），其中 Computer Science 大类排名第 22 位。

（3）术语描述

GNP：国民生产总值 = 国内生产总值 + 来自国外的净要素收入，反映了现代产业结构的变化。

GNPDEF：国民生产总值平减指数 = 报告期价格计算的国民生产总值 / 不变的国民生产总值（以 2012 为基准），反映价值指标增减过程中与物量变动同时存在的价格变动趋势和程度的价格指数。

M2：广义货币供应量 = 流通的现金（M0） + 企业活期存款（M1）+ 准货币（定期存款、居民储蓄存款、其他存款），通常反映的是社会总需求变化和未来通胀的压力状态。

3MTB：三个月国库券，国库券的利率是市场利率变动情况的集中反映。

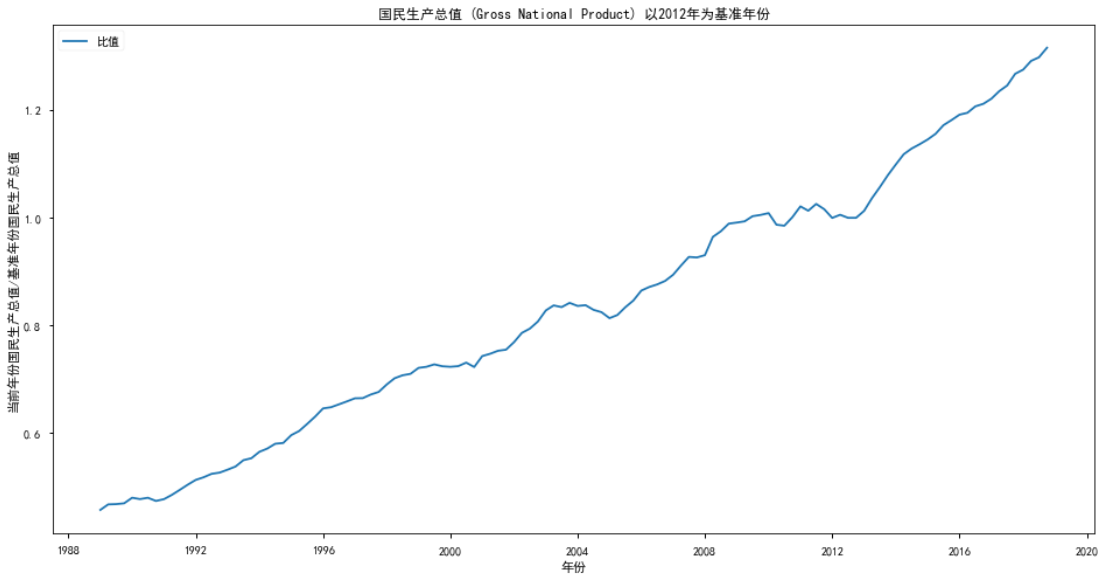
（4）预期结果

根据已有的近年经济数据，观察和根据符号回归进行预测，将两者相比较，得出公式的准确度，以便满足未来的需求。

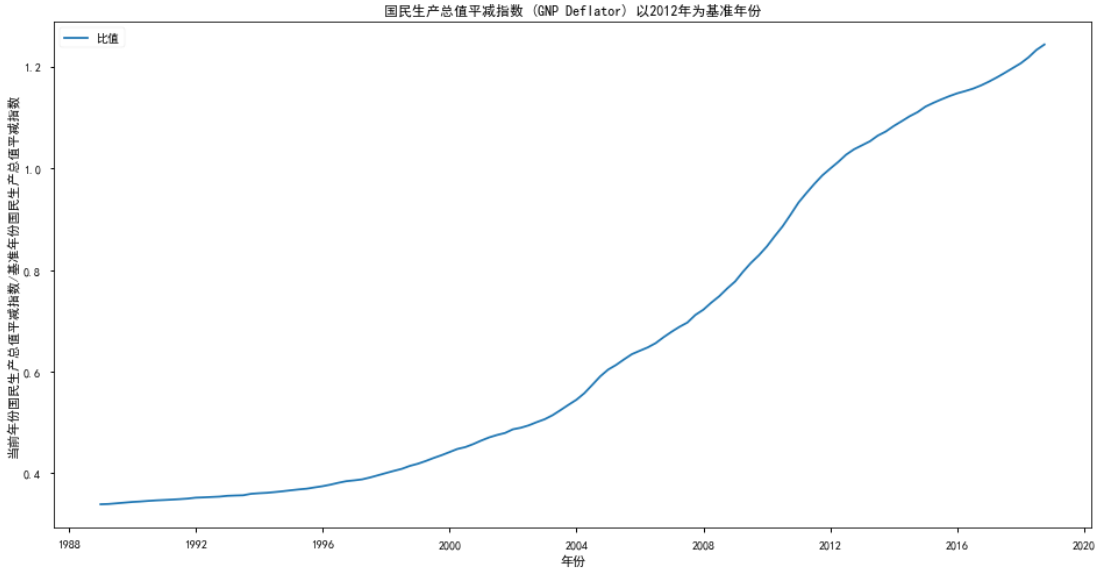
得出预测的模型以及预测得分，并与论文进行比较。

4. 结果分析

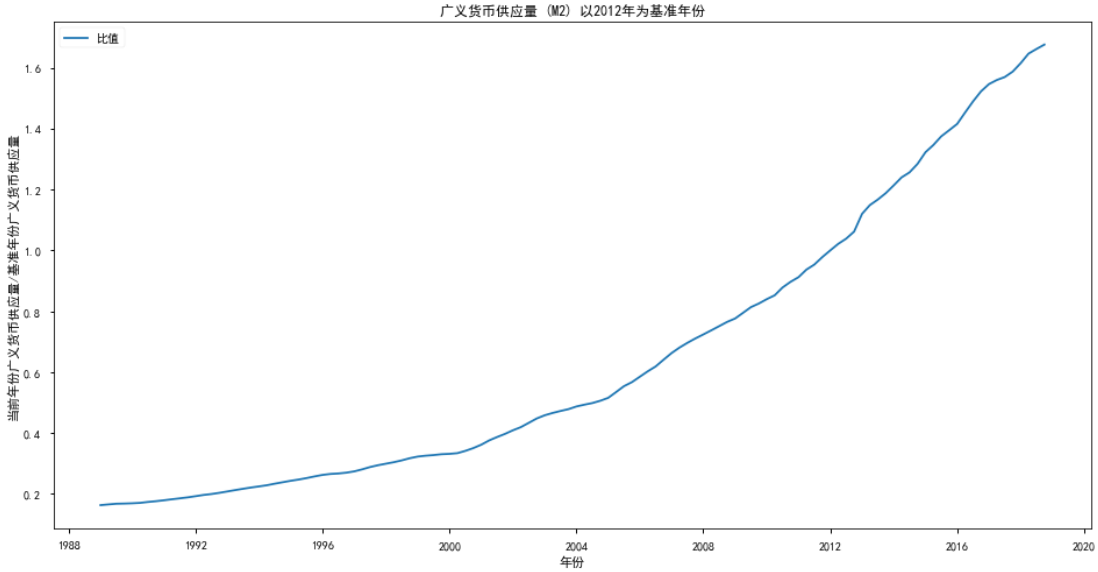
（1）国民生产总值数据图



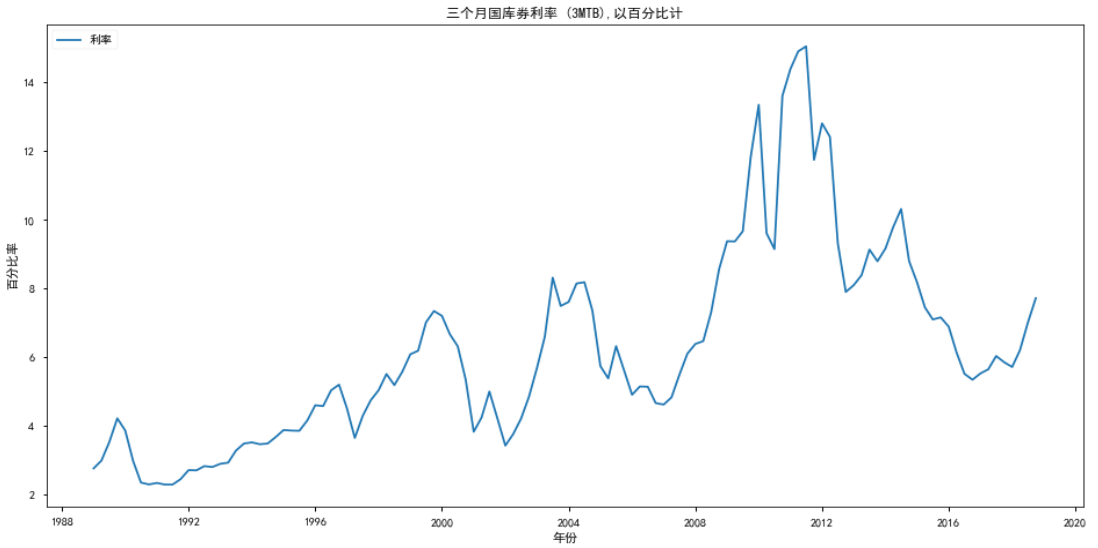
（2）国民生产总值平减指数数据图



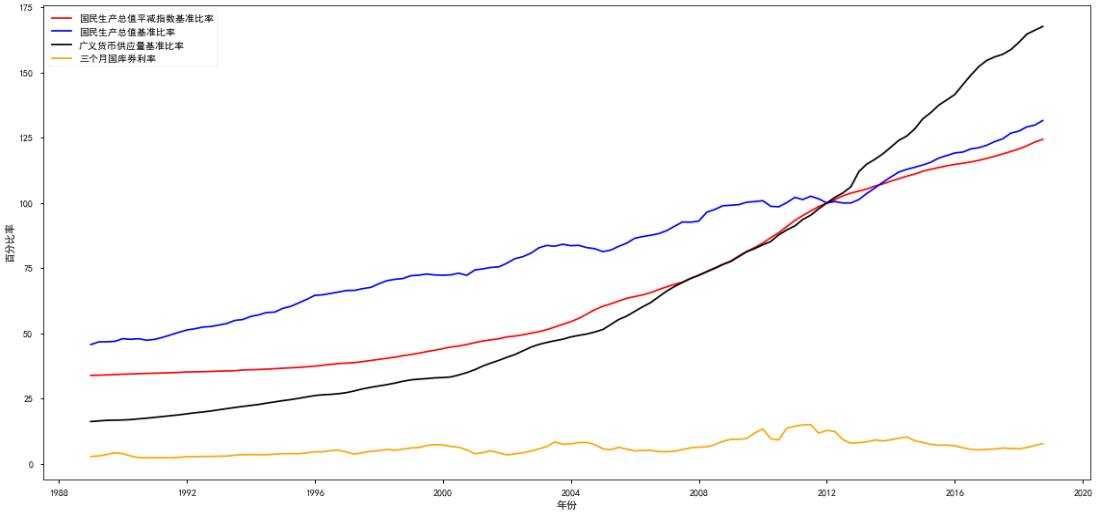
（3）广义货币供应量数据图



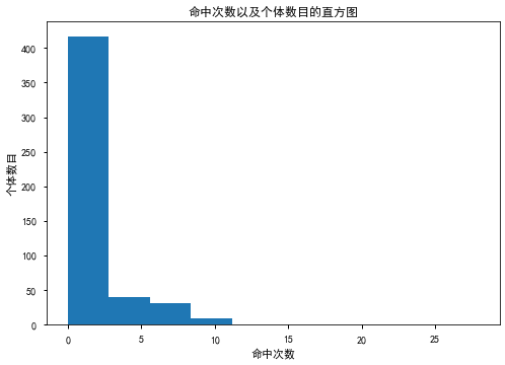
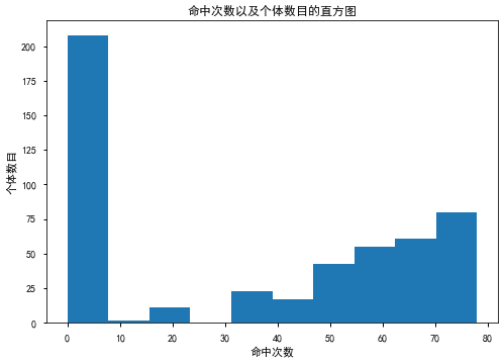
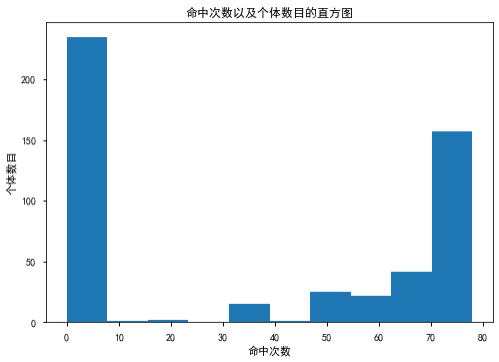
（4）三个月国库券利率数据图



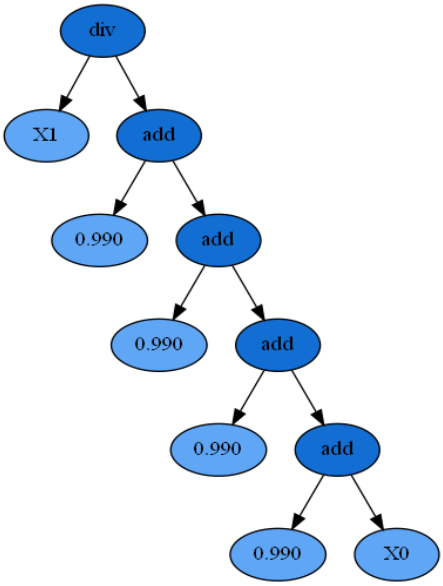
（5）整合总览数据图



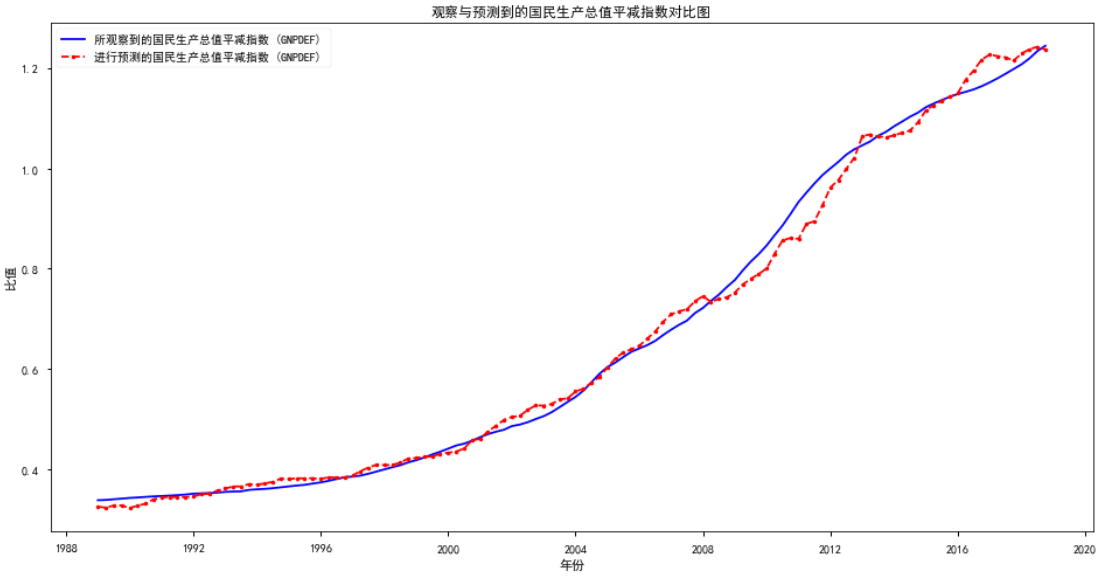
（6）遗传迭代过程图

迭代次数	分布图	说明
1		迭代次数过少导致个体命中次数过少
10		随着迭代次数的增加，高命中次数个体逐渐增多
20		种群分布随着个体的变化发生大幅度的改变

（7）二叉树表达式图



（7）观察以及预测模型对比图



（8）预测模型得分及平方和误差

说明：数据取自 1989-2018 这 30 年间，按照一年 4 个季度分，存在 120 个区间。

A．实现结果

数据范围	1—120	1—80	81—120
预测得分	0.99447	0.98916	0.92959
平方和误差	0.06143	0.09123	0.15823

B．论文结果

数据范围	1—120	1—80	81—120
预测得分	0.99348	0.99795	0.99061
平方和误差	0.07539	0.00927	0.06611

C．结论

虽然在总体的训练中借助最新的 gplearn 库精度略高于论文，但是在局部的训练中低于论文，可见在二十多年前的知识储备已经有了初步的模型，值得我们学习与探索。

六、参考资料

论文：

[1] [CN]遗传算法中适应度函数的研究

[2] [CN]用遗传算法求解旅行商问题及其代码设计

[3] [EN]基于标签传播的局部搜索遗传算法检测动态社区

[4] [EN]遗传编程是一种对计算机编程的自然选择的方法

[5] [EN] Tuning of the Structure and Parameters of Neural Networks using an Improved Genetic

[6] [CN]基于遗传算法的移动机器人路径规划研究_崔建军

[7] [CN]用遗传算法求解 TSP 问题_任昊南

网页：

[1] 基于遗传编程(Genetic Programming)的符号回归(Symbolic Regression)简介

[EB/OL]. https://blog.csdn.net/likehightime/article/details/5275264 -2021/01/02

[2] 遗传编程的示例（二次多项式的符号回归）

[EB/OL]. http://www.genetic-programming.com/gpquadraticexample.html -2021/01/02

[3] 遗传算法求解旅行商问题

[EB/OL]. https://zhuanlan.zhihu.com/p/137351343 -2021/01/02

[4] 百度百科----遗传算法

[EB/OL]. https://baike.baidu.com/item/遗传算法 -2021/01/02

[5] 遗传算法 (Genetic Algorithm)

[EB/OL].https://mofanpy.com/tutorials/machine-learning/evolutionary-algorithm/intro-genetic-algorithm/ -2021/01/02

[6] 遗传算法 (Genetic Algorithm) 例子 旅行商人问题 (TSP)

[EB/OL].https://mofanpy.com/tutorials/machine-learning/evolutionary-algorithm/genetic-algorithm-travel-sales-problem/ -2021/01/02

[7] 目标优化智能算法之遗传算法

[EB/OL]. https://tianle.me/2017/04/19/GA/ -2021/01/02

[8] 人工智能 8—遗传算法实验

[EB/OL]. https://blog.csdn.net/qq_43653930/article/details/103142930 -2021/01/02

[9] 经济机器是怎样运行的

[EB/OL]. https://www.youtube.com/watch?v=rFV7wdEX-Mo&feature=emb_rel_end -2021/01/02

[10] 【10 分钟算法】遗传算法-带例子和动画/Genetic Algorithm

[EB/OL]. https://www.bilibili.com/video/BV1yt4y1a7RY?from=search&seid=3179391201366450659 -2021/01/02

[11] 随机森林算法 OOB_SCORE 最佳特征选择

[EB/OL]. https://www.cnblogs.com/dinol/p/11614352.html -2021/01/02

[12] 利用 gplearn 进行特征工程

[EB/OL]. https://bigquant.com/community/t/topic/120709 -2021/01/02
[13] 用遗传算法实现符号回归——浅析 gplearn
[EB/OL]. https://zhuanlan.zhihu.com/p/31185882 -2021/01/02
[14] 使用 gplearn 自定义特征自动生成模块
[EB/OL]. https://zhuanlan.zhihu.com/p/76047703?from_voters_page=true -2021/01/02
[15] 【算法】超详细的遗传算法(Genetic Algorithm)解析
[EB/OL]. https://www.jianshu.com/p/ae5157c26af9 -2021/01/02
[16] 中国每年国民生产总值和货币供应量的数据
[EB/OL]. http://www.360doc.com/content/19/1108/18/34989057_871927918.shtml -2021/01/02
[17] Graphviz 的安装及纠错
[EB/OL]. https://blog.csdn.net/qq_28409193/article/details/79880886 -2021/01/02
[18] STATISTICS AND COMPUTING
[EB/OL]. https://www.shengsci.com/sci/6408.html -2021/01/02
[19] 优化算法系列-遗传算法（1）——基本理论枯燥版本
[EB/OL]. https://www.cnblogs.com/haimishasha/p/9816735.html -2021/01/02
[20] TSP 问题—启发式遗传算法
[EB/OL]. https://www.jianshu.com/p/b3cd8e674ff0 -2021/01/02