# 论文要求

一、毕业论文（设计）必须经过答辩才能评定成绩。

二、毕业论文（设计）必须打印。文中所用的公式、图表及程序代码，在条件许可时，应打印输出。

三、撰写 500 字左右的中文论文摘要，提倡以中外两种文字书写，外文摘要附在中文摘要之后，关键词一般 3-5 个。

四、毕业论文（设计）一律装订为横开本，左侧装订。

五、文中所用的符号、缩略词、制图规范和计量单位，必须遵守国家规定的标准或本学科通用标准。作者自己拟定的符号、记号缩略词，均应在第一次出现时加以说明。

六、注序要与文中所提的页码一致，序号不能用[1],[2],[3]等数码表示，以免与参考文献的序号相混淆。

七、文后要注明参考文献和附录，参考文献要写明作者，书名（或文章题目及报刊名）、版次（初版不注版次）、出版地、出版者、出版年、页码，序号使用[1],[2],[3]……，中译本前要加国别。

**吉林大学教务处制**


吉林大学

# 本科生毕业论文（设计）

中文题目 基于流形和回归遗传算法的改进

英文题目 Improvement of Genetic Algorithm Based on Manifold and Regression

学生姓名 唐梓峰 班级 五班 学号 55160530
学　　院 软件学院
专　　业 软件工程
指导教师 韩宵松 职称 讲师

# 吉林大学学士学位论文（设计）承诺书

本人郑重承诺：所呈交的学士学位毕业论文（设计），是本人在指导教师的指导下，独立进行实验、设计、调研等工作基础上取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的作品成果。对本人实验或设计中做出重要贡献的个人或集体，均已在文中以明确的方式注明。本人完全意识到本承诺书的法律结果由本人承担。

学士学位论文（设计）作者签名：

2020 年 5 月 20 日

# 基于流形和回归遗传算法的改进

## 摘要

约翰与他的学生在上世纪七十年代年发明了一个全新的全局搜索模型，并称之为Genetic algorithm[1]（下文称遗传算法）。该算法参考了生物界中“优胜劣汰”的原则，利用相应的适应度函数评价群体中每一个独立的个体，所计算出的值被称为个体适应度，同时利用相应的规则（如轮盘法则），使得个体被保留下来的可能性随着个体适应度而增大。此后再通过模拟大自然在进化过程中所发生的一些现象，例如染色体的交叉重组、染色体上的碱基上发生的变异等等，对原有的数据进行相关的一些操作，继而能产生解空间上产生新的解，从而达到全局搜索的目的。通过不断的循环进行进化，更加接近全局最优解的个体被留存，则最终能得到一个全局最优解或一个近似的最优解。

遗传算法等这一类优化算法，没有对于欲求最优解的目标函数的连续性以及可导性的要求，且同时具有较为卓越的并行操作性，该算法通过每一轮的迭代，群体中个体的不断进化，从而达到能够有进行全局搜索的一个目的，且搜索的方向具有一定概率意义。同时，从使用的便捷角度考虑，利用该算法时对所求问题并不需要有特别深入的了解，只需在种群初始过程中随机生成个体，算法即可开始全局上的最优解搜索。

然而对于一些求解较为复杂的问题，譬如预测金融市场走势等，问题往往建立在高维度的空间上，需要不断的调用适应度函数,同时适应度评价无法以具体公式的形式给出，需要借助额外的辅助手段，然而我们常常无法弄清楚辅助工具的内部机制，这类问题被称为HEB（即问题是建立在高维度上、算法时间复杂度较高、算法本身内部的机制难以摸清）问题。面对这一类问题，虽然利用遗传算法可以计算求得到一个最佳解决方案，然而常常由于需要面对大量个体适应度所带来的庞大计算量，往往计算要消耗大量的时间成本，并且适应度计算也较为依赖于外部的辅助手段，从而使得进行迭代的成本变得进一步高昂。多方面的原因使得遗传算法在求解上述问题时常常表现效果不尽人意。

为了解决上述较为复杂一类的问题，一种经典思路是采用基于多元线性回归等

机器学习相关的理论的代理模型，来预测个体的实际适应度值，进而达到显著减少调用适应度函数次数的目的。然而，很多时候实际的实践中，我们所面临的往往是一个高维的复杂数据集，直接使用该数据集进行训练往往很难达到一个较为精准的效果。

为了达到同时为了避免训练代理模型时所使用的样本较多且分散不均匀所导致的准确降低的问题，在使用流形学习进行降维之后，再使用 AP 聚类选取一些较有代表性的样本进行训练。

**关键字：** 遗传算法，多元线性回归，流形学习，AP 聚类

# Improvement of Genetic Algorithm Based on Manifold and Regression

Author:Tang Zifeng

Tutor:Han Xiaosong

## Abstract

In the 1970s, John and his students developed a new global search model called Genetic algorithm [1].

This algorithm refers to the principle of "survival of the fittest" in the biological world, and USES the corresponding fitness function to evaluate each individual in the population. The value calculated is called the individual fitness. At the same time, it USES the corresponding rules (such as the rule of roulette wheel) to make the possibility of the individual being retained increase with the individual fitness.

After that, some phenomena occurred in nature during the evolutionary process, such as the crossing and recombination of chromosomes and the variation of the bases on chromosomes, were simulated, and some relevant operations were carried out on the original data to generate new solutions in solution space, so as to achieve the purpose of global search.

Through continuous cycles of evolution, individuals that are closer to the global optimal solution are retained, and eventually a global optimal solution or an approximate optimal solution can be obtained.

Etc. This kind of optimization algorithm, genetic algorithm (ga) not to desire the continuity of the objective function of optimal solution and the conductivity of the requirements, and have more excellence parallel operation at the same time, the algorithm through each round of iteration, the evolution of the individuals in the group, so as to achieve to have a purpose, for the global search and search in the direction of the probability of a certain significance.

At the same time, from the perspective of convenience of use, the algorithm does

not need to have a particularly deep understanding of the problem to be solved when using the algorithm. As long as individuals are randomly generated in the initial population process, the algorithm can start searching for the optimal solution on the whole.

But for some solving more complex problems, such as financial market movements, etc., often built in the high dimension space, need to keep the invocation of the fitness function, at the same time, fitness evaluation cannot be in the form of specific formula, need additional auxiliary means, however, we often fail to understand AIDS, the internal mechanism of this kind of problem is called HEB (i.e., the problem is built on high dimension, high time complexity, the algorithm itself the internal mechanism is difficult to find out the problem.

It is well known that when the fitness function is relatively complex, the optimization time cost of the genetic algorithm will be extremely large. To address this issue, the surrogate model was employed to predict the fitness value of the optimization problem in order to reduce the number of actual calculated fitness values. In this paper, a MLR(Multiple Linear Regression), the manifold learning method, and Affinity Propagation clustering algorithm were fused in the genetic algorithm to evaluate partial individuals' fitness. Sufficient benchmark numerical experiments were conducted, and the results proved that the strategy could reduce the number of calculations of the fitness function with similar accuracy to that of a simple genetic algorithm.

Facing this kind of problem, although can be calculated by using the genetic algorithm and get a best solution, however, often because of the need to face a large number of individual fitness brought about by the huge amount of calculation, often to calculate the cost of consumes a lot of time, and fitness calculation are dependent on external auxiliary means, which makes the cost of the iterative further high.

For many reasons, genetic algorithm is often unsatisfactory in solving the above problems.

In order to solve the above complex problems, one of the classic ideas is to use the proxy model based on multiple linear regression and other machine learn-related theories to predict the actual fitness value of individuals, so as to significantly reduce the number of fitness function calls.

However, many times in practice, we are often faced with a complex data set with high dimensions, and it is often difficult to achieve a more accurate effect by directly using the data set for training.

In order to avoid the problem of accurate reduction caused by the large number of samples and uneven distribution in the training agent model, AP clustering was used to select some representative samples for training after the dimension reduction by manifold learning.

**Keywords**: Genetic Algorithm, Multiple Linear Regression, Manifold Learning, Affinity Propagation Clustering Algorithm

# 目　　录

# 第 1 章　绪论

## 1.1　研究背景及意义

不管是在日常生活中，亦或是在科学研究中我们常常需要去面对各种各样的优化问题。所谓最优化问题，其一般性的描述指的是在给定的相应限制性条件下，得到欲求问题的最优解。譬如物流公司在运输货物的过程中如何合理安排合适的车型以及数量，并规划路线，才能达到以最快的速度将货物送至目的地，同时最大程度地减小运输成本；其他类似的问题还有例如涉及药物的最佳靶点、如何合理规划城市中的学校、工厂、居民区等等其他的不同区域。在科研、工程等领域，随着科学家和工程师们想要解决的问题、设计的系统越来越复杂，伴随而来的是问题的规模也愈发的庞大，复杂度也相应的变高。而传统的优化类算法例如微分极值法、拟牛顿法、梯度下降法等算法虽然具有较为优秀的局部寻优能力，然而只能在解决线性、较低维度、单极值等相对较简单问题时能够有较为不错的表现，在复杂的高维度问题中往往很难找到最优解。因而迫切地需要一种有效的算法来解决这一类的问题，启发式算法应运而生，本文所重点讨论的遗传算法就是一类启发式算法，其他的还有粒子群算法、模拟退火算法等等。

遗传算法通过不断的在解空间众迭代向更优的解集方向进行“进化”从而寻得最优解。GA 起始于包括一些潜在的解决方案个体的种群，其中每个解决方案个体即为染色体，由一组基因编码所构成。接着依照适者生存的规则，对群体中的每个个体进行个体选择，染色体交叉重组，染色体上碱基对发生变异，等相应操作，进而产生潜在的包含更优解的群体。如此往复，当收敛于特定值，或者迭代次数达到预先所设定的最大进化代数，选择最有个体并将其解码，作为欲求问题的一个最优解。GA 从其被提出之时开始，因其优秀的全局搜索能力，已经被成功地运用于众多研究、工程领域。

对遗传算法的流程步骤分析我们很容易发现，传统遗传算法的每一轮迭代都必不可少地需要大量的适应度计算，尤其在面对一些较为复杂的适应度函数时，该算法的时间成本将变得十分高昂，且许多来自工业领域的优化问题还常常面临高维度的参数，要求较为频繁地调用适应度函数。同时，在很多实际生产生活中

的工程应用里，欲求问题的目标函数常常是需要多条件来约束的一个复杂函数，在某些较为特殊的情况下甚至没有办法通过写成目标函数的形式来表达，譬如在金融市场等等领域，适应度计算过程往往往往需要借助具备局部优化能力的特定辅助工具给出，然而这一类软件其自身运算就需要耗费大量的时间成本，而在传统遗传算法中，每一轮的迭代中对于个体的适应度评价就需要调用一次相应的适应度函数，往往会造成相当巨大甚至无法承受的时间成本开销，因而该困境已成为影响遗传算法提高性能以及应用的推广的首要障碍。

因此，要解决该困境（调用适应度函数次数过多）的一个最直观的解决方案就是要尽量地去减少适应度函数的调用次数。然而，如果只是单纯地减少调用适应度函数的次数，那么这将意味着群体的迭代次数或着群体规模将变少，将导致算法的全局搜索能力有显著地下降，并最终降低算法的性能。而从另一方面来说，当我们纵观遗传算法的优化迭代过程，每一个新个体的产生都是建立在前一代个体的基础之上，换而言之，在搜索所求问题的潜在解空间时，是需要建立在之前已经搜索过的区域解的基础之上，才能对未知解空间进行预测性探索。巧合的是，这个过程恰好时机器学习和模式识别关键的问题。继而我们可以联想到通过借助机器学习的一些相应方法，通过构造对应适应度的预测模型，从而达到既能保持原有群体个体数量不变，且显著地减少对于适应度函数的调用，最终事先减少算法计算时间成本的目的。对于这样的一个利用已知解区域进行学习并将其用来预测未知的区域解的模型，我们将其称为代理模型。

对于上述代理模型，即位通过利用已知解空间的解进行训练，并利用相关的机器学习方法（如多元线性回归）来构建相应的预测模型，达到部分个体不需要调用适应度函数但仍可以预测尚未搜索区域解分布情况的目的，从而能够进一步引导遗传算法在迭代过程中的进化方向，使其能够避开局部极值并往全局最优解方向进化。该代理模型在工程实践中有重大的意义，因为它在不显著地牺牲一定准确度的情况下，能够大量缩短算法的计算时长，提高整体的运行效率，使得遗传算法能够更好的在实际工程中被应用。

## 1.2 国内外研究现状

国内外许多的研究者给出了很多不同的研究方案。Grefenstette 等人在 1985 年提出了让 GA 算法通过部分预测适应度的一个模型，达到降低实际适应度调用

次数的目的，并将其运用在优化数字减影技术中[2]。1997 年，Papadeakakis 等人提出了利用神经网络模型作为代理模型来对优化问题的适应度进行预测，进而能够减少真实适应度的调用次数[3]。2000 年，Rasheed 提出了一种采用适应度逼近模型来对遗传算法进行初始化操作，以及指引其在解空间中的进化方向的算法[4]。一年之后，由 Kim 等人研究的聚类遗传混合算法，表现出了优良的优化性能[5]。2005 年，Branke 基于提前获得的临近解空间的个体的适应度值，来预测当前染色体的实际值，并且通过在拟合中进行插值以及回归，最后也获得了优良的实验表现[6]。在此之后，Branke 和 Jin 等人还一起合作，根据之前的适应度估计的理念，开展了一些基于非确定环境下的实验[7,8]。2008 年，Micheal 等人提出了一个基于混合进化适应度估计模型，这个模型参考了群智能算法的理念来构建模型的参数，最后在采用该模型去预测部分未知个体的适应度，同样得到了很不错的实验结果[9]。2012 年，Xiaosong Han 等人创新性地利用了亲和传播聚类算法，将其与传统的遗传算法相结合，提该该方法展现了优秀的性能[10]。最近的几年以来，该领域引入了一些用于替代适应度函数的算法，并取得了很多研究成果[11]。

## 1.3 论文结构及创新点

### 1.3.1 组织结构

本文内容从结构上一共将被分为五章：

当前章节为绪论，主要介绍了遗传算法提出缘由以及相应的生物学一处，并且概述了遗传算法的具体工作流程以及其对应的不足之处；讨论了代理模型的存在意义以及国内外研究者在该领域的研究现状，并给出了本文的结构。

第二章为综述，将介绍了与本文实验相关的算法理论，包括经典遗传算法的理论、结构、工作流程，多元线性回归，降维与流形学习，以及 AP 聚类方法。

第三章将提出一种基于流形和聚类的回归适应度代理模型。本章首先概述了与有关算法相应的背景理论知识以及详细的实现细节，同时详述了本文提出的代理模型，并进行理论分析、算法流程以及算法的复杂度分析。

第四章则为实验部分，引入了多个基准函数，对运用了不同方法的代理模型以及传统遗传算法进行测试，多次实验后，得出并分别比较算法的真实适应度计算次数，适应度的最优值以及多次实验最优值的适应度标准差。

第五章则对全文进行了展望总结，本章节结合了第四章的实验数据，进一步分析本文所述的代理模型的优缺点，讨论了算法未来可能的改进方向。

### 1.3.2 创新点

本文的主要创新点，是为了针对目前遗传算法中适应度函数过于复杂而导致需要消耗大量的计算时间成本的问题，结合现有的一些机器学习以及模式识别的方法，通过对于已经搜索过的区域的训练，构建出一个适应度估计模型，用来估计未知解空间的个体适应度，从而引导遗传算法的进化方向。详细过程如下：

将高维空间中较为消耗时间成本的适应度计算问题通过流形学习算法映射到低维空间中，转换为一个简历适应度预测估计模型的问题。本模型采用最小二乘回归，且模型如上所述建立在低维空间中（本实验中为二维或三维），则可以在保证模型的预测精度下较高效率地建立估计模型。同时，该训练集的可代表性决定了模型相应的泛化能力，因此，本文的训练模型同时采用了 AP 聚类算法来生成合适的训练集作为估计模型的参数。

# 第 2 章　相关理论介绍

## 2.1　遗传算法理论

遗传算法(GA)作为一类全局搜索技术，常常被运用于寻求问题的最优或者近似最优解。可以用“生成”与“检测”模式来体现遗传算法典型的迭代过程，如下所示为该算法的基础框架：通过模式定理可以说明其进化寻优的迭代过程，详细算法可以参见[12]。

详细的算法迭代工作流程如下所示：

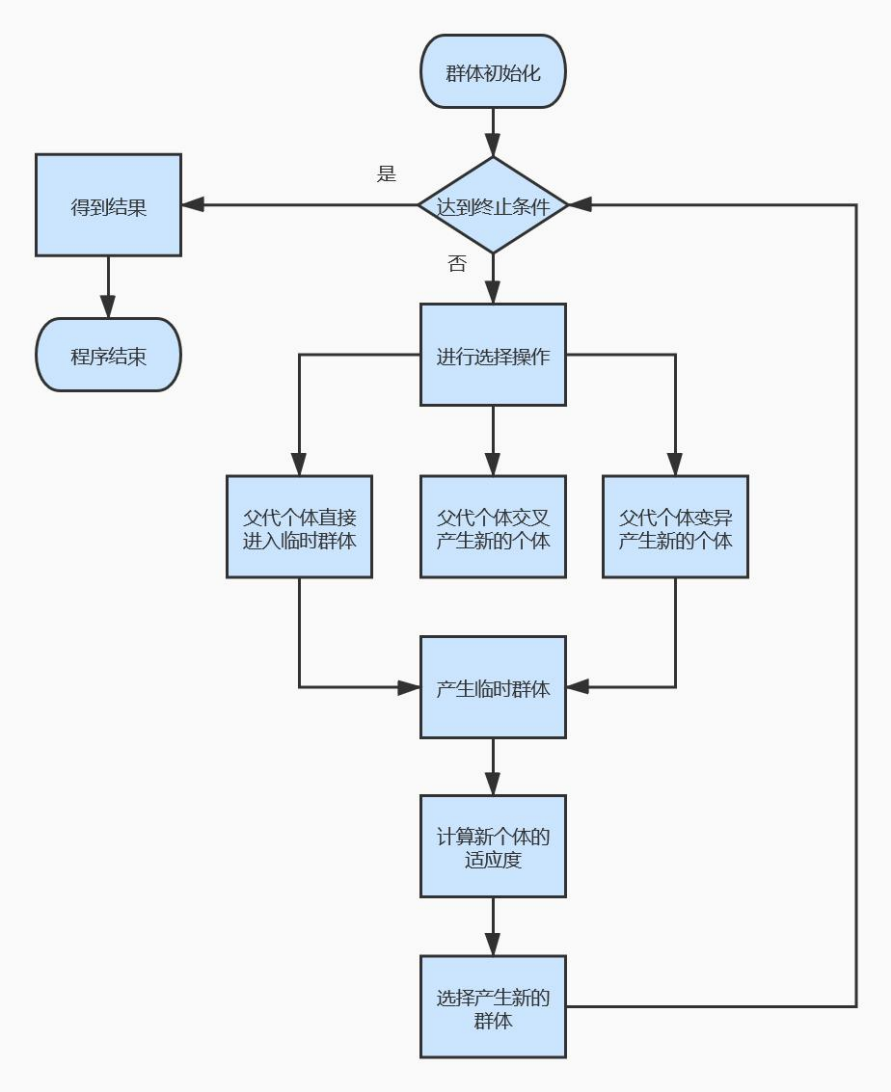


图 2-1　遗传算法流程

1. 首先，需要随机初始化种群中的个体；
2. 为了评价每一个个体在群体中的适应能力，需要事先定义一个满足要求的

适应度函数，之后个体优劣的评判都取决于适应度值的大小。

3. 根据适应度函数评判后的结果，我们优先选择适应能力更优秀的个体，然后将其不同染色体之间依照概率进行重组，并随机地对基因进行突变，从而在解空间上产生一个新的解决方案（个体），形成新的种群进行下一轮的迭代过程。应当注意到的是，此时产生的新个体是向随机方向进化（搜索）的，因此并不能不能保证新的个体一定比父代个体适应度更高，只是提供了一种选择的可能性，避免了过快陷入局部最优。

4. 在每一轮迭代的最后时刻，算法将采取通过一些特定的法则（如俄罗斯轮盘法则）使得适应度较低的个体更有可能被淘汰掉，反之则更能适应环境（即有较高适应度）的个体有更大几率被保存。

5. 两种情况将导致算法的结束：第一种是算法达到了最初的预期效果，找到了满足条件的解；另一种则是算法的迭代次数已经达到初始设定的次数而跳出循环，有可能尚未找到最优解。

当达到最大的迭代次数,或获得满足条件的最优解时，循环停止，同时返回所求的最优解。

在第一轮的迭代中，算法中的初始群体由许多随机生成的个体构成。对于群体设定的规模大小往往也没有固定规模限制，需要视具体不同的问题需求而定。譬如对于解空间较小的问题，往往不需要太大的群体规模；而对于一些及我为复杂的问题，其解空间比较大时，则一般需要定义较大种群规模，从而利于算法更广泛高效地搜群全局最优解。

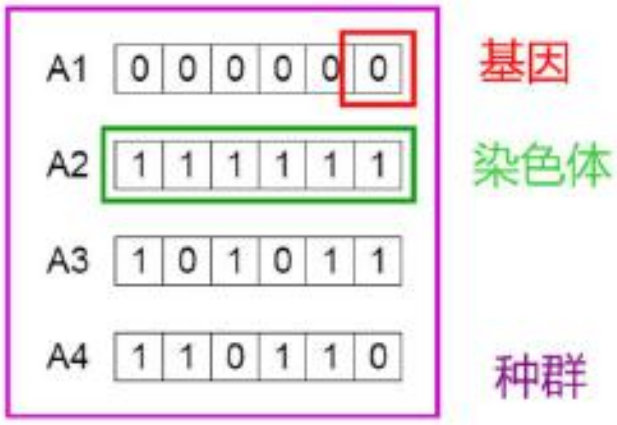


图 2-2　初始化后的群体

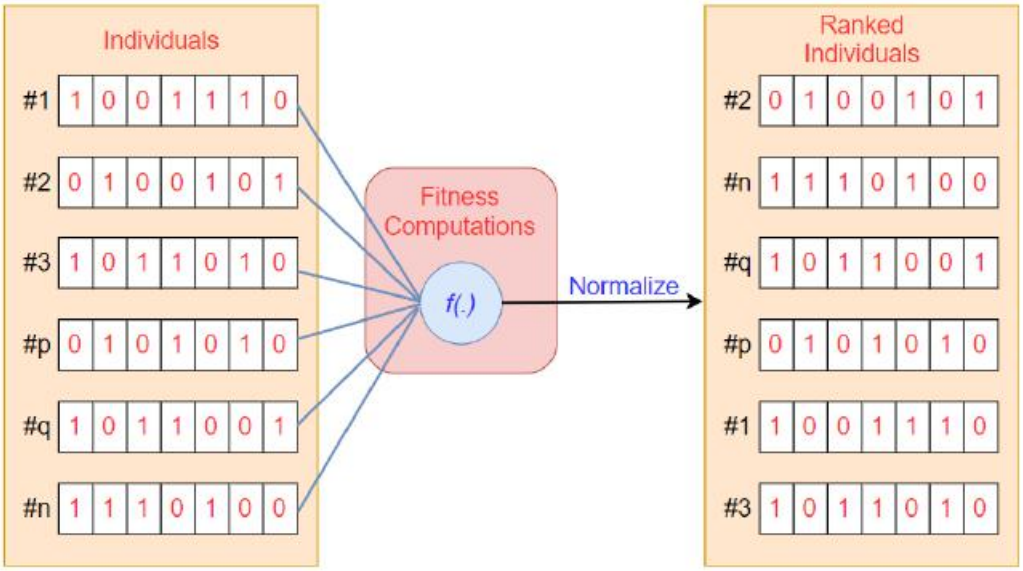


图 2-3　适应度函数的作用

个体适应度的高低由适应度函数由适应度函数来评判。譬如在著名的背包问题之中，一个旅客准备出门，他想尽可能夺得携带有价值的物品，从而使得背包中的物品的贵重程度达到最高，然而背包本身是有容量的最大限度的。为了解决该问题，可以定义一个数组，其中每个元素只能时 0 或者 1，其中 0 代表背包中不含某个物品，而 1 代表背包中包含了某个物体。则我们最后通过该数组来计算当前的组合方式下背包中的总价值是多少。此时同时要注意背包中的物体是否会超出先前背包设定的最大容量，从而使得所求得的且并不符合欲求问题的限制条件。另外可能会面临一些很难以适应度函数描述的一些复杂问题，例如前文所述的 HEB 问题。另外值得注意的是，传统的遗传算法中，适应度恒为非负的，因此我们需要输出负数值的函数加以处理，使得其结果为非负。在后面的章节及将会详述相应的处理方法。

我为了避免减少种群的多样性，同时确保种群中高适应度的个体更容易被保存下来，我们一般选择使用轮盘赌的方式来选择个体。

假设有这样的一个轮盘，我们将其分为 m（群体中个体的数量）个分区，其中每个个体所对应的面积和其相应的适应度值成正比。

表 2-1　染色体面积占比

| | 适应度值 | 对应面积 |
|---|---|---|
| 染色体 1 | 28 | 28.9% |
| 染色体 2 | 23 | 23.7% |

| | 适应度值 | 对应面积 |
|---|---|---|
| 染色体 3 | 12 | 12.4% |
| 染色体 4 | 34 | 35.1% |

基于上述值，可以构造一个轮盘赌轮。

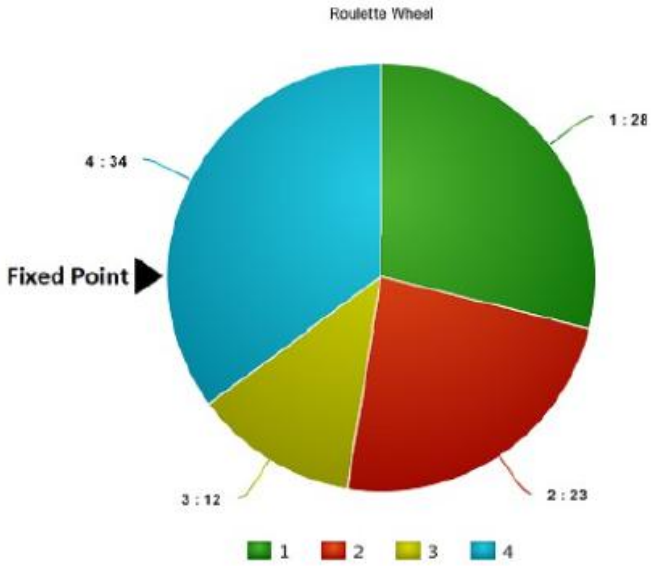


图 2-4　只包含一个锚点的轮盘

此时则可以想像将该轮进行旋转，最后选取锚点所指定的区域对应的个体作为其中一个父代个体。重复上述过程，则可获得第二个父代个体。

在另一种情况下，我们可以尝试选择两个锚点，如下所示：

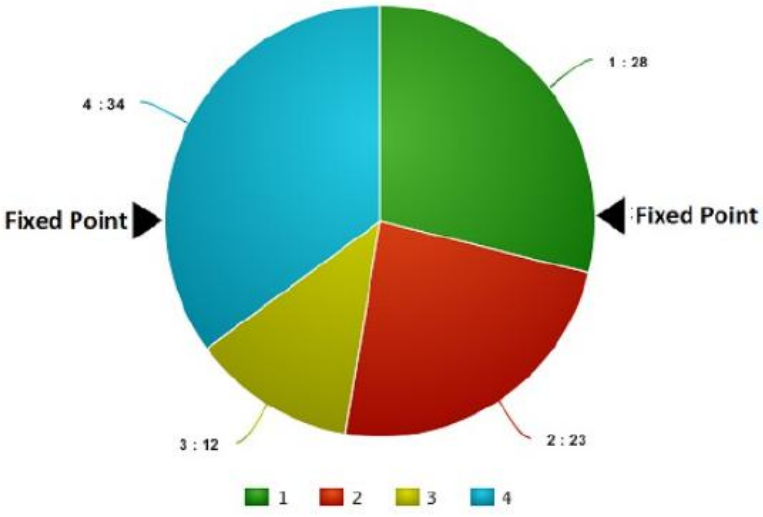


图 2-5　包含两个锚点的轮盘

上述方法被称作为随即通用选择法，通过该方法可以一次性地得到两个父辈个体。

在之前的一步中，我们论述了如何对将父代染色体进行选择的具体流程。而从生物学意义上来看，染色体进行交叉的过程亦是生物繁殖过程中的一个重要环节。故我们将从上一步通过选择得到的染色体 1 以及染色体 4 进行交叉。具体过程如下所示：

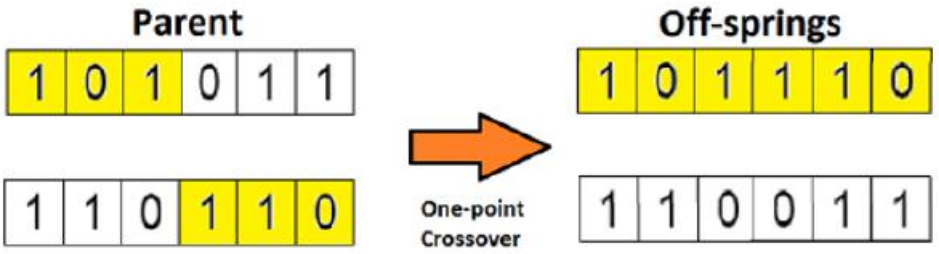


图 2-6　单点交叉

上图所示的交叉方式被称作单点交叉，是最基础的一种交叉形式。在交叉的过程中，染色体中一个随机的位置将被作为锚点，并以此锚点对两个染色体进行交换，进而能够产生一个新的子代个体。

除了上述的单点交叉法，我们还可以选择适应多点交叉法，顾名思义即有多个随机交叉点。下面我们以两个交叉点作为例子：

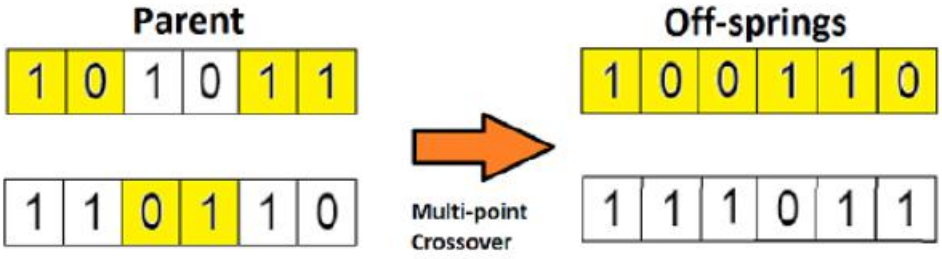


图 2-7　多点交叉

然而事实上从生物进化的角度来看，子代的染色体并不是完全地将父代的染色体继承并随机组合。事实上，染色体上的基因常常会小概率地发生变异，这给生物的多样性提供了更大的可能。同样的，在遗传算法里也引入了变异的概念，将对染色同中的基因进行随机变化，使得算法的全局搜索能力变得更强。下面是一个变异过程的简单例子：


图 2-8　变异操作

总结来说，遗传算法的整个工作流程大致如下：

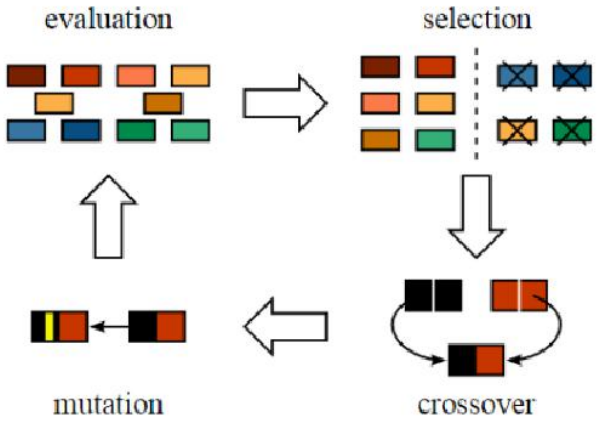


图 2-9　GA 一次迭代中的流程

通过上述的几个步骤，遗传算法将产生新的临时群体，此时调用适应度函数对种群进行评估，淘汰原种群中适应度较低的个体，产生新的种群，进行下一轮的迭代。

## 2.2　多元线性回归

多元线性回归囊括了多个变量，如 $X_1,X_2,X_3,...,X_p$。较为直观地表达了不同属性在进行预测时的重要性，因此其可理解性较好。譬如要评判一个哈密瓜是不是好瓜，其评价标准往往不仅取决于色泽，还需要考虑根蒂以及敲声等来判断一个好瓜。通过训练得到的不同属性的所占权重，即可以确定哪些是更重要的自变量，即哪些属性的变化会对因变量产生更大的影响。因此，为了便于进一步的用于预测目的，我们对因变量和自变量之间建立的数学关系。鉴于多个影响因变量的自变量包含于模型之中，其间的关系可以通过线性回归得到，对该模型整体拟合检验过程，则可以通过预测适应度标准差、回归系数以及对系数进行判定($R^2$)来进行实现。

可以假设$(Y_1,X_1),(Y_2,X_2),...,(Y_n,X_n)$为定义域内的 n 个点。我们可以据此给出一个简易的线性回归模型：

$$Y = a + bX + e \quad \text{.....(2.1)}$$

e 表示为回归误差，其值代表回归模型 $Y = a + bX + e$ 于实际的 $Y$ 之间的差值。我们可以通过最小二乘法使得误差 e 最小化。根据上述给出的式子，对于以$(Y_i,X_i)(i=1,2,..,n)$来表示 n

个样本点，继而能得到一个简易的回归模型：

$$Y_i = a + bX_i + e_i \quad (2.2)$$

继续上述我们关于判断一个哈密瓜是不是好瓜的例子，一个好瓜往往取决于好几个变量，如含水量、色泽、敲声、根蒂等等。此时我们使用多元线性回归来构建一个模型：

$$Y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 + e \quad (2.3)$$

其中 $Y$ 表示哈密瓜的优劣程度，$X_1$ 表示含水量，$X_2$ 表示色泽，$X_3$ 表示敲声，则可以称之为一个多元线性回归模型，其中包含了三个回归/独立变量，其应变量 $Y$ 与未知参数 $B_0$，$B_1$，$B_2$,$B_3$ 呈线性关系。由上述模型可以推广到任意自变量的多元线性回归模型。我们假设 $Y$ 为应变量，$X_1$,$X_2$,$X_3$,...,$X_p$ 为 p 个自变量，则可以得到如下的一个多元回归模型：

$$Y = B_0 + B_1X_1 + B_2X_2 + \cdots + B_pX_p + e \quad (2.4)$$

在上述式子中，参数 $B_0$，$B_1$，$B_2$,...,$B_p$ 被称之为回归系数。即每当其余的回归变量均为常数时，$X_i$ 的值变化与预测中 $Y$ 的变化成比例。从出于方便的角度考虑，可以在式子中附加一个虚变量 $X_0$ 以及截距 $B_0$。对于所有的 n 次预测，$X_0$ 的取值均为 1。则上述模型 2.4 可以以如下形式进行表述：

$$Y = B_0X_0 + B_1X_1 + B_2X_2 + \cdots + B_pX_p + e \quad (2.5)$$

在上述多元线性回归模型 2.5 中，可以考虑一个简单线性回归的特例，即当 $X_0$=1，$B_0$=a，$B_1$=$b$ 且 $B_i$=0，($i$ 大于等于 2）。而对于参数 $B_0$，$B_1$，$B_2$,...,$B_p$ 的解释为，在其他独立变量保持不变的情况下，单个参数表示对应变量的单位变化的 $Y$ 变化量。我们将这些参数定义为偏回归系数。顾名思义，我们称上述模型为多元线性回归，因其包括两个或以上的多个变量，且与参数 $B_0$，$B_1$，$B_2$,...,$B_p$ 呈线性关系。进一步分析，$X_i$ 可以推广为任意的连续函数，例如 $logX$ , $X^{-1}$,$X^2$,$X^3$ 等等。然而必须注意到，此时的等式仍应为线性的。根据上述分析，我们给出如下一个多项式模型：

$$Y = B_0 + B_1X_1 + B_2X^2 + \cdots + B_pX^p + e \quad (2.6)$$

此时若让 $X_1$=$X$, $X_2$=$X^2$, $X_3$=$X^3$ 等等，即可带入等式 2.5 的线性模型。应当说明，在该模型中我们认为 e 为正态且独立的一个分布，平均值为 0，方差为$\sigma^2$。

我们可以利用均方误差来求 2.5 中的回归系数。均方误差在几何意义上有很好的表现，因为其基于较常使用的欧几里得距离。我们熟知的最小二乘法积，就是

通过求解最小均方误差来实现的。在上述的线性回归模型中，利用最小二乘法，试图找到一条直线，使得样本上所有的观测值距离该直线的欧几里得距离最小。设定观测的次数为 n(>p)，$y_i$ 表示第 i 个实际的观测值，$x_{ij}$ 表示关于回归变量 $X_i$ 的第 $j$ 个实际观察值。训练数据如下所示：

**表 2-2　训练数据示例**

| 响应变量 Y | 回归变量 | | | |
|---|---|---|---|---|
| | $X_1$ | $X_2$ | … | $X_p$ |
| $\boldsymbol{y_1}$ | $x_{11}$ | $x_{21}$ | … | $x_{p1}$ |
| $\boldsymbol{y_2}$ | $x_{12}$ | $x_{22}$ | … | $x_{p2}$ |
| $\boldsymbol{y_3}$ | $x_{13}$ | $x_{23}$ | … | $x_{p3}$ |
| ⋮ | ⋮ | ⋮ | … | ⋮ |
| $\boldsymbol{y_n}$ | $x_{1n}$ | $x_2$ | … | $x_{pn}$ |

那么可以构建第 $i$ 次的多元回归模型如下：

$$Y_i = B_0 + B_1X_{1i} + B_2X_{2i} + \cdots + B_pX_{pi} + e_i,\ i = 1,2,\cdots,n \quad \text{.....(2.7)}$$

这当中的 $X_{1i}, X_{2i}, ..., X_{pi}$ 是 $p$ 个回归系数所对应的值，而 $B_0$ 为对应的截距，而参数 $B_0$，$B_1$，$B_2$,...,$B_p$ 则是与自变量 $X_1$,$X_2$,$X_3$,...,$X_p$ 相对的回归系数。接下来进行 e 的最小化操作。容易推算出，模型 2.5 的均方误差公式为：

$$E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}\left(Y_i - B_0X_{0i} - B_1X_{1i} - \cdots - B_pX_{pi}\right)^2 \quad \text{.....(2.8)}$$

为了能够计算出模型的具体参数，分别使得 $E$ 对于 $B_0$，$B_1$，$B_2$,...,$B_p$ 进行微分，使得其结果计算为 0。求出 $E$ 关于 $B_j$ 进行求导的正规方程为:

$$\frac{\partial E}{\partial B_j} = -2\sum_{i=1}^{n}\left(Y_i - B_0X_{0i} - B_1X_{1i} - \cdots - B_pX_{pi}\right)X_{ji} = 0, j = 0,\cdots,p \quad \text{.....(2.9)}$$

将上述等式 2.9 进行化简，可以求得一个最小二乘正规等式：

$$
\begin{aligned}
nB_0 + B_1\sum_{i=1}^{n} X_{1i} + B_2\sum_{i=1}^{n} X_{2i} + \cdots + B_p\sum_{i=1}^{n} X_{pi} &= \sum_{i=1}^{n} Y_i \\
B_0\sum_{i=1}^{n} X_{1i} + B_1\sum_{i=1}^{n} {X_{1i}}^2 + B_2\sum_{i=1}^{n} X_{1i}X_{2i} + \cdots + B_p\sum_{i=1}^{n} X_{1i}X_{pi} &= \sum_{i=1}^{n} X_{1i}Y_i \\
B_0\sum_{i=1}^{n} X_{2i} + B_1\sum_{i=1}^{n} X_{2i}X_{1i} + B_2\sum_{i=1}^{n} {X_{2i}}^2 + \cdots + B_p\sum_{i=1}^{n} X_{2i}X_{pi} &= \sum_{i=1}^{n} X_{2i}Y_i \\
\vdots \quad + \quad \vdots \quad + \quad \vdots \quad + \quad \cdots \quad + \quad \vdots \quad &= \quad \vdots \\
B_0\sum_{i=1}^{n} X_{pi} + B_1\sum_{i=1}^{n} X_{pi}X_{1i} + B_2\sum_{i=1}^{n} X_{pi}X_{2i} + \cdots + B_p\sum_{i=1}^{n} {X_{pi}}^2 &= \sum_{i=1}^{n} X_{pi}Y_i
\end{aligned}
\qquad \text{.....(2.10)}
$$

该方程即为 $p$+1 正规方程，能够通过联立的方法对该方程进行求解。所求解解为最小二乘估计，依此为 $\hat{B}_0, \hat{B}_1, \cdots, \hat{B}_p$ 。

处于方便的原因，我们重新构建 2.4 中的模型，集中自变量 $X_1$,$X_2$,$X_3$,...,$X_p$，具体来讲就是根据不同变量的平均值来获得差异：

$$
\begin{aligned}
Y &= B_0 + B_1\bar{X}_1 + \cdots + B_p\bar{X}_p + B_1(X_1 - \bar{X}_1) + \cdots + B_p\left(X_p - \bar{X}_p\right) + e_i \\
&= B_0'X_0 + B_1(X_1 - \bar{X}_1) + \cdots + B_p\left(X_p - \bar{X}_p\right) + e_i
\end{aligned}
\qquad \text{.....(2.11)}
$$

从中可以得到，$B_0' = B_0 + B_1\bar{X}_1 + \cdots + B_p\bar{X}_p$，$\bar{X}_1, \cdots, \bar{X}_p$为 $p$ 个独立/回归变量的均值。基于上述的这些条件，正规方程又可以转换为如下的形式：

$$
\frac{\partial E}{\partial B_j} = -2\sum_{i=1}^{n}\left(Y_i - B_0'X_{0i} - B_1(X_{1i} - \bar{X}_1) - \cdots - B_p\left(X_{pi} - \bar{X}_p\right)\right)\left(X_{ji} - \bar{X}_j\right) = 0,
$$

$$
j = 0, \cdots, p \qquad \text{.....(2.12)}
$$

此时因该意识到，对于所有的 $i$ 有 $X_{0i}$=1,且回归系数 $B_1$，$B_2$,...,$B_p$ 仍然保持不变，而截距则由 $B'_0$。一旦计算得到了所有回归系数的估计值，我们即可通过如下式子求得 $\hat{B}_0$：

$$
\hat{B}_0 = \hat{B}_0' - \hat{B}_1\bar{X}_1 - \cdots - \hat{B}_p\bar{X}_p \qquad \text{.....(2.12)}
$$

## 2.3 流形学习

作为机器学习以及数据挖掘等领域的重要问题之一，如何在海量的数据中进行特征的选择以及抽取，成为了科研、工业界内的广泛关注。随着信息革命对科技

所带来的发展，人们在生产生活中所获得的信息呈爆发式的增长，例如在日常浏览网页中留下来的数据信息、金融交易活动数据等等。这一类的数据往往有着数据量较庞大、高维度、高度冗余、增长速度较快的问题。譬如，在我们日常接触的多媒体流中包含着视频、音频等复杂的内容，这些数据常常共同所要传达内容的一部分甚至表达相同的意义。在这样的背景下，研究如和高效地对这些海量数据进行分析处理，并抽取出其中有效的信息，成为了一个热门的方向。以以往普遍的经验拉坎，在对大量的数据单纯进行传统的机器学习方法进行训练时，往往会面临许多不可避免的阻碍。首先，由于训练时输入的维数较为庞大，容易导致出现“维数灾难”的问题，继而使得要对算法进行更频繁的参数优化，且过于复杂的模型会产生难以承受的时间消耗成本，以至于整个算法都无法进行成功的学习过程。譬如在处理 256*256 的图像时，若采用神经网络算法，则此时有 65536 个输入结点，算法的计算代价将大大的提高。从另一个方面来看，高维的样本往往含有很多的冗余信息以及大量的噪音，且不同维度之间的变量也并不是完全的相互独立的，换而言之，高维的数据并不一定能更好、更全面的刻画事物的本质，此时若直接采用高维数据做为训练集，常常无法得到一个于其的优良效果。所以，在面对高维数据时，常常要求先对数据进行预处理，即对数据的特征进行选择或抽取。

特征选择意味着从初始数据样本中，分理处那些表示数据潜在关键信息的属性，忽略与描绘实际没有意义的噪声、冗余属性的过程。依照特征选择的不同方法，可以将其分为随机式、启发式等等类型。而特征抽取则不单单是对不同维度的选择，而是通过对原始数据中的不同维度进行一定的组合、变换等操作，从而产生一个新的特征，然后利用这个新产生的特征去反映数据本质。同时，降噪和除去冗余信息的目的也在这个过程中达到了。因为特征抽取的过程中往往同时会导致维数的降低，故亦称之为“降维”。利用特征抽取后的数据，因其能更好的体现数据的本质在回归和分类等问题中神经网络等算法往往可以获得更好的效果。根据不同的降维方式，可以将数据简化方式分为线性以及非线性两个大类。具体的定义如下所示：

我们假设从 $D$ 维空间中对样本$\{x_1, x_2, x_3, x_4\}$进行采样，则 F 映射可以被定义为：

$$F: X \to Y, x_i \to F(x_i) = y_i \quad x_i \in R^D, y_i \in R^d, d \ll D \qquad (2.13)$$

此时我们则可以称 $X$ 到 $Y$ 为数据集的降维 $F$,相应的，其逆操作 $F^{-1}$ 即嵌入映射，定义如下：

$$F^{-1}: Y \to X,\ y_i = F^{-1}(y_i) = x_i \quad (2.14)$$

此时，如果映射 F 是线性的，则称其为线性降维；反之则定义其为非线性降维。

### 2.3.1 非线性降维

非线性降维算法总的来说主要涵盖了以下集中算法：神经网络算法、核方法，以及近些年新兴起来的流形学习等等类型，每一种类型的方法包括若干种相关算法，下文将进行大致介绍。

神经网络：采用神经网络进行数据降维的思路主要是由 Kohonen 提出的一个自组织特征映射发展而来(Self-Organizing Feature Map, 下文称 SOM)，由代表初始数据的输入部分以及表示输出低维数据集构成。在对数据进行训练的时候，输入层模拟生物的神经元对自然界输入信息的接受，同时输出层模拟神经元对于相应刺激的处理以及反馈。SOM 利用样本数据中相邻的一些节点，使得其彼此之间进行相互竞争以及训练，继而能维持其原本数据对应的拓扑结构，从而达到了对数据进行降维的一个目的。然而，不可避免的时，SOM 同时也兼具了传统神经网络常常需要面对过拟合以及容易陷入局部极值困境的特点。

利用核方法进行降维：核方法的采用使得传统的线性降维方法与非线性降维方法产生了关联，将若干基于线性数据样本的降维方法推广到了非线性空间，产生了一套根据非线性进行数据降维的处理方法。现如今，该方向称为非线性降维的一个热门方向。从本质上来收，核方法即为通过将初始空间中不可以线性分离的样本通过一个核函数映射到一个高维（常为无穷）的一个再生希尔伯特空间之中，此时因为数据在高维空检相对更为稀疏，从而可以对其进行线性分离。该类降维方法主要涵盖了 Scholkopf 等人提出的核主成分分析等等。然而，在采用该类降维算法时，常常需要面临因较高维数而产生的更高计算时间成本，以及对核函数如何选择缺乏系统的理论知道的困境。

流行学习：

如果想用精准的数学公式定义流形是一件相对较为复杂的事情，因而本文在此给出一个相对简单的语言描述。我们可以想像如下的一种情况：在地球这一个球体（假设地球为一个完美的圆）上，当我们从任意一个点前往另一个地点，容易推断得出，此时的路径一定是一条弧线。然而因为地球相对于我们来说非常的大，所以在在局部微观的角度看来，我们会认为自己所处的空间是一个平坦的空间。

据此推广开来，我们也可以将拓扑流形当成是一种拓扑空间，对于这一类拓扑空间，当从其局部的角度来观测，其往往是平坦、无起伏的，如同普通的欧式空间所体现出来的特征，同时其维持了固有的拓扑结构。从某种角度来看，我们可以看作将众多低维的拓扑流形片段进行拼凑，从而得到一个高维的连续空间。我们假定这些流形片段都是光滑的，对其进行组合即可达到完成流形学习问题。利用该思想进行降维的过程可如下图所示：A 列表示原始的数据空间，B 图则为对该数据空间进行离散采样，C 点则为利用流形学习进行降维后的结果。

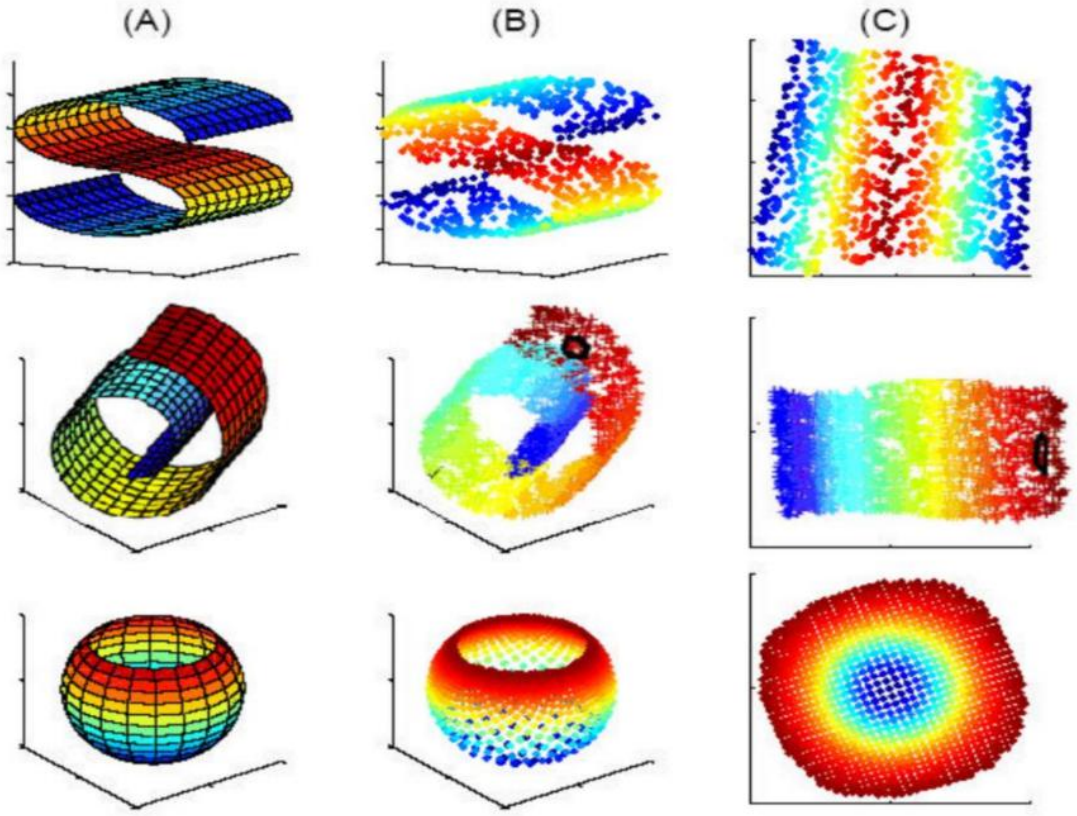


**图 2-10　流形降维流程**

现在我们给出相应的数学定义：

关于流形：

通俗的来讲，这个定义可以表示为，对于假设数据样本来说，其源于一个潜在的低维流形结构上，流形学习的目标是对高维度的数据进行学习从而计算出可能的低维空间结构，并构造出其相应的一个函数关系，从而使得数据维度能够降低[17]。

自从 2000 年 Tenenbaum 以及 Roweios 等研究者提出了等距离映射、局部线性嵌入这两种流形学习的新算法，众多根据不同思想发展而来的流形学习算法在之后的几年内被提出，下面将对其中的几种进行概述：

等距离特征映射(ISOMAP)[17]：此算法为 MDS 算法改进而来，在该方法中，初始时需要创建不同个体点彼此之间的测地距离 D：

$$D=\begin{bmatrix} d_{11} & d_{12} & \dots & d_{1N} \\ d_{21} & d_{22} & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \dots & d_{NN} \end{bmatrix}, \qquad d_{ij}=\begin{cases} \sqrt{\sum_{k=1}^{d}\left(x_{ik}-x_{jk}\right)^2}, if\, d_{ij}^{R}<\epsilon \\ \min\limits_{l\notin\{i,j\}}\{d_{il}+d_{lj}\}, otherwise \end{cases} \quad \text{.....(2.21)}$$

经过降维后，可以将低维度坐标求解问题转化为一个最小化重构误差问题：

$$\min E=\|\tau(H)-\tau(H_Y)\|_{L^2} \quad \text{.....(2.22)}$$

在这其中，$H_Y=\{d_y(i,j)=\|y_i-y_j\|\}$，$\tau(H)=-USU/2$，$U=\delta-1/n$，$S=D^TD$。若 $Y$ 取值为$\tau(H)$最大的几个特征值相应的特征向量是，上述式子成立。

局部线性嵌入(LLE)[18]：此方法的思路是假设数据集中所有的点较为聚集地分布在一块适当大小局部领域，其分布服从一个线性的结构，故在这一个局部结构之中，每一个样本点皆能够通过临近点来表示。我们可以称线性表示所有临近点的矩阵为权值矩阵。而 LLE 的目的是在降维的过程之中保持其局部权值不变，而根据重排领域重叠信息可以得到一个全局上的拓扑排序。

我们定义数据集合$X=\{x_1,x_2,\dots,x_n\}\in R^{D*n}$，则可得到一下式子：

$$KNN(x_i)=\{x_{\Phi(j)},j=1,\dots,k\},\quad d_{i\Phi(j)}=|x_i-x_{\Phi(j)}|<d_{il},\\ l\notin KNN(x_i),x_{\Phi(j)}\in X \quad \text{.....(2.23)}$$

设定局部领域样本重构误差，并使得其最小化，可以计算得到重构权值矩阵 W：

$$e(W)=\sum_{i=1}^{n}\left\|x_i-\sum_{j=1}^{n}w_{ij}x_j\right\|^2,$$

$$s.t.\sum_{j=1}^{n}w_{ij}=1\ and\ if\ x_j\notin KNN(x_i)\ then\ w_{ij}=0 \quad \text{.....(2.24)}$$

通过利用权值矩阵进行求解得到流形嵌入坐标 $Y$ 此时可以被简化为下述极小值问题：

$$\Phi(Y)=\sum_{i=1}^{n}\left\|y_i-\sum_{j=1}^{n}w_{ij}y_j\right\|^2=\|YI-YW^T\|^2=\|Y(I-W^T)\|^2=tr(YMY^T)\\ M=(I-W)^T(I-W),\\ s.t.\sum_i y_i=0,\sum_i y_iy_i^T=I \quad \text{.....(2.25)}$$

此时 $Y$ 值应为特征值最小是所对应的特征向量。

t-SNE：是一种用于数据降维的非线性降维算法，常常被用于将原始数据重高维度空间映射到二维或者三维空间上，从而能够进行可视化。该算法基于 SNE 改

进而来，尽管其已经能做到映射到现实物理世界中，但若想要对该算法进行优化则有一定的难度。同时，不可以忽视的是，该使用 SNE 算法时常常需要面对拥挤问题(crowding problem)。因此，为了满足处理实际数据的需求，Hinton 等人在原有 SNE 算法的基础之上，提出了 t-SNE 算法。主要做出了如下改进：

(1) 使用了对称 SNE(Symmetric SNE)，从而达到了对梯度公式进行简化的目的；

(2) 在低维空间下，摒弃了高斯分布，转而使用 t 分布来代表任意两点彼此之间的相似度。

上述的拥挤问题指的是各个不同簇混杂在一起，从而导致无法区分。考虑这样的一种情况，当高维度的数据通过降维到 10 维空间时，仍然可以有达到一个较为清晰的表达效果。然而，此时若进行进一步的降维，比如把维数降低到两维时，此时则没有办法得到一个可信映射。举个例子：原始数据中有 11 个样本点，假定此时对该数据集进行降维，当降维到 10 维时，该 11 个点彼此之间的距离彼此相等。此时若进一步进行降维，如降维到三维空间时，最多只能有四个点彼此之间距离相同，因而则无法满足条件。我们可以更进一步的进行推论，以一个数据点 $x_i$ 为中心，半径为 r 的 m 维球体，其体积的变化是基于 $r^m$ 的。具体的变化如下图：

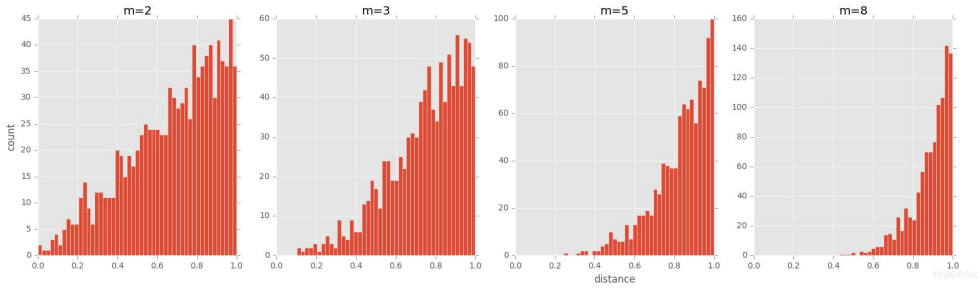


图 2-11　距离随着维度变化

从上面的信息可以得出，随着数据维数的变大，绝大部分的个体实际上都会聚集在 m 维球的表面，分布十分的不均匀。此时若直接将该距离关系保留进行降维，必然会导致产生拥挤的问题。

通常来说，如果要对 UNI-SNE 进行优化，首先是先让ρ为 0，然后再采用标准的 SNE 优化，接着利用模拟退火的方法，逐渐地增加ρ。因为初始ρ不为 0，如果想要立即对该算法进行优化是不可取的。这也就意味着，在之后的迭代过程之中就很难再将两个初始是就被分隔很远的聚类中心拉近了。

### 2.3.2 利用流形学习进行降维所面对的问题

通过采用基于流形学习的算法进行降维即使带来了许多好处，但是不可否认的是，该类算法在实际的实践过程之中，仍然存在着一定的问题。现在我们对所遇到的问题作出如下的一些总结：

往往有较高的计算复杂度：基于对于上述几种流形学习算法的原理，进行原理的深入分析，我们可以得知，所谓的流形算法，事实上是组合现有的一些经典算法，或者是对其加以改进。这样就导致了这一类的算法往往有交复杂的工作步骤，且在迭代的过程中，每一个流程往往也都较为繁杂。故而，这一类算法在遇到输入的样本数据集较为庞杂时，往往会导致运算所消耗的时间成本较高。因而是目前制约流形算法在现实世界中进行应用的一道障碍。

如何准确的估计本征维数：在运用流形学习算法的过程之中，需要实现设定一个降维维数，而如何对于估计具体多少维的低维流形能够表达高维空间种的原始数据集，成为了一个关键的问题。考虑以下的两种情况：当我们对低维流形的维度数估计过高，则可能导致生成的低维流形仍然存在一定的数据冗余，并没有完全发挥本身算法的潜力，同时产生一些额外的计算时间成本；与此相反，如若对于低维流形的维度数估计过低，则可能导致在降维过程中导致一些数据损失，从而无法构建流形的结构，样本中的个体之间的相互关系与降维之前不再一致，严重的情况下，很有可能会导致整个算法得出一个完全失败的结果。

无法进行增量学习：目前流形学习降维的目标是针对已有的样本数据，而在工程实践之中，往往会时不时地产生新的数据，而此时流形学习算法没有办法直接将新产生的数据直接映射到已有的低维流形上，而是必须重新以整体的形式进行计算，这就导致在面对新的数据时，往往要产生很多的额外开销。究其原因，是因为，目前使用较广泛的一些流形算法生成的可近似看成保持了原有数据样本分布的低维流形，是基于原始的高维样本集合的拓扑结构进行计算的，故而该算法的迭代过程中进行的计算与数据集是紧紧关联的，此时添加一个新的数据，很有可能完全破坏原有的数据拓扑结构。同时，考虑到流形学习属于非线性降维算法，故而该算法没有办法提供一个表示映射的相应公式。综上所述，对于一个已经生成的低维流形，此时若要添加一个新的数据到该低维流形中，往往要面对很大的困难。

除了上述的一些问题，流形学习中还常常会遇到例如监督学习、噪声等等方面

的问题。基于上述的问题，如何对流形学习进行提高以及改进，俨然已经称为了相关领域的研究热点方向。

## 2.4 聚类算法

聚类算法作为模式识别的重要问题之一，学界对其有广泛的定义。其中一种较为简洁、能够表述聚类本质的定义如下：将具有类似性质的数据归为同一类。聚类算法可以有如下数学定义进行正式表达：假定$X \in R^{m\times n}$表示 n 维上的不同数据样本$x_i, i = 1, \dots, m$集合，则此时聚类算法的目标即是把 $X$ 中的数据归并到类别$C_k, k = 1, \dots, K$之中，直至令所有被归并于同一组的数据之间的相似度要大于不同小组之间数据的相似度。此时我们即可以称每一小组 $C_k$ 为一个类。最终算法的结果，即为达到通过映射 $X \rightarrow C$ 从而使得数据 $X_i$ 可以映射到 $C_k$ 的结果。应当主要到，上述对于聚类结果类别的数量并不一定是人为事先设定的，也有可能是算法在迭代过程中自动得出的。从本质上来看，可以将聚类问题看作为一个求非线性全局最优解问题，这一类问题往往都是 NP 难问题。

聚类算法介绍：有许多种不同的方式，对聚类问题进行阐述，故而得到的聚类结果，以及如何对其进行诠释，常常于如果规划聚类问题所相互关联。考虑一下的几种情况：譬如说，假定可以唯一区分每一个类别，则此时样本集中的每一个数据一定属于某一个特定的类别；在相反的情况下，即对于不同类的划分有可能产生重叠，即样本中的某一个数据有可能同时属于几个不同的类别。又或者说进行聚类的方法是基于概率的，则每个样本点对应不同的类别有不同的概率大小。因此，如何对聚类问题进行规划，将会产生完全不同的一个迭代结果。虽然有不同发来对聚类算法来分类，本文在此仅考虑通过以分离数据的方式为标准，对算法的种类进行区分：非参数化聚类算法，以及参数化聚类算法。

参数化方法：该类算法往往通常预设所有样本点分布在一个相对固定的拓扑结构，具体进行分类可包括概率模型、重构模型两类。

概率模型：该类算法假定输入的样本 $x_1, x_2, \dots, x_m$ 是分别在 $K$ 个未知分布 $E_1, E_2, \dots, E_3$ 之上的一个观测，此时，若定义 $x_k$ 相对应于 $E_i$ 的概率密度分布为$f_i(x_k|\Theta)$，此时$\Theta$是一个未知的变量。同时，定义 $T_k^i$，表示为 $x_k$ 隶属于 $E_i$ 的概率大小值。在通常的情况下，我们往往会假定一个点对应一个相应的分布，同时需要满足相关

的约束条件$\sum_{i=1}^{K} \mathrm{T}_k^i = 1$，从而能够找到问题所对应需要的两个变量$\Theta$以及 T 的问题，可以被等价于一个求解最大似然函数的问题：

$$L(\Theta, \mathrm{T}) = \sum_{r=1}^{n} \ln(\sum_{i=1}^{K} \mathrm{T}_k^i f_i(x_k|\Theta)) \quad \text{.....(2.26)}$$

该方法需要面临的一个阻碍是，当我们尝试允许$\Theta$值进行任意的增减，那么我们在解决一些大型问题的时候，如果要试图找到一个全局之上的最优解，那么很有可能会耗费额外的一些时间和空间。通常地来讲，在对算法进行初始化的时候，往往有可能将算法引入局部最小值的困境之中。故我们一般上认为，该类算法在处理噪音问题方面仍然有很大的改进空间。

重构算法：经典的一些重构算法往往尝试让相应的代价函数进行最小化，而该类算法下的不同算法之间并没有什么本质上的区别，区分他们的只不过是不同算法所被采用的模型，和它相对应的代价函数。很长一段时间内，重构算法中最富盛名的当属 K-means 算法，该算法能够根据预先设定的聚类组数，将初始样本划分到不同的组之中，从而达到一个聚类的效果。当利用其他距离公式，来替换原本的欧几里得距离公式时，即为 K-means 算法的变体。

虽然该算法从流程上来讲并不是十分的复杂，对其进行实现也相对较为简单，然而从本质上来说，该算法在一定程度上仍然只是一种贪心算法。故而我们可以得出这样的一个结论，该算法将注定面临陷入局部最优解的困境之中。我们为了避免这样的困境，往往需要采用一些特定的步骤进行初始化，然而这也同时导致了初始化样本的选择对于算法最终的结果有较为深刻的影响。因此，譬如模拟退火算法等一类随机性方法常常被采用，从而改进该类聚类算法。

除了 K-means 意外，另外有一类算法，被称之为拓扑图聚类。该类算法与其他传统的聚类方法有一个明显的不同之处，即它不再是初始数据个体被分配到局部空间中的中心，而是将样本点经过非线性映射处理，从而转换到一个与局部空间所相对应的高维空间之中。由于基于核函数，我们可以进行一个内积运算，可以到到一个完成低维到高维的映射过程，因此这类算法一般无须一个非常明确的映射。

K-means 聚类算法：事实上，该算法的流程非常的通俗易懂，大致上被定义为如下流程：首先，算法获得一个输入的初始集合，接着依此地把所有的样本点分配到任意一个类之中，则此时所有数据点的平均值即被当作为各个类的中心点位

置。将上述的过程不断地进行重复，直到算法最终收敛到一个固定的结果。该算法所遵循的一条规则是，在每一次的迭代过程之中，样本点将选择一个在欧式距离上中心点离自己最近的一个类之中。可以看出，该方法虽然流程上较为简单，同时容易被实现，但是不可以忽略的问题是，该算法在每一轮迭代之中，都需要更新点与类中心的距离，故该算法相对而言效率较为低下；另外一个问题即为上述的容易陷入局部最优解的问题，解决方案是需要采用特定流程选择初始样本，因此该问题从某种角度上来说得到了缓解。

非参数化算法：该类算法中最具代表性的莫过于以下的两种：凝聚以及分裂这两类分层聚类算法。这两种算法有一定的相似之处，都是基于当前初始数据中个体之间的不相似性来进行计算的。通过利用不同个体之间的相似性，凝聚方法将一定的个体进行聚合从而能有达到分类的效果；而在另一方面，分裂方法则是基于不同个体之间的相似度来将一些类中的个体分离。在下文中，我们将着重描述前一类算法，当然其思想可以被发散应用于后者。在迭代过程之中，当需要考虑如何描述当前需要将哪一些类合并到一起时，可以尝试采用树形结构。具体而言，即是在尝试在预定的范围里选择一个类来连接时，可以通过破坏树形结构来达成目的。而在如何确定聚类簇数方面，则可以利用基于聚合阈值相关的特定函数来达成要求。通俗来讲，即最终的聚类簇数将随着其阈值的改变而发生变化：当阈值较小，例如等于零的时候，每一一个样本各为一类，即样本个数即为最终的聚类簇数；相反的，当阈值较大的时候，则很有可能所有的样本同属一类，此时的聚类簇数即为 1。应当注意到，正如其名字所表达的那样，该类型的结构实际上也是一种树，因此从对于数据的增加以及存储的角度来看，该类结构相对较为高效。然而不可否认的时，该类算法仍然有些不可忽略的劣势，即上一代的聚类结果与当前迭代结果有着较强的联系，同时数据之间的从属关系禁止被调换；当然，该类算法的吸引力也是显而易见的，即该类方法与别的算法不同，并不规定在数据初始化时预定相关的先验分布，在这一点上，他特别体现在不用采用基于欧几里得距离的数据，而是基于一个相似度矩阵，该矩阵由数据点彼此之间的预定距离所构成。应当注意到，当要进行聚类操作的数据在属性、大小、密集度上面存在较为剧烈的变化、或有较大的冗余时，该算法的结果往往不太尽如人意。

下面，我们将引入一种近期被广泛追捧的聚类算法，它被称为亲和传播聚类算法（在下文中我们称其为 AP 聚类）。在后续构建代理模型的工作中，我们将运用到该聚类算法。

亲和传播聚类算法(AP):

相较于上述的 K-means，本算法最大的一个特点是其并不要求在进行迭代之前，手动设定相应的聚簇数，与此同时，该方法最后完成迭代所输出的分类结果往往较为固定，因此，在面对需要处理中等或者较小型体量的样本时，使用该类算法常常能获得一个不错的效果。该算法在科学杂志上被 Frey[19]等人提出。

第一步，我们首先要选定一种恰当的度量方式，譬如说最广泛采用的欧几里得距离等等。根据选定好的度量方式，根据所得距离构建出一个相似度矩阵 $S=[s(i,k)]_{n\times n}$，此时需要先对矩阵中对角线上的元素$s(k,k)$进行一个设定，该参数被设定的愈大，则所对应的个体本选择为聚簇中心的概率也就越高。在进行初始化时，同时还需要设定参数 preference，该参数也会影响最终聚簇中心的数量。另外，还有几个比较关键的矩阵，依此为吸引矩阵 $R=[r(i,k)]_{n\times n}$，以及归属矩阵 $A=[a(i,k)]_{n\times n}$。事实上，该算法的运行流程就是上述这两个矩阵不断地进行更新的一个过程，具体的步骤如下：

$$r(i,k)\leftarrow s(i,k)-\max_{k's\cdot t\cdot k'\neq k}\{a(i,k')+s(i,k')\}$$

$$a(i,k)\leftarrow \min\left\{0,r(k,k)+\sum_{i's\cdot t\cdot i'\notin\{i,k\}} max\{0,r(i',k)\}\right\}$$

$$a(i,k)\leftarrow \sum_{i's\cdot t\cdot i'\neq k} max\{0,r(i',k)\} \qquad .....(2.27)$$

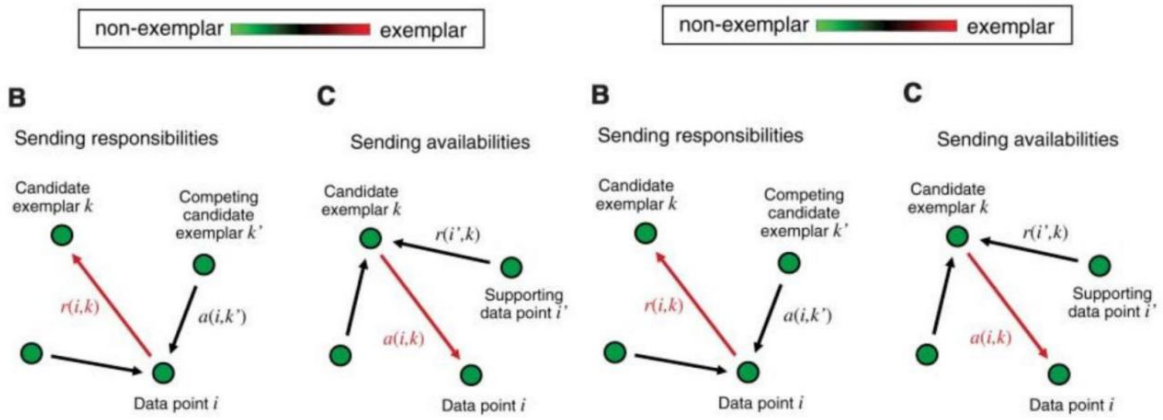


图 2-12　AP 算法的运行流程

该算法在很多工程实践中运用的都十分的广泛，效果十分良好。因而本文中的聚类过程将采用 AP 聚类进行操作。

# 第 3 章　代理模型构建

本小节将基于以下几个部分进行阐述：无降维无聚类代理模型、带有降维无聚类的代理模型，同时带有降维以及聚类算法的代理模型。

## 3.1　基于多元线性回归的代理模型的迭代过程

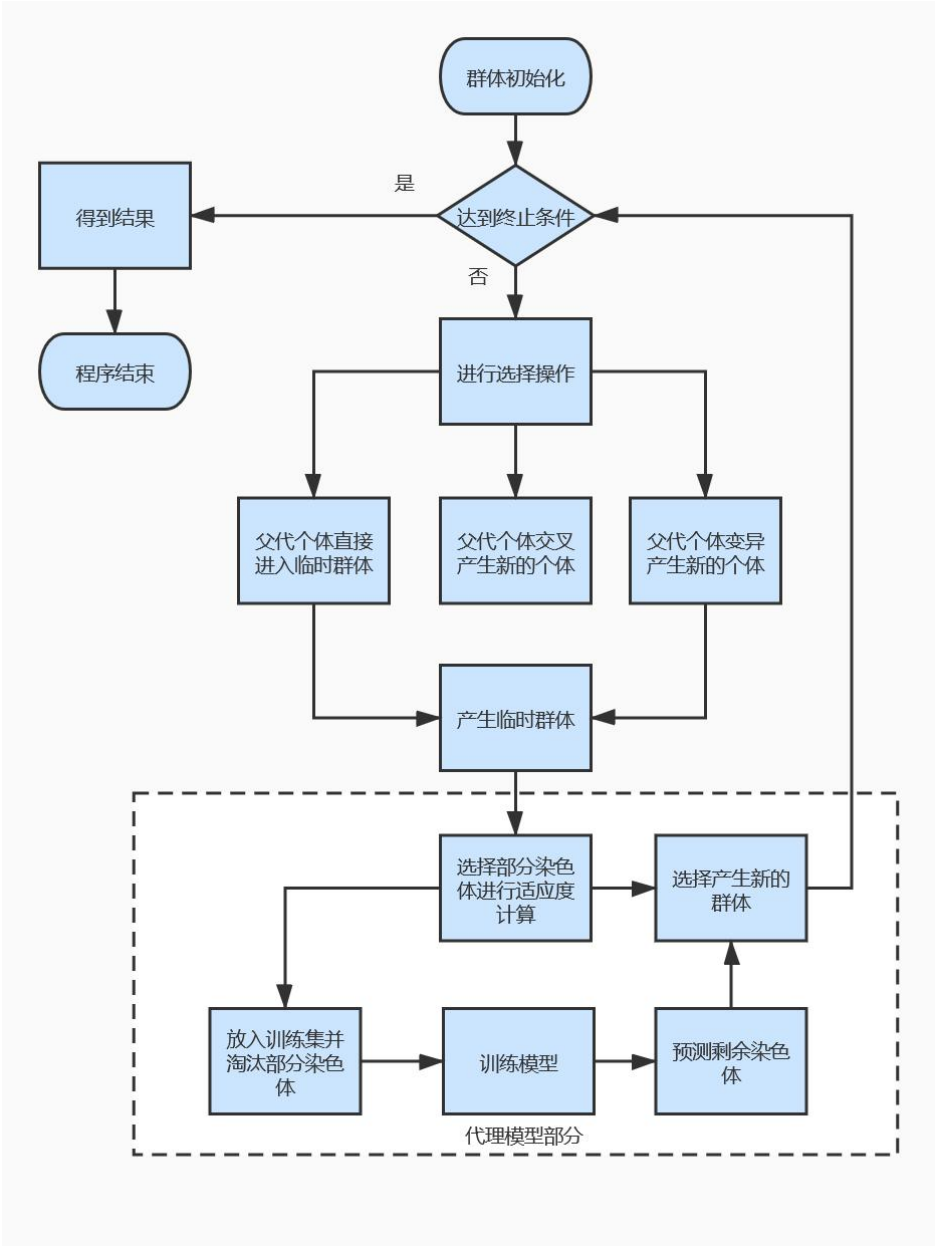


图 3-1　利用代理模型进行计算

在本文所做的工作中，将传统的遗传算法加以改进，具体的工作流程大致如下：

一开始，需要设定好各个参数（如种群大小、进化代数、染色体长度、基因突变概率、染色体重组概率），基于上述等参数，首先对群体中的个体进行初始化。由于一开始初始化的群体将决定初期的进化方向，因此，为了达到能够搜索到全局最优解的效果，同时考虑简便程度，我们一般使用随机的方式来产生第一代种群。完成上述的工作之后，即可开始进入不断进化的过程。在每一轮进化中，我们基于事先所设定的基因突变概率以及染色体重组概率，对上一代的种群进行操作，产生一个临时种群。此时，该临时种群需要进行选择，从而能够进行下一代的进化。而进行选择则需要一个合理评判尺度。在经典的遗传算法之中，对个体进行评判方式就是直接调用事先定义的适应度函数，通过它来计算每个个体的适应度值，从而能够选择相对较优良的个体。但正如我们在前文中所述的那样，在某些特定的工程实践中，适应度函数往往非常复杂，频繁地对其进行调用，会产生大量额外的时间成本开销。因此，在本文中，我们选择引入该代理模型。在每一轮进化中，我们只选择一部分的个体直接调用适应度函数进行计算，同时把染色体上的基因以及最终适应度值保存进一个训练集中，然后再利用一个搭建好的预测模型对剩余未进行计算的个体进行预测，预测结果作为其对应的适应度值。最后，我们再基于之前直接计算以及预测出来的适应度值，进行选择操作，形成新一代的个体，从而能够进行下一轮的进化。在本文中，将采用多元线性回归作为预测模型中所采用的方法。其大致流程如下所示：

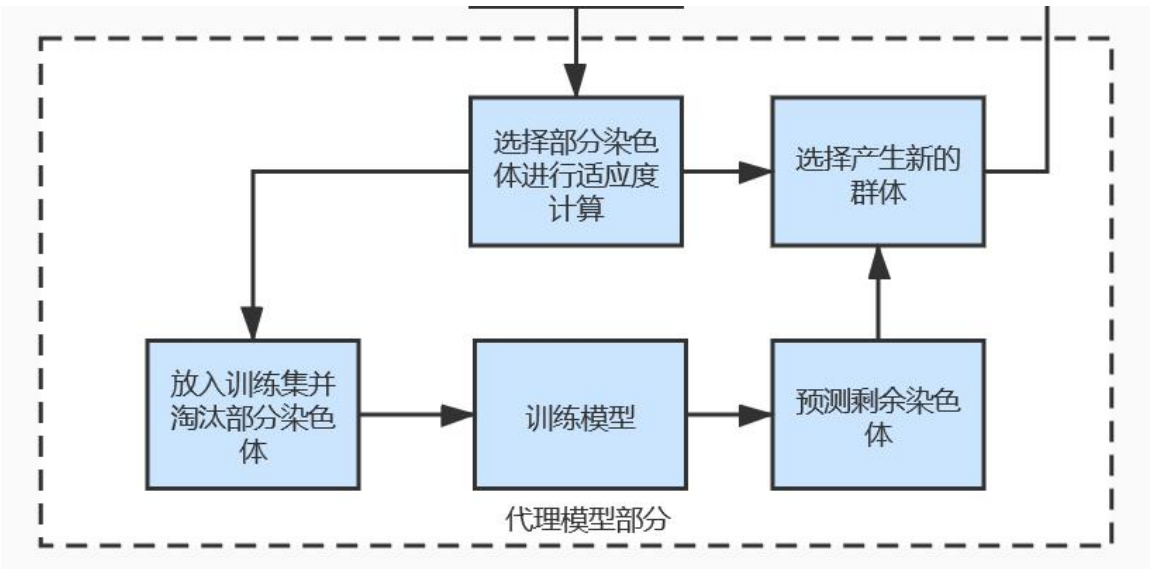


图 3-2　基于多元线性回归的代理模型

## 3.2 基于线性回归以及流形学习的代理模型的迭代过程

上述的代理模型在一定的程度上呈现出了不错的效果，然而不可否认的是，上述算法在实践中仍然会不可避免的遇到一些阻碍。无论遗传算法尝试要解决的是怎样一个具体的问题，事实上都可以将其看作为一个在尝试在高维曲面上寻找一个最优解的过程。事实上，在很多工程实际应用中，该高维面实际上很有可能是非常繁杂以及不连续的，从而没有没有办法通过特定的一个数学公式来进行操作，因此，想要利用上述带模型的多元线性回归方式，来获得一个接近原始适应度函数的预测模型其实也是比较困难的，因此往往上述代理模型在此时会遇到预测精度较低的问题。然而，根据拓扑流形的的一些概念我们可以知道，如果上述曲面可以在局部满足某些特殊的条件：如局部的拓扑结构同胚于低维流形，则我们可以采用降维的方式，将该局部的拓扑结构映射到低纬度的空间中，此时对所得到的低维流形上的重叠坐标部分进行一个替换，从而能够计算得出一个在全局上的拓扑结构。当上述过程完成之后，就能够根据所计算出的流形进行适应度的预测，从而能有在极大的减小计算代价的情况下保持一个相对较高的预测精度，因为在通常的情况下，经过降维之后的流形常常只有两到三维。具体的流程如下图所示：

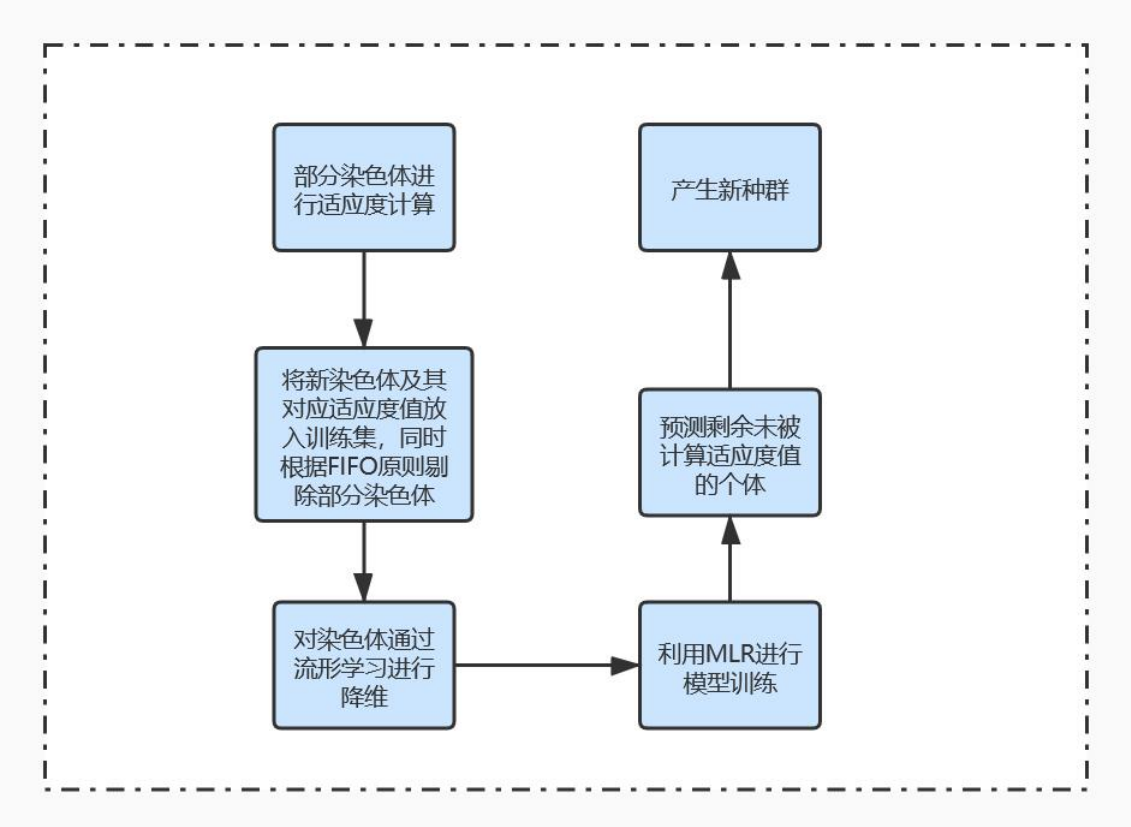


图 3-3 利用流形学习进行降维的代理模型

## 3.3 利用聚类算法的改进代理模型的迭代过程

如上述流程所示，在面都高维度的数据时，将原始数据进行流形学习，再把降维后的结果进行多元线性回归，往往可以得到一个更准确的预测效果。然而，此时应当注意到，训练集中的数据事实上随机筛选出来的，如果能选取具有代表性的个体进行多元线性回归训练，那么可能可以得到一个更加精确的预测效果。因此，在上述的基础之上，我们在代理模型中又利用了 AP 聚类，利用该聚类算法找出训练集中具有代表性的个体（即聚类中心），然后把所有代表性个体放入模型中进行训练。此时的代理模型具体如下：

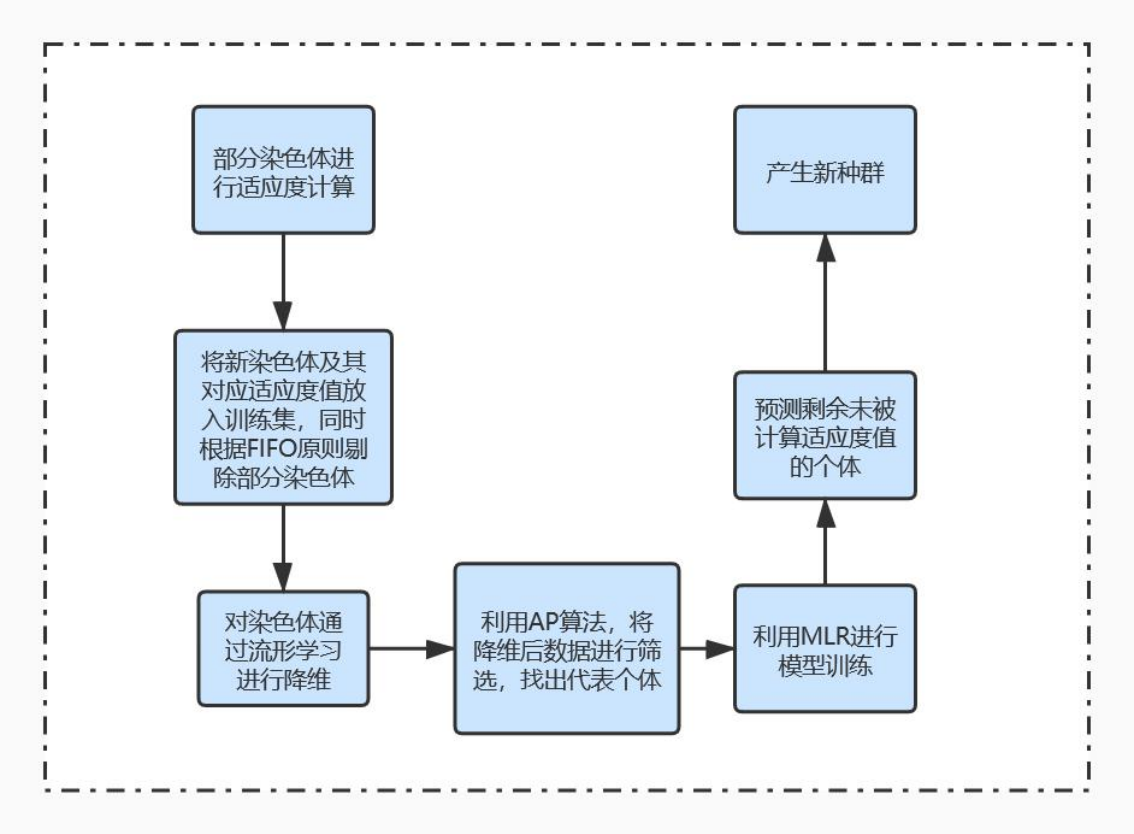


图 3-4 加入 AP 聚类的代理模型

关于训练集的维护策略，其实非常简单，采用的 FIFO 原则，即所谓的先进先出原则。从另一个角度来说，事实上从时间的维度上来讨论，上述代理模型实际上是一个根据局部来构建的一个训练集，而不是记录历史上所有染色体的一个完整训练集。之所以采用这样的一种方式，主要是有如下的一些考量：1.避免训练时间过长：随着染色体的逐渐增加，如果一直只进不出，那么到了进化后期可能所需的进化时间将会非常的大。2.保持预测进度：如果一直保持

较为久远的个体，而随着进化的不断进行，古老的个体可能会对当前的预测造成比较大的偏差。

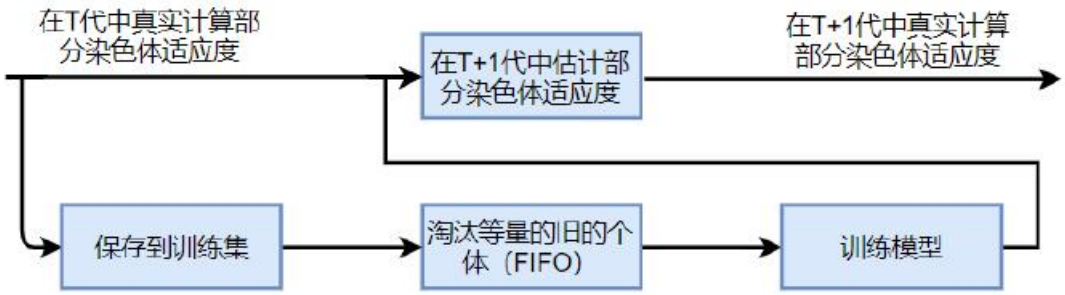


图 3-5 代理模型中训练集的维

# 第 4 章　数值实验

## 4.1　基准函数介绍

如果想要直观的评价上述改进算法的性能，拥有合适恰当的基准测试函数十分关键。在一个较为理想的情况下，测试函数应该囊括不同的一些属性，从而能有较为合理的评判算法的性能

具体的函数请参见如下表格

表 4-1　基准函数类型

| No. | Function Name | Function Formula | Domain |
|---|---|---|---|
| 1 | Step function | $f(x)=\sum_{i=1}^{n}x_i^2$ | $[-100,100]^n$ |
| 2 | Sphere function | $f(x)=\sum_{i=1}^{n}x_i^2$ | $[-100,100]^n$ |
| 3 | Schwefel function | $f(x)=418.9829\times n+\sum_{i=1}^{n}x_i\,sin(\sqrt{\lvert x_i\rvert})$ | $[-500,500]^n$ |
| 4 | Ackley function | $f(x)=-20\,exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i))+20+e$ | $[-32,32]^n$ |
| 5 | Rosenbrock Function | $f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$ | $[-2.048,2.048]^n$ |
| 6 | Griewank Function | $f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos(\frac{x_i}{\sqrt{i}})+1$ | $[-512,512]^n$ |
| 7 | Ridge Function | $f(x_1\cdots x_n)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$<br>$-64\le x_i\le 64$<br>minimum at $f(0,\cdots,0)=0$ | $[-64,64]^n$ |
| 8 | Whitley Function | $f(x_1\cdots x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{(100(x_i^2-x_j)^2+(1-x_j)^2)^2}{4000}-cos(100(x_i^2-x_j)^2+(1-x_j)^2)+1)$ | $[-10.24,10.24]^n$ |

| No. | Function Name | Function Formula | Domain |
|---|---|---|---|
| 9 | Moddouble Function | $f(x_1 \cdots x_n) = \sum_{i=1}^{n}(\sum_{j=1}^{i}(x_j - j)^2)$<br>$-10.24 \leq x_i \leq 10.24$<br>minimum at $f(1, 2, 3, \cdots, n) = 0$ | [-10.24,10.24]$^n$ |
| 10 | Quartic Function | $f(x_0 \cdots x_n) = \sum_{i=0}^{n} i x_i^4 + \text{random}[0, 1)$<br>$-1.28 \leq x_i \leq 1.28$<br>minimum at $f(0, \cdots, 0) = 0 + noise$ | [-1.28,1.28]$^n$ |
| 11 | Rastrigin Function | $f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$<br>$-5.12 \leq x_i \leq 5.12$<br>minimum at $f(0, \cdots, 0) = 0$ | [-5.12,5.12]$^n$ |

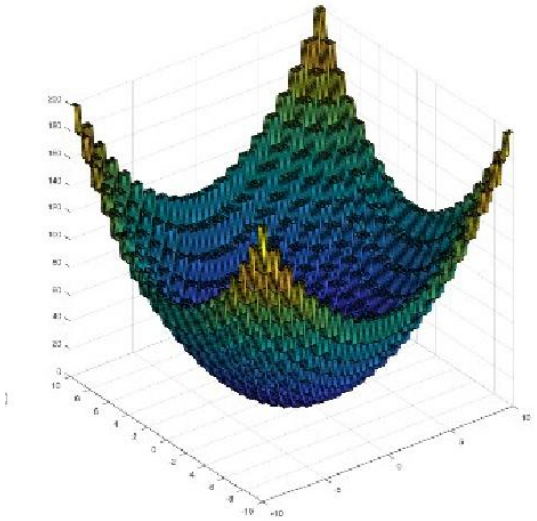

图 4-1 function 1: Step Funtion

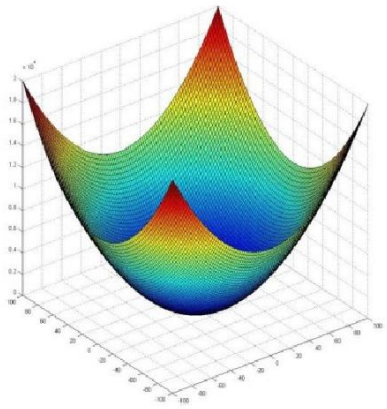

图 4-2 function 2: Sphere Funtion

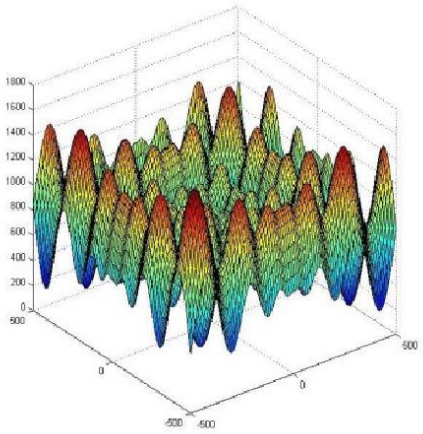


图 4-3 function 3: Schwefel Funtion

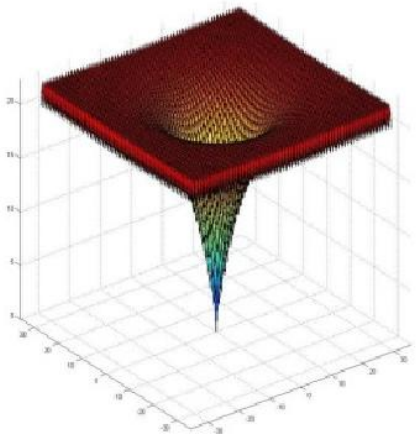

图 4-4 function 4: Ackley's Funtion

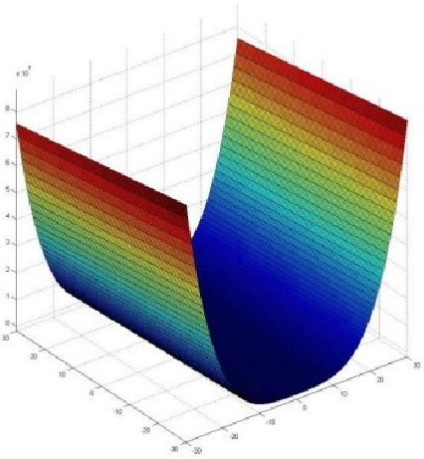

图 4-5 function 5: Rosenbrock Funtion

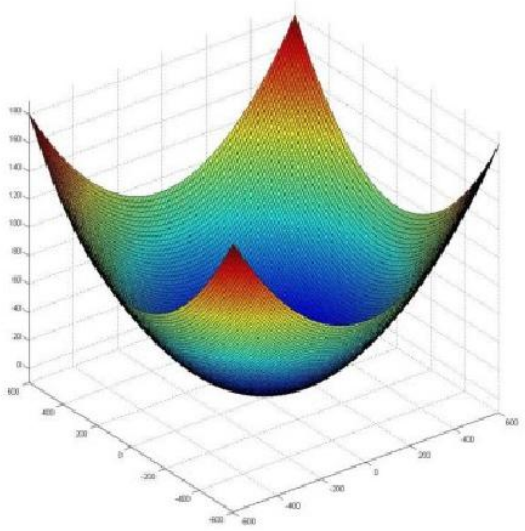

图 4-6 function 6: Griewank Funtion

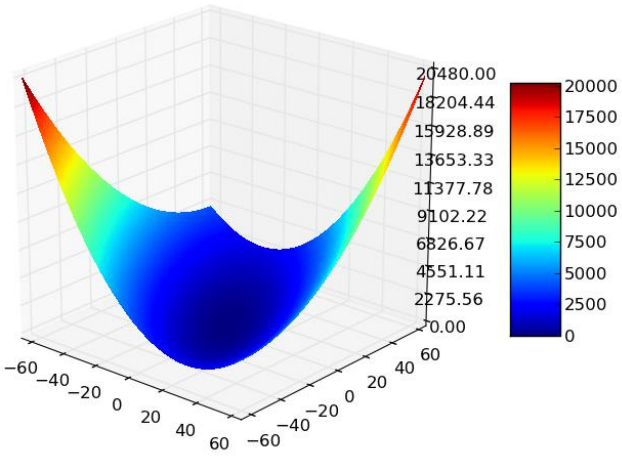


图 4-7 function 7: Ridge Funtion

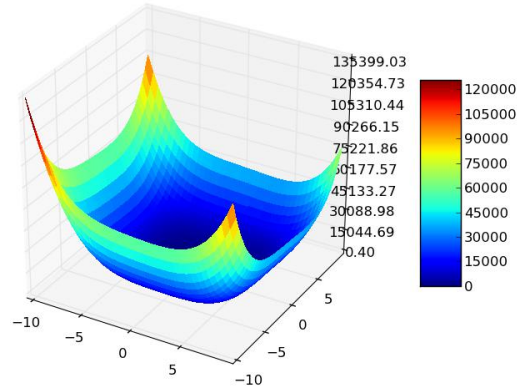


图 4-8 function 8: Whitley Funtion

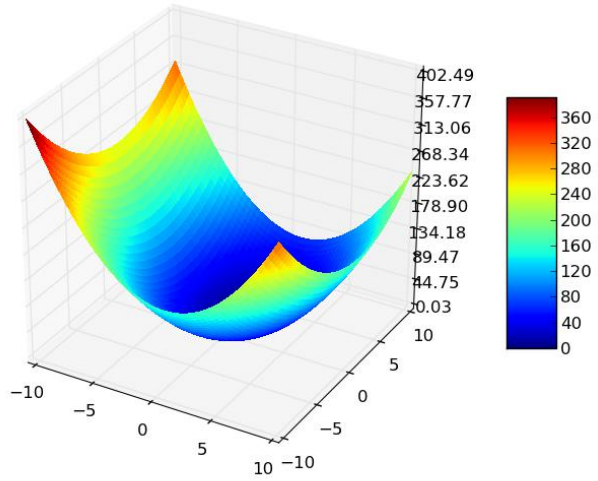


图 4-9 function 9: ModDouble Funtion

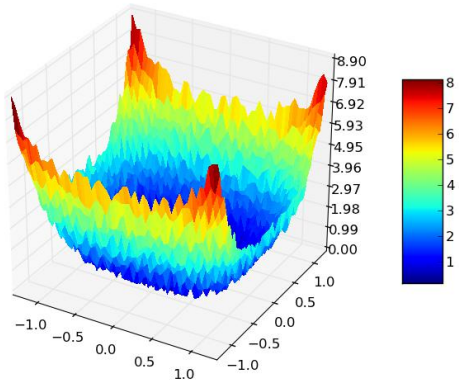


图 4-10 function 10: Quartic Funtion

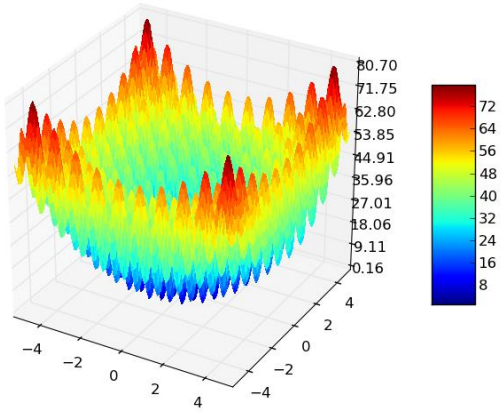


图 4-11 function 11: Rastrigin Funtion

## 4.2 实验设计

本文基于 python 环境上的遗传算法工具箱 Geatpy2 进行改进，具体的实验设计如下：我们利用前文所述的几种不同代理模型，对所有的函数进行测试。为了确定不同流形学习算法的有效性，我们将对同时对 t-SNE、LLE、Isomap 三种不同流形算法进行实验，同时为了确定 AP 聚类对于预测精度的影响，我们也将对随机抽样、AP 聚类两种筛选方式进行实验。对于不同的算法，我们设置了相同的种群大小、训练集容量以及迭代次数。在选择编码的策略上，本文选择实数编码。为了获得一个普遍性的结论，每个算法将运行十次。参数设置大致如下：

**表 4-2 参数设置**

| 类别 | 参数内容 |
| --- | --- |
| 遗传算法 | 群体大小=50；进化迭代次数=100；训练集选择概率=0.5；<br>训练集大小=200；问题维数=30；算法重复运算次数=10； |
| LLE | 降维后维度=3；eigen_solver=dense |
| t-SNE | 降维后维度=3；init=pca;method=barnes_hut;angle=0.2;<br>n_iter=1000 |
| Isomap | 降维后维度=3； |
| AP 聚类算法 | Preference 系数设置为中位数；damping 系数设置为 0.5； |

为了方便结果展示以及对不同改进算法性能的对比，下面的实验结果将分别通过不同代理模型之间的比较（即仅仅基于多元线性回归的代理模型、通过流形学习进行改进的代理模型、同时使用流形学习以及聚类方法的降维模型，其中流形方法选择了 LLE 为代表）以及采用了不同流形算法之间的比较。其中，性能较为优异的属性被加粗。

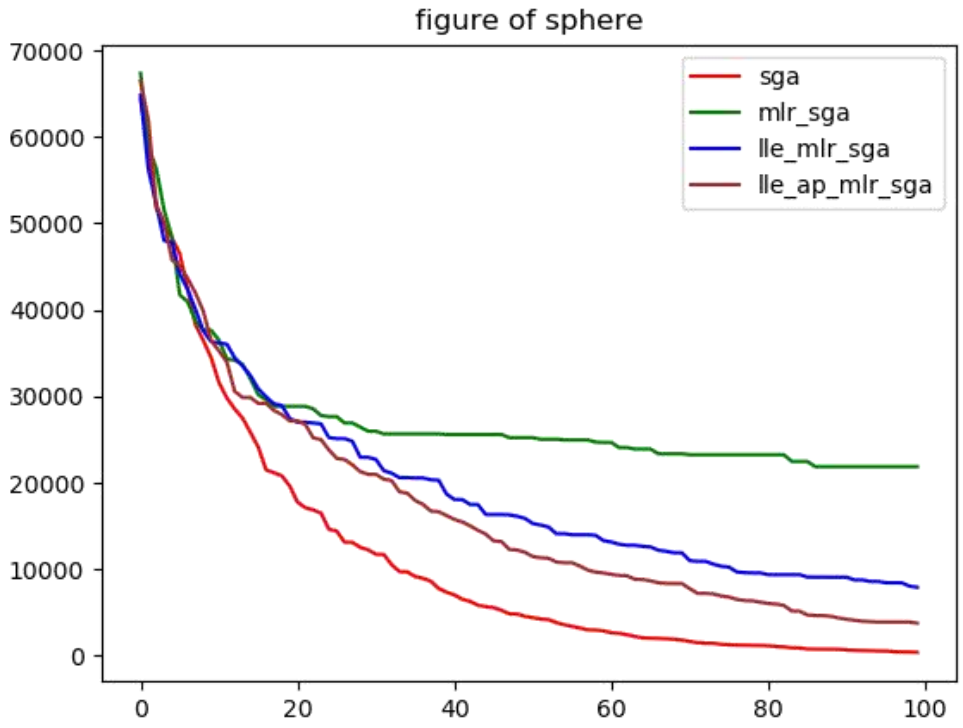


图 4-12 基于 Sphere function 的算法结果

表 4-3 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 170 | 16620 | 5050 | 2836 |
| 平均适应度 | 407.3 | 21871.3 | 7925.7 | 3772.1 |
| 适应度标准差 | 146.889 | 3360.423 | 1929.286 | **613.447** |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.165 | **0.440** | 3.862 | 9.038 |

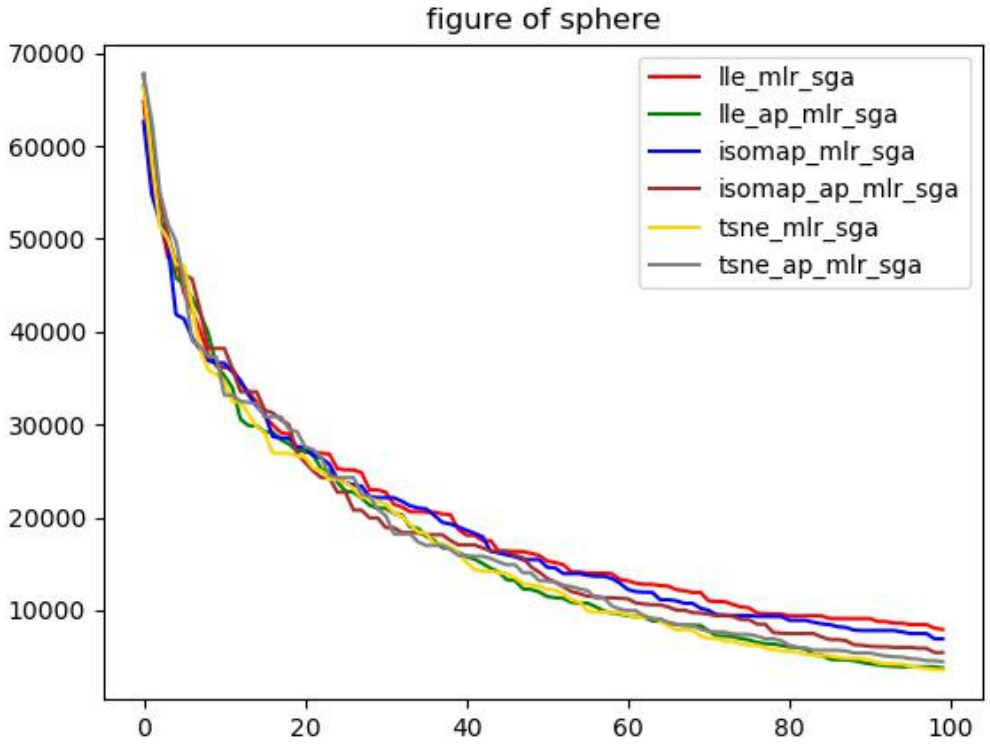


图 4-13 基于 Sphere function 的算法结果

表 4-4 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 3821 | 4388 | **2612** | 2799 |
| 平均适应度 | 6916.9 | 5448.5 | **3601.4** | 4457.6 |
| 适应度标准差 | 2057.502 | 649.773 | 984.521 | 1100.707 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.676 | 6.588 | 208.393 | 209.663 |

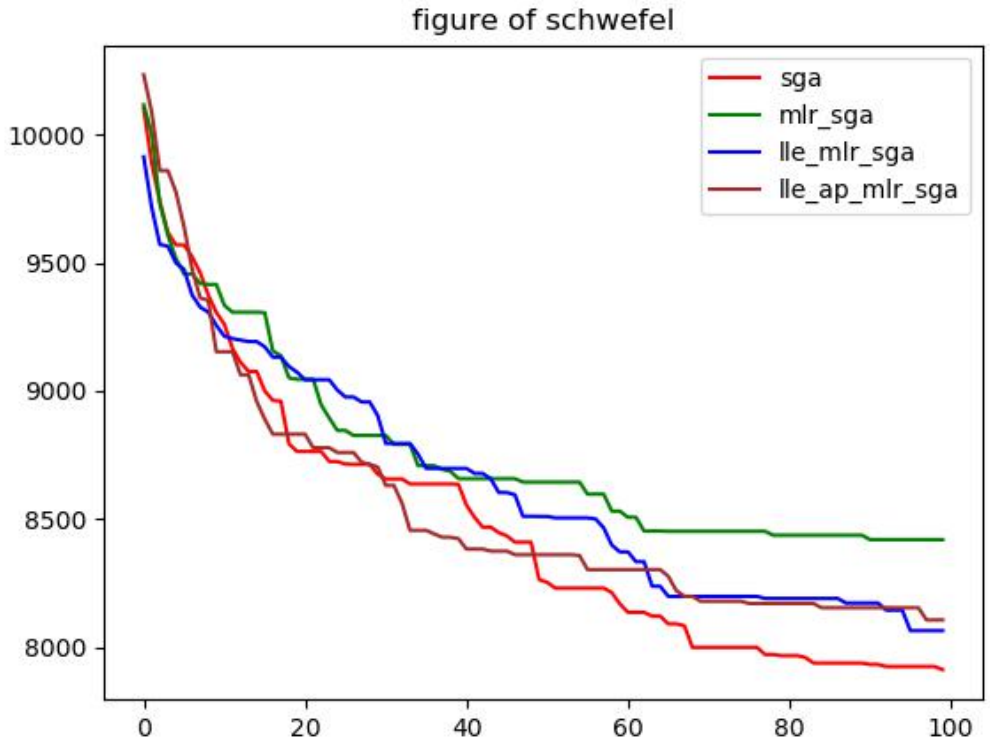


图 4-14 基于 Schwefel function 的算法结果

表 4-5 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 6957 | 7858 | **7243** | 7511 |
| 平均适应度 | 7911.2 | 8419.3 | **8063.8** | 8105.6 |
| 适应度标准差 | 348.939 | 306.184 | 345.998 | 322.854 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.328 | **0.361** | 3.769 | 9.271 |

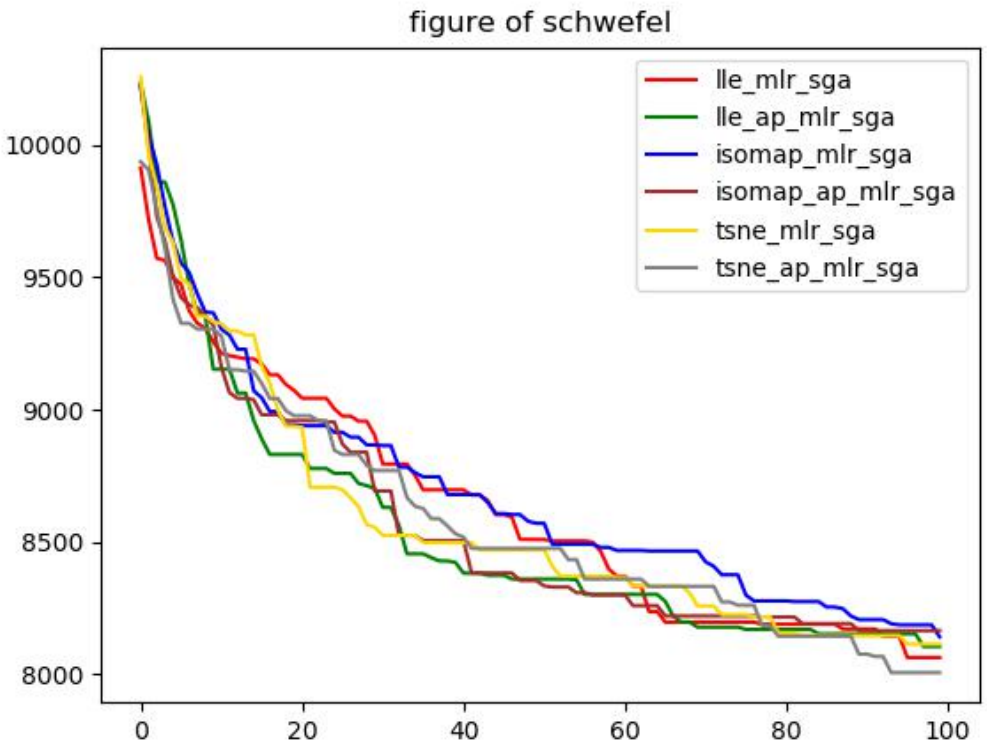


图 4-15 基于 Schwefel function 的算法结果

表 4-6 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 7379 | 7654 | 7316 | 7346 |
| 平均适应度 | 8141.9 | 8165.4 | 8114.6 | 8006.9 |
| 适应度标准差 | 349.519 | **266.261** | 326.296 | 389.065 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.867 | 6.630 | 210.251 | 213.026 |

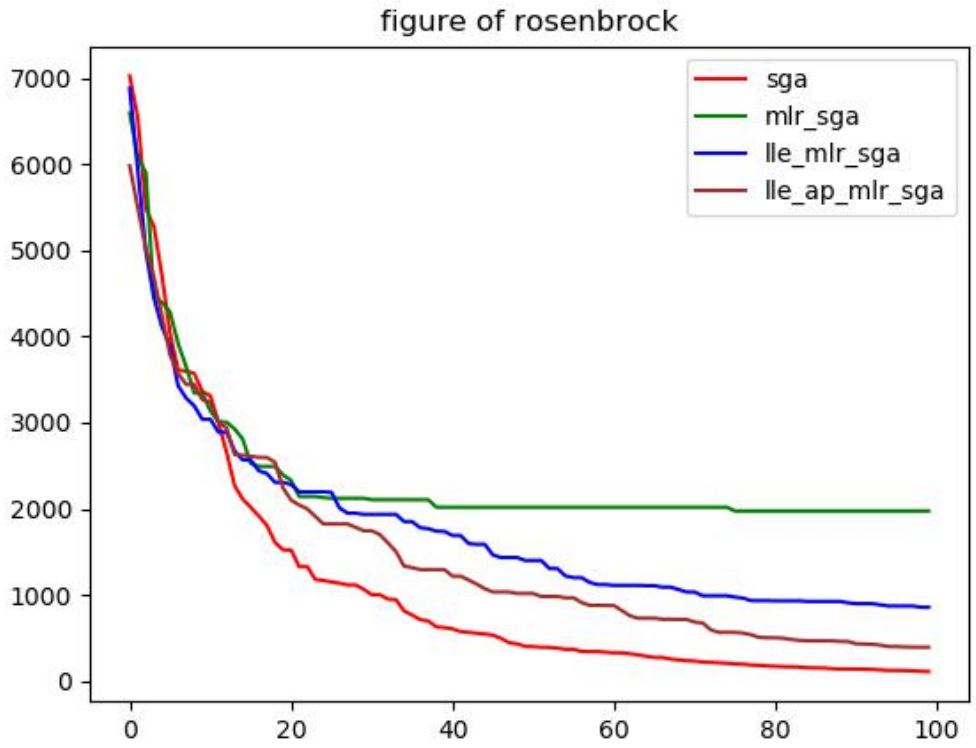


图 4-16 基于 Rosenbrock function 的算法结果

表 4-7 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 67 | 916 | 464 | 310 |
| 平均适应度 | 110.6 | 1974.6 | 857.8 | 393.6 |
| 适应度标准差 | 24.662 | 622.557 | 197.069 | 65.290 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.130 | **0.262** | 3.720 | 8.539 |

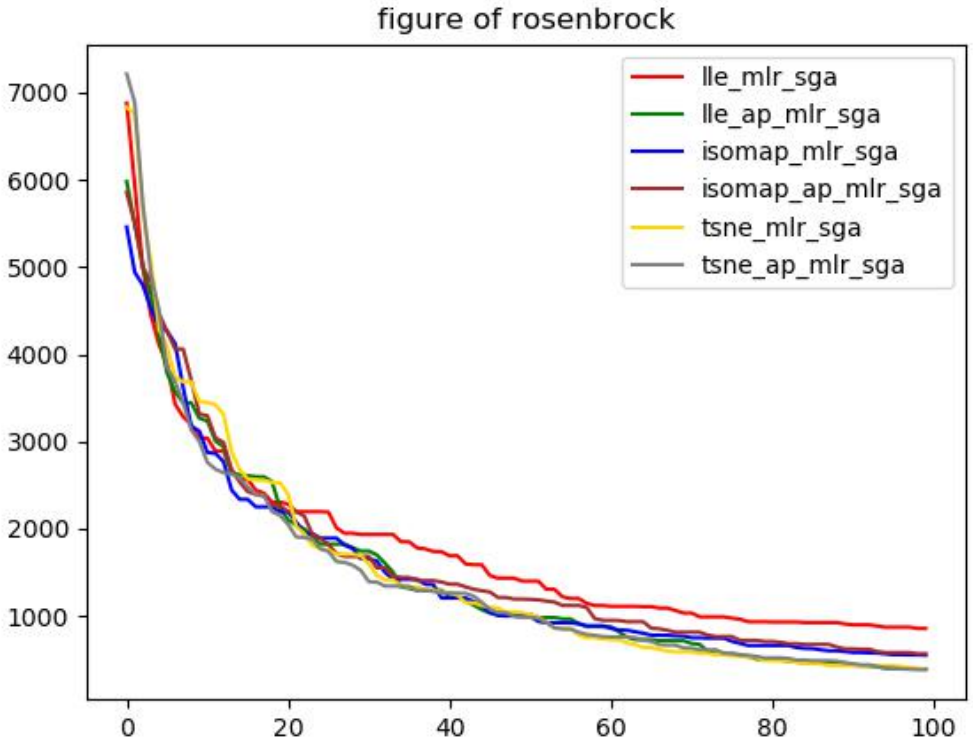


图 4-17 基于 Rosenbrock function 的算法结果

表 4-8 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 389 | 439 | 307 | **247** |
| 平均适应度 | 551.4 | 567.6 | 385.9 | **386.1** |
| 适应度标准差 | 110.513 | 90.417 | **49.560** | 101.320 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.149 | 6.714 | 204.906 | 213.601 |

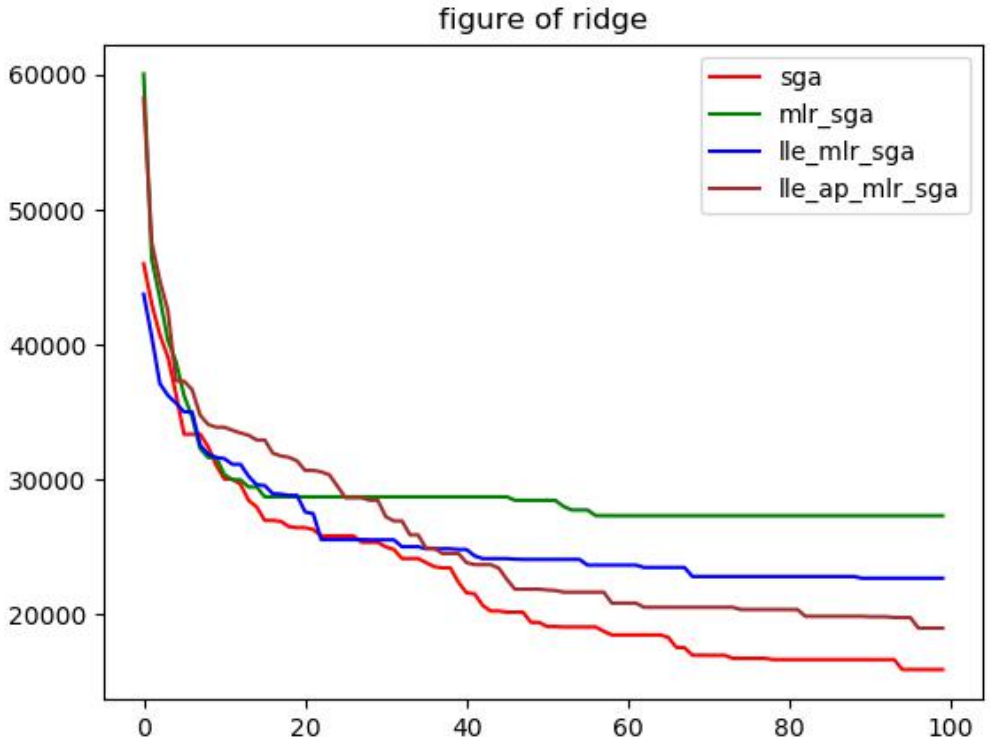


图 4-18 基于 Ridge function 的算法结果

表 4-9 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 10766 | 20313 | 16965 | **9807** |
| 平均适应度 | 15916.2 | 27322.3 | 22690.2 | 18992.8 |
| 适应度标准差 | 2986.816 | 4214.491 | 3476.690 | 5065.221 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.194 | **0.319** | 3.787 | 8.390 |

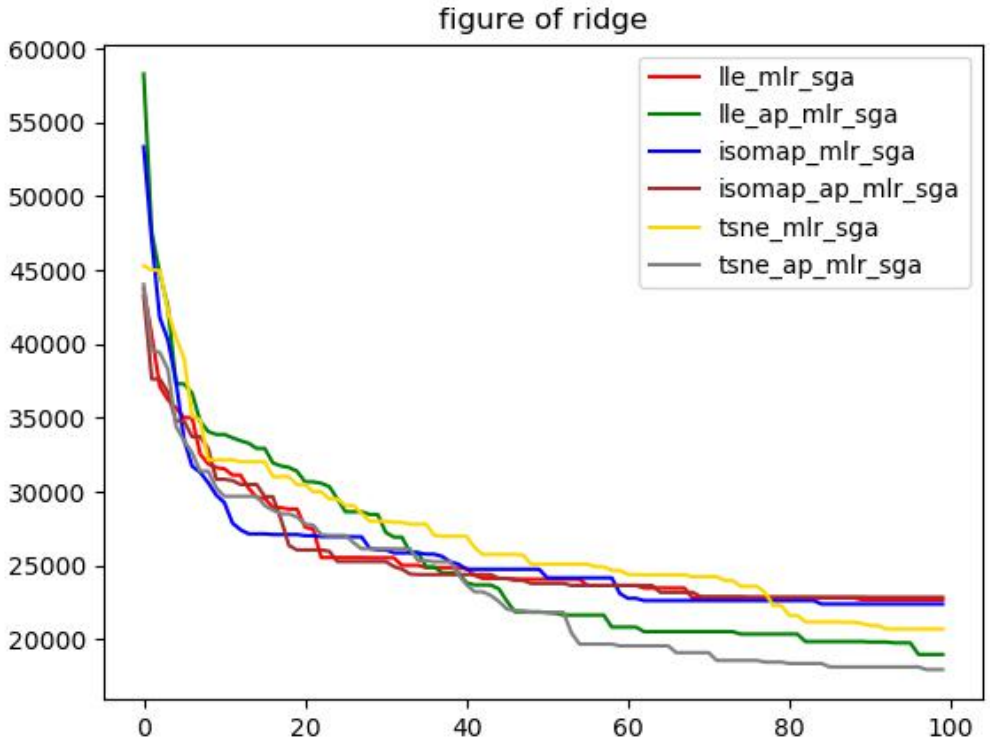


图 4-19 基于 Ridge function 的算法结果

表 4-10 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 17123 | 17879 | 16111 | 11277 |
| 平均适应度 | 22405.8 | 22866.7 | 20712.9 | **17967** |
| 适应度标准差 | 2965.637 | 3739.972 | **2637.149** | 3612.585 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.462 | 6.579 | 204.493 | 204.042 |

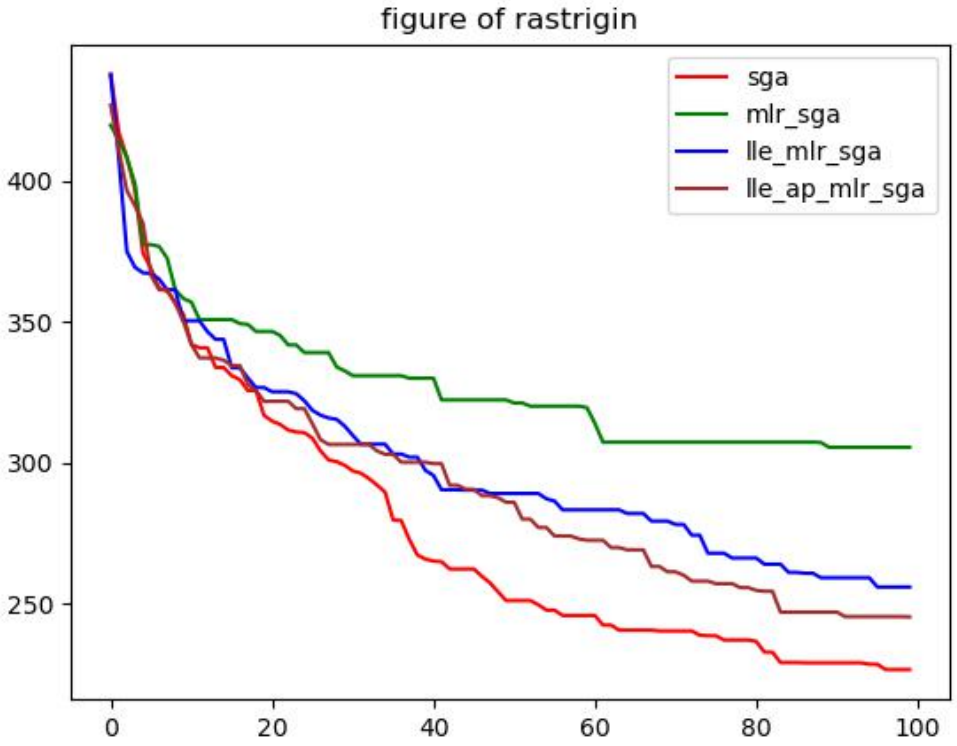


图 4-20 基于 Rastrigin function 的算法结果

表 4-11 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 204 | 250 | 218 | **216** |
| 平均适应度 | 226.5 | 305.4 | 255.8 | **245.2** |
| 适应度标准差 | 10.623 | 27.586 | 19.893 | 19.968 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.407 | **0.409** | 3.853 | 9.095 |

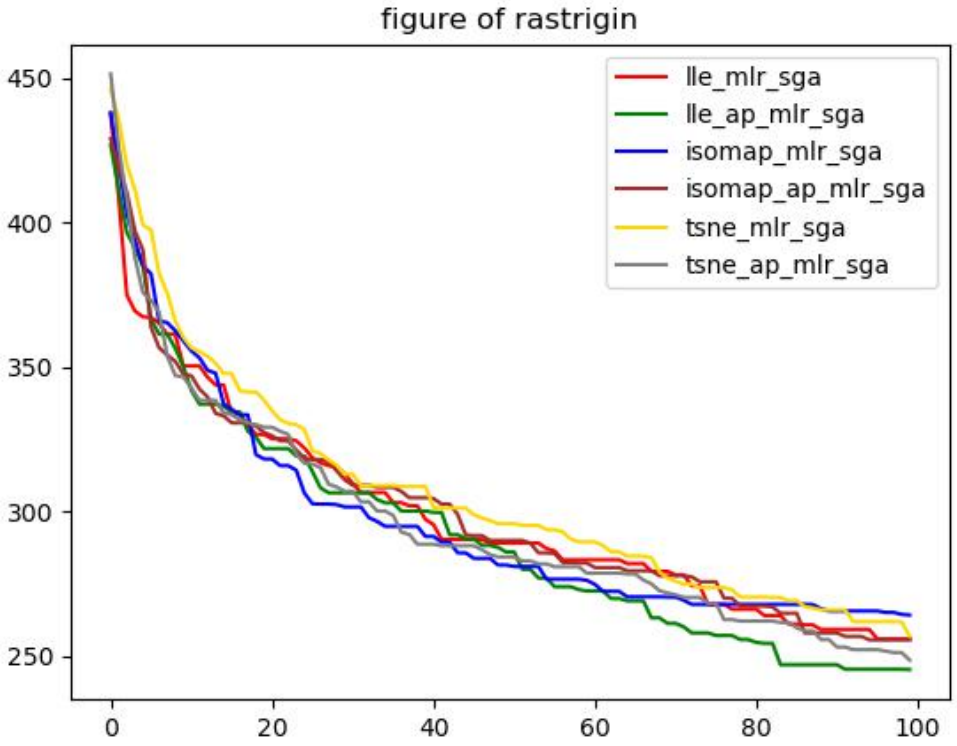


图 4-21 基于 Rastrigin function 的算法结果

表 4-12 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 244 | 227 | 219 | 234 |
| 平均适应度 | 264.1 | 255.5 | 256.8 | 248.5 |
| 适应度标准差 | 12.242 | 18.249 | 18.252 | **8.879** |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.874 | 6.826 | 207.788 | 205.254 |

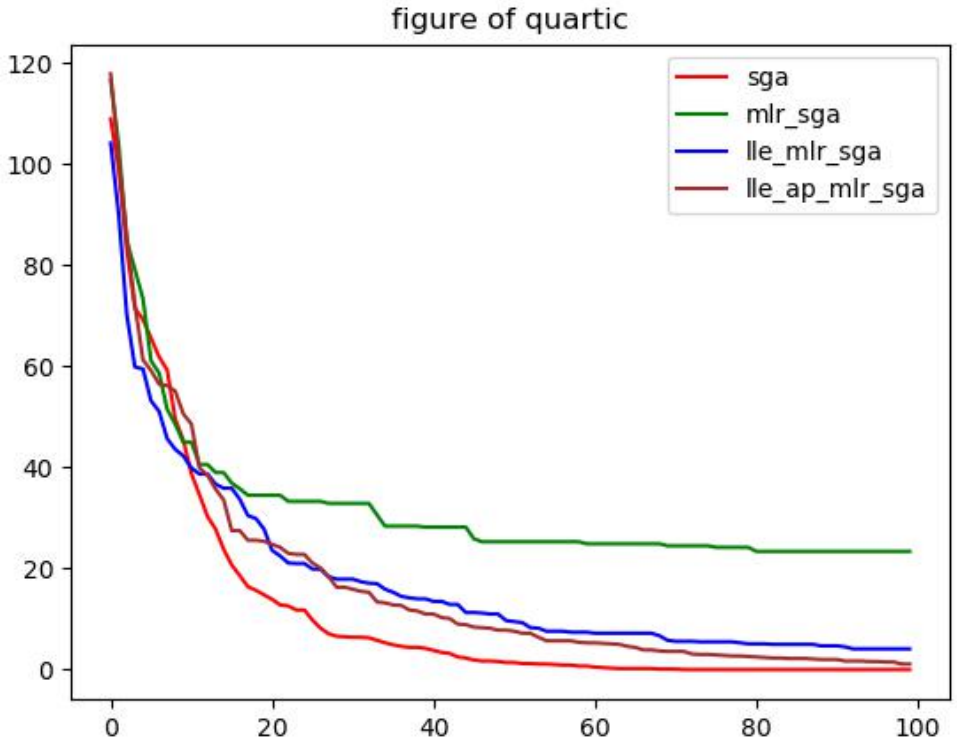


图 4-22 基于 Quartic function 的算法结果

表 4-13 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 0 | 14 | 2 | **0** |
| 平均适应度 | 0 | 23.4 | 4.1 | **1.1** |
| 适应度标准差 | 0 | 8.991 | 1.3 | 0.7 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.133 | **0.283** | 3.711 | 9.337 |

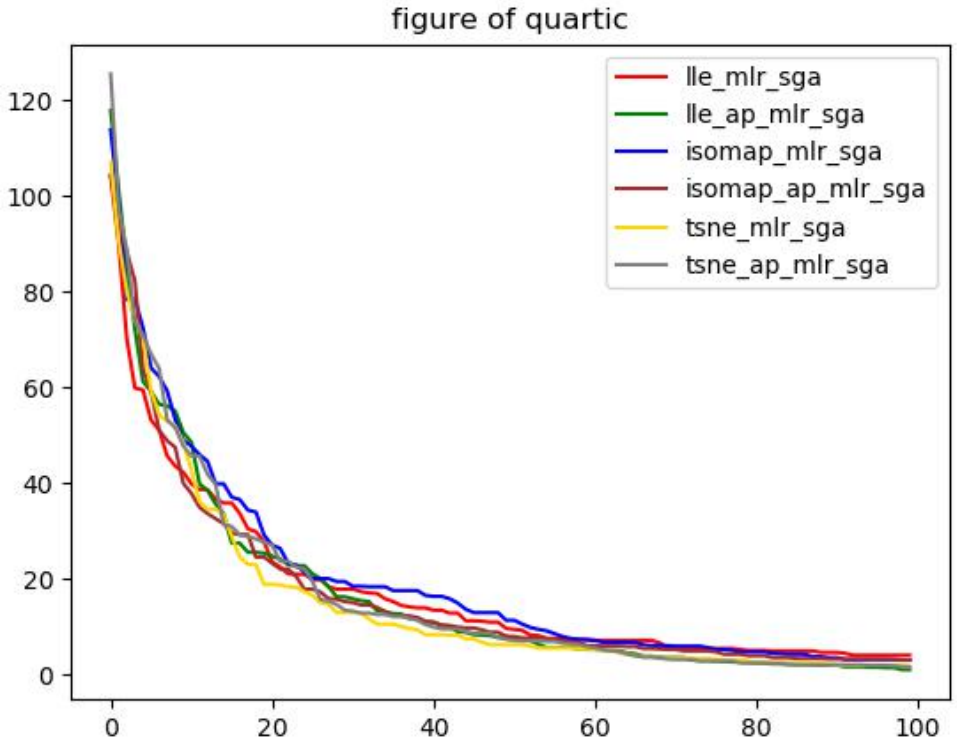


图 4-23 基于 Quartic function 的算法结果

表 4-14 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 1 | 2 | 1 | 1 |
| 平均适应度 | 3.1 | 3.1 | 1.7 | 1.7 |
| 适应度标准差 | 1.044 | 0.7 | **0.458** | 0.781 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.273 | 6.316 | 204.166 | 211.520 |

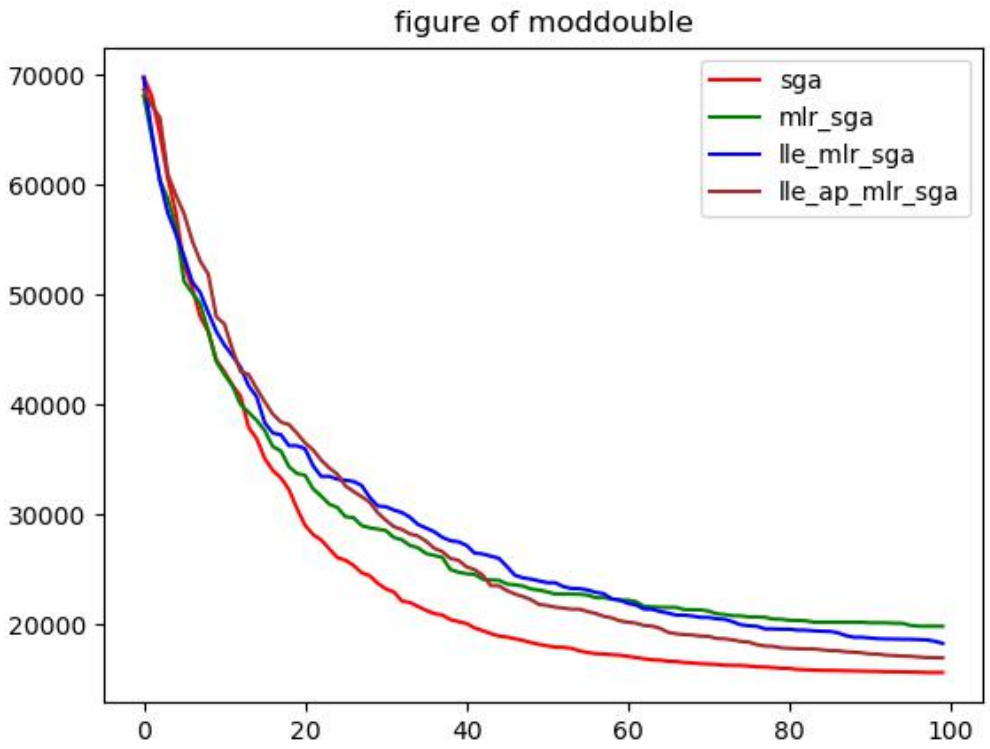


图 4-24 基于 Moddouble function 的算法结果

表 4-15 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 15600 | 17458 | 17395 | **16364** |
| 平均适应度 | 15655.8 | 19869.5 | 18313.7 | **16989.9** |
| 适应度标准差 | 34.507 | 1634.776 | 547.520 | **353.293** |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.312 | **0.4726** | 3.915 | 9.723 |

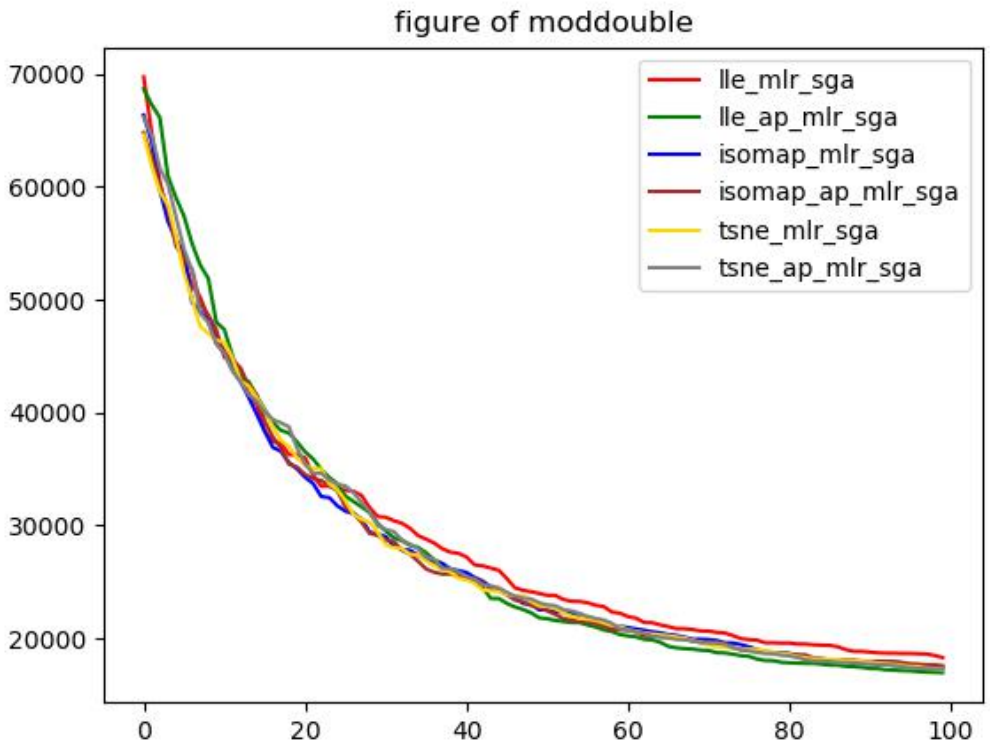


图 4-25 基于 Moddouble function 的算法结果

表 4-16 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 16514 | 16818 | 16603 | 16725 |
| 平均适应度 | 17404.1 | 17592.1 | 17300.5 | 17317.4 |
| 适应度标准差 | 669.777 | 359.814 | 444.802 | 420.085 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.462 | 6.881 | 210.667 | 209.037 |

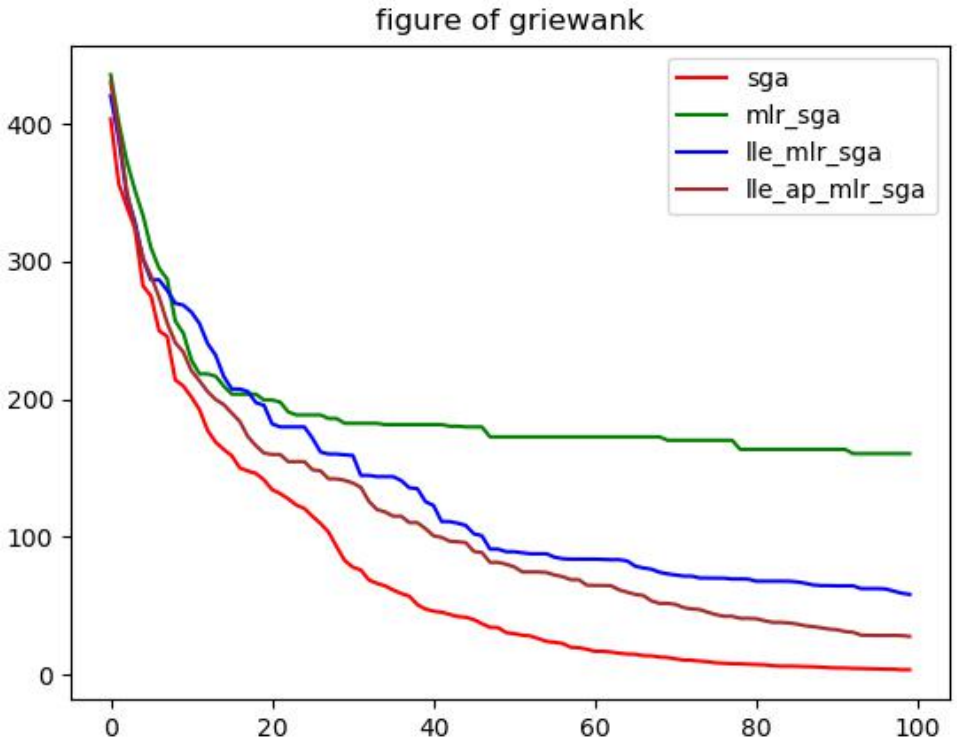


图 4-26 基于 Griewank function 的算法结果

表 4-17 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 2 | 115 | 23 | **23** |
| 平均适应度 | 3.5 | 160.4 | 58.2 | **27.7** |
| 适应度标准差 | 0.80 | 27.525 | 22.493 | **4.450** |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.230 | **0.309** | 3.796 | 9.270 |

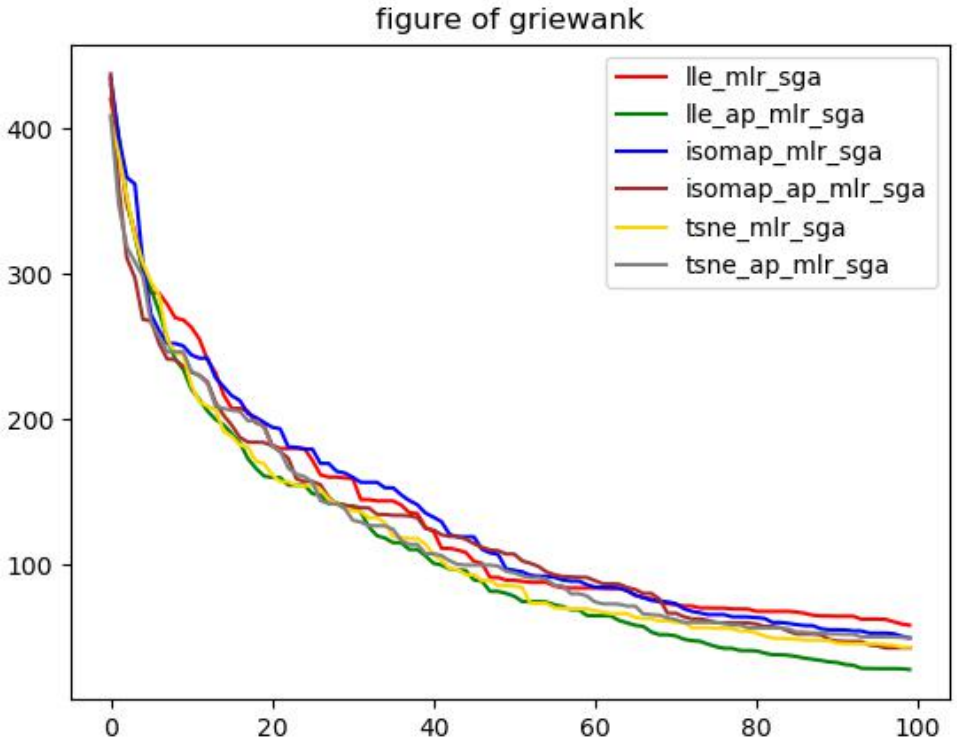


图 4-27 基于 Griewank function 的算法结果

表 4-18 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 30 | 30 | 35 | 33 |
| 平均适应度 | 49.4 | 42.6 | 43.1 | 49.4 |
| 适应度标准差 | 15.239 | 7.172 | 6.410 | 12.916 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.878 | 6.881 | 204.386 | 212.966 |

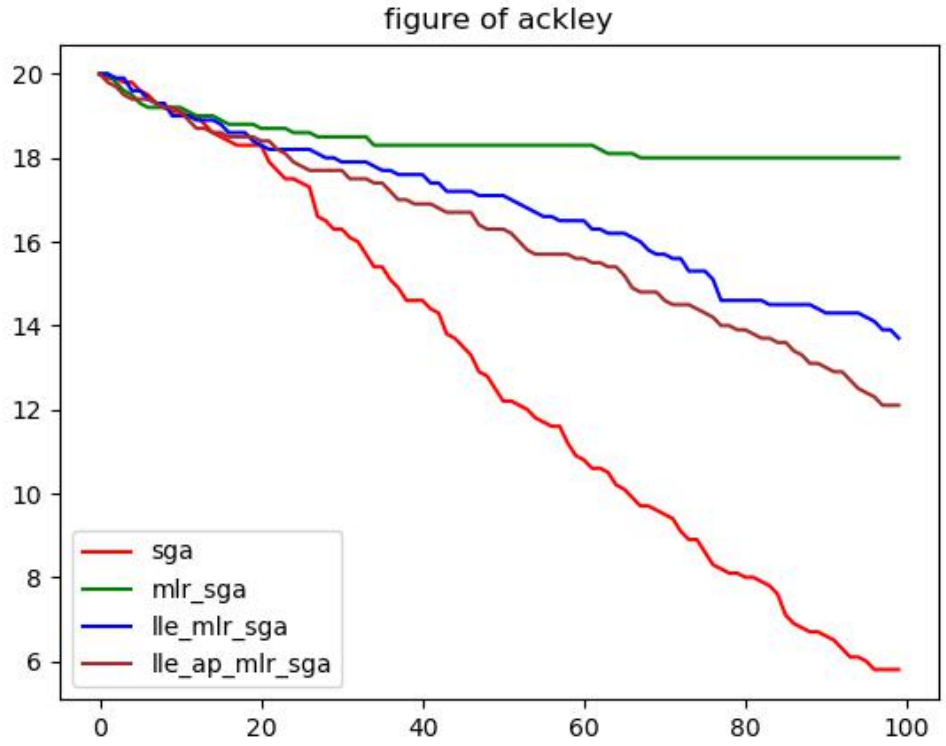


图 4-28 基于 Ackley function 的算法结果

表 4-19 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 5 | 16 | 12 | 11 |
| 平均适应度 | 5.8 | 18 | 13.7 | **12.1** |
| 适应度标准差 | 0.6 | 1 | 1.004 | **0.7** |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.419 | 0.405 | 3.922 | 9.092 |

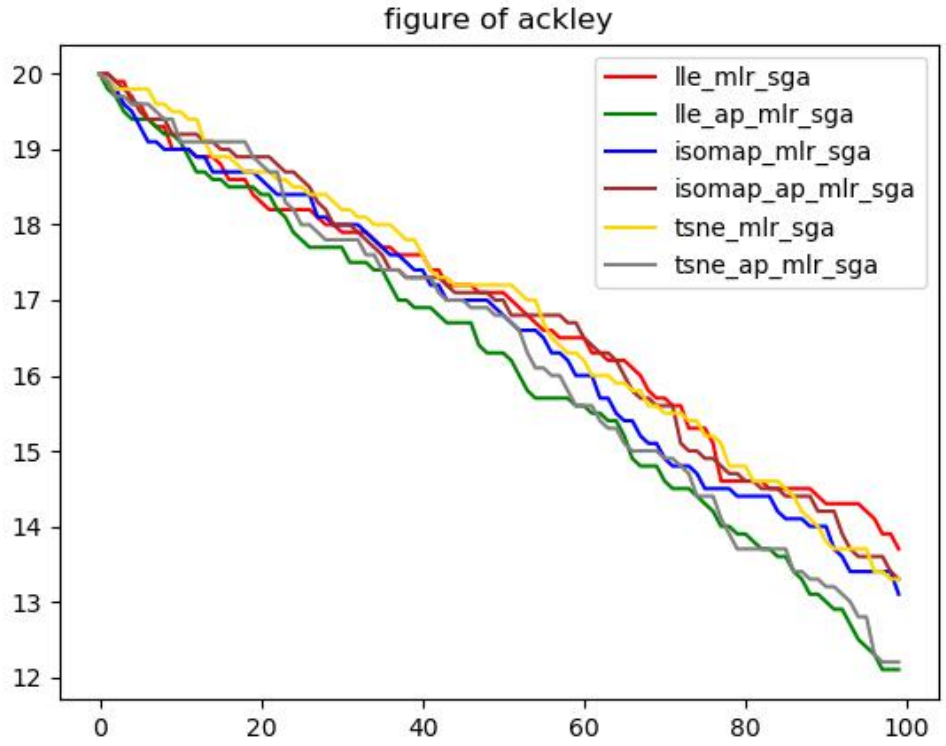


图 4-29 基于 Ackley function 的算法结果

表 4-20 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 12 | 12 | 12 | 10 |
| 平均适应度 | 13.1 | 13.3 | 13.3 | 12.2 |
| 适应度标准差 | 0.830 | 1.004 | 0.9 | 0.979 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.604 | 6.786 | 205.910 | 209.627 |

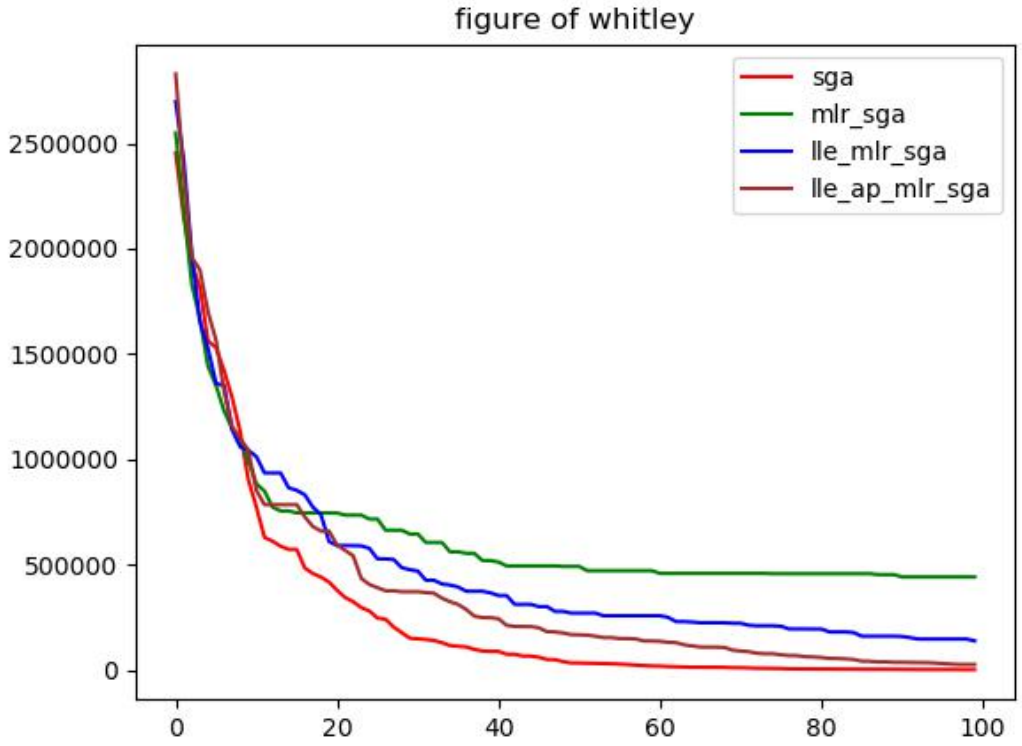


图 4-30 基于 Whitley function 的算法结果

表 4-21 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 1791 | 220260 | 35872 | **12196** |
| 平均适应度 | 2429.1 | 441952.7 | 138758.1 | **26637.8** |
| 适应度标准差 | 424.242 | 174932.544 | 70087.986 | 15876.762 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 4.212 | **3.385** | 6.696 | 12.349 |

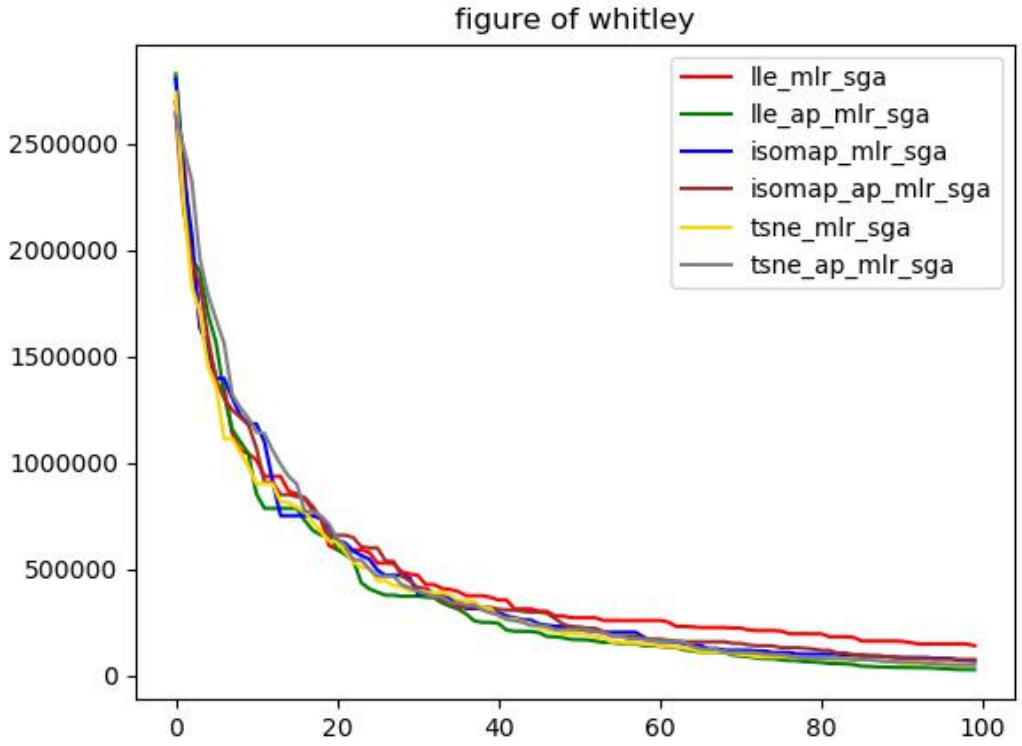


图 4-31 基于 Whitley function 的算法结果

表 4-22 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 38408 | 46348 | 19061 | 24269 |
| 平均适应度 | 66423.5 | 74476.7 | 47682.3 | 49880.5 |
| 适应度标准差 | 19434.750 | 30885.618 | **15026.498** | 20600.581 |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.139 | 6.258 | 204.647 | 207.285 |

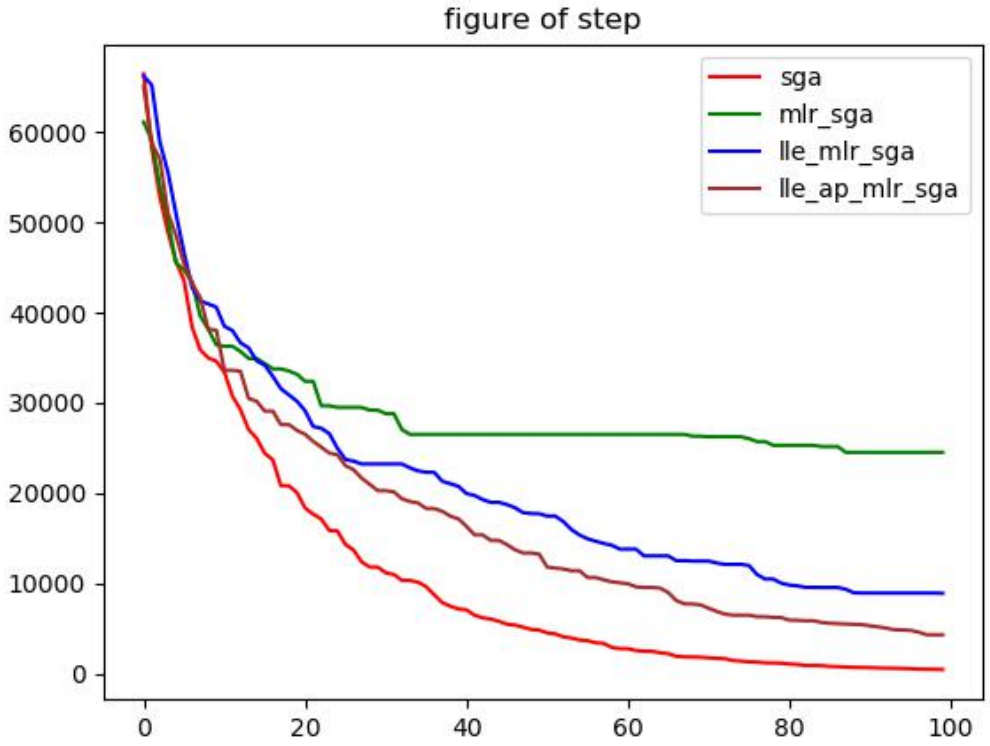


图 4-32 基于 Step function 的算法结果

表 4-23 不同算法结果

| | SGA | MLR_SGA | LLE_MLR_SGA | LLE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 237 | 19348 | 5866 | 2414 |
| 平均适应度 | 428.8 | 24512 | 8883.3 | 4290 |
| 适应度标准差 | 120.329 | 4023.854 | 2299.488 | 800.411 |
| 适应度函数调用次数 | 5000 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 0.131 | **0.299** | 3.741 | 9.573 |

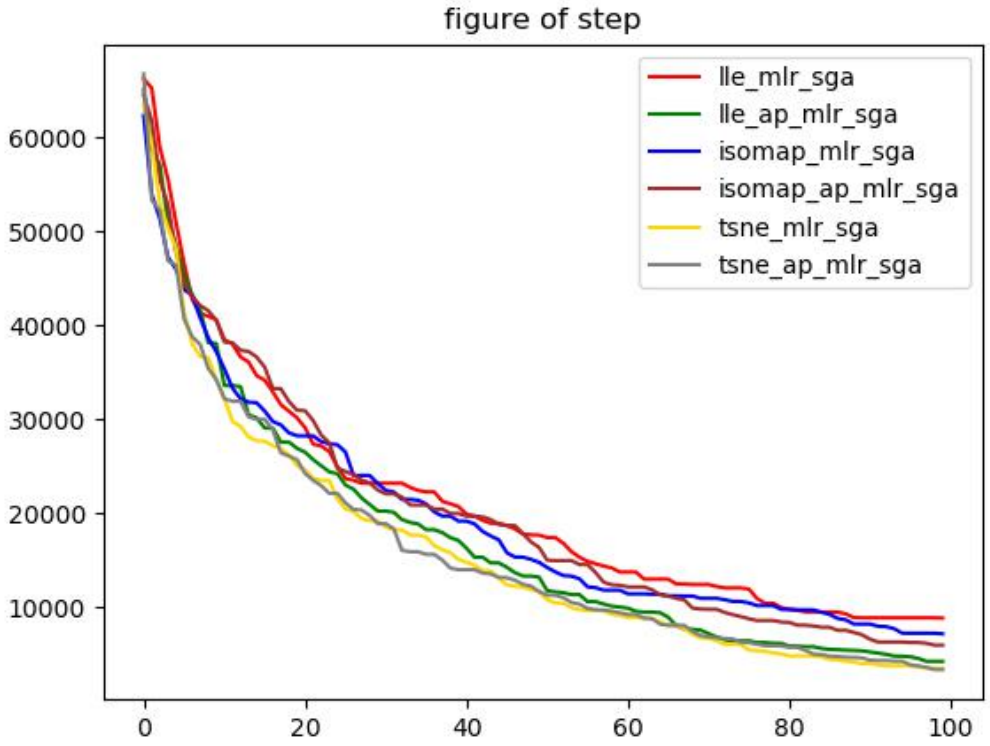


图 4-33 基于 Step function 的算法结果

表 4-24 不同算法结果

| | ISOMAP_MLR_SGA | ISOMAP_AP_MLR_SGA | TSNE_MLR_SGA | TSNE_AP_MLR_SGA |
|---|---|---|---|---|
| 最佳适应度 | 5231 | 3610 | 2354 | **2237** |
| 平均适应度 | 7211.6 | 6010.9 | 3550.6 | **3412.6** |
| 适应度标准差 | 1070.445 | 1140.610 | 736.0183 | **665.335** |
| 适应度函数调用次数 | 2600 | 2600 | 2600 | 2600 |
| 算法平均运行时间 | 6.37 | 6.848097532 | 211.067 | 209.577 |

## 4.3 实验结果分析：

通过上述的实验，我们可以看到 SGA、MLR_SGA、LLE_MLR_SGA、LLE_AP_MLR_SGA、ISOMAP_MLR_SGA、ISOMAP_AP_MLR_SGA、TSNE_MLR_SGA、TSNE_AP_MLR_SGA 基于不同代理模型的算法，对于不同适应度函数的最佳适应度、平均适应度、适应度标准差、适应度函数调用次数、算法平均运行时间等数据。从实验的运行结果来看，传统的遗传算法运行时间运行最快。然而这并不影响我们本身的结论，因为在我们的假设中，因为代理模型中所涉及的多元线性回归算法、流形学习算法以及聚类算法事实上都是有一定的算法时间复杂度的，而本身在上述实验中使用的测试函数都为相对较简单的函数，故反而在使用了代理模型之后，由于整体算法复杂度的增加，时间消耗反而比原先的传统遗传算法要高。对于加速，程度我们可以利用下述公式进行表达：

$$N=\frac{t'_{\text{遗传算法流程}}+t'_{\text{适应度函数调用总时间}}+t'_{\text{代理模型消耗时间}}}{t_{\text{遗传算法流程}}+t_{\text{适应度函数调用总时间}}} \quad \text{.....(4.1)}$$

其中 $t'$ 代表加入代理模型后的时间。由于容易得知，代理模型消耗的时间以及遗传算法流程本身的消耗时间并没有随着代理模型的改变而将发生显著变化，所以可以得出代理模型最大的值为：

$$\begin{aligned} N_{\max} &\approx \frac{t'_{\text{适应度函数调用总时间}}}{t_{\text{适应度函数调用总时间}}} \\ &= \frac{1}{P_{\text{训练集选择概率}}} \\ &= \frac{C_{\text{种群大小}}}{C_{\text{种群直接调用适应度函数进行计算的个体数量}}} \end{aligned} \quad \text{.....(4.2)}$$

在使用了代理模型的算法之间进行比较，则 MLR_SGA 最快，LLE_MLR_SGA 次之，LLE_AP_MLR_SGA 则最慢。这一结果也符合本实验开始的预期结果，因为随着代理模型的复杂程度的加剧，算法的复杂度必然也随之提升。

单纯从上述角度来评价代理模型，无法突出流形学习、聚类算法在代理模型

中的作用。结合前面的分析，即代理模型在面对高维度、较为复杂的问题时，很可能并没有办法得到一个精确度很高的预测效果，下面我们则对代理模型的预测准确性进行分析。

如果要对每一轮迭代中代理模型的预测结果与实际值进行比较分析，则分析过程会变得十分繁琐。事实上，因为所有用于测试的基准函数都是寻找全局最小值，因此我们可以通过观察在不同基准函数中，基于不同算法的代理模型的函数值下降速度来判断不同预测模型的准确性。因为一个更加准确的预测模型可以提供给遗传算法一个与现实更相近的进化方向。可以看到，从函数值下降速度的角度来看，总体上 LLE_AP_MLR_SGA，LLE_MLR_SGA 次之，而 MLR_SGA 的结果则相对不稳定。因而 LLE_AP_MLR_SGA 所代表的基于流形学习以及降维算法的代理模型预测的准确性最高，MLR_SGA 最低。

在确定了基于流形学习以及降维算法的代理模型的确有不错的优化性能后，我们可以尝试比较采用了不同流形学习策略的代理模型性能。可以看到总体而言，基于 t-SNE 以及 LLE 的代理模型有相对较为接近的预测精度，而 Isomap 整体来看相对的预测准确性较低。然而在考虑算法复杂度的情况时，可以发现 t-SNE 所耗费的时间要远远的高于 LLE 以及 Isomap，因此其加速效果在实际的应用之中很可能无法较为显著地体现。综上所述，基于 LLE 的代理模型综合表现更加优秀。

为了体现上述基于代理模型的加速效果，我们通过对适应度函数额外地添加一个延迟，来模拟适应度函数较为复杂的情况。通过不断地实验，我们发现，当我们设定额外延迟时间为 0.005 秒左右时，可以达到使用代理模型的遗传算法运行时间短于传统遗传算法运行时间的效果，具体数据如下所示：

**表 4-25　不同算法结果**

| 算法类型 | 适应度函数调用次数 | 算法运行时间 |
|---|---|---|
| 传统遗传算法 | 5000 | 25.295 |
| 基于 LLE 进行降维的遗传算法 | 2600 | 23.446 |
| 基于 LLE 以及 AP 聚类的遗传算法 | 2600 | 23.484 |

## 4.4 本章小结：

本章我们通过多个基准函数，对基于不同代理模型的算法的性能进行测试。之后，通过对于不同适应度函数的最佳适应度、平均适应度、适应度标准差、适应度函数调用次数、算法平均运行时间等数据对不同的代理模型进行分析。可以看到，通过使用 LLE 流形学习降维算法，AP 聚类算法以及多元线性回归，我们可以达到在显著降低调用代理模型次数的基础之上，同时兼顾代理模型的预测准确度，并能找到全局最优解。

# 第 5 章　总结与展望

## 5.1　总结：

在面对一些较高维度、且具有高度非线性性质的一些科学数值计算以及大程度的工程寻找最优解的问题中，相较于以往的可用公式表达的精确优化算法，遗传算法有着较为显著的优势。然而该类算法其实也有非常明显的不足之处，即算法在计算过程中常常有着较高的复杂度，以及需要频繁的调用适应度函数从而确定能找到一个全局最优解。但在一些适应度函数较为复杂的问题中，过于频繁地调用适应度函数，往往会产生不可估量的巨额时间开销，这给实践中问题的解决带来了巨大的阻碍。因此，如何对对算法进行改进，减少适应度函数 的调用，是一个非常重要的问题。

本文大致的工作内容如下：

(1) 研究了遗传算法的表现，探讨了为何遗传算法能在科研、工程等领域有着较高的运用价值，同时分析传统遗传算法的阻碍，以及为了解决相关缺陷而引入的其他算法的理论知识，如流形学习、多元线性回归以及聚类等相关算法的大致原理。

(2) 为了改进传统的遗传算法，在传统框架上加入了代理模型，结合使用了流形学习、聚类以及多元线性回归的方法，从而使得遗传算法调用适应度函数的次数大幅地减少，同时保证了一定的预测精准性。

## 5.2　展望：

由于其搜索方向随机的特定，遗传算法事实上不能有一个确定的收敛性或者收敛区间。因此，对于遗传算法的确定性以及随机性该如何进行一个合适的权衡，并入和对其设定相应的参数，亦或是给出相应参数的的选择范围，是一个值得深入探讨研究的方向。

# 参考文献

[1] HollanD J. Adaptation in Natural and Artificial Systems[M]. New York: MIT PRESS, 1992.

[2] Grefenstette JJ, Fitzpatrick JM. Genetic search with approximate function evaluations [C]. In Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp.112-120, 1985.

[3] Papadrakakis M, Lagaros N, and Tosmpanakis Y. Structural Optimization using evolution strategies and neural networks. In Congress on Computational Mechanics, 1997

[4] Rasheed K. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In Proceeding of genetic and Evolutionary Computation Conference, PP628-635,2000

[5] Kim H.S., Cho S.B. An efficient genetic algorithm with less fitness evaluation by clustering [C]. In Proceedings of IEEE Congress on Evolutionary Computation, pp. 887-894, 2001.

[6] Branke J, Schmidt C.Faster convergence by means of fitness estimation [J].Soft Computing,2005,9(1):13-20.

[7] Jin Y, and Branke J. Evolutionary Optimization in Uncertain Environments—A Survey [J]. IEEE Transactions on Evolutionary Computation, 2005, 9(3): 303-317.

[8] Ingo Paenke, Jürgen Branke, and Yaochu Jin. Efficient Search for Robust Solutions by Means of Evolutionary Algorithms and Fitness Approximation [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(4): 405-420.

[9] Michael D.Shmidt and Hod Lipson, Coevolution of Fitness Predictor. In IEEE Transactions Evolutionary Compution, PP 324-335, 2008

[10]Xiaosong Han, Yanchun Liang, Gaoyang Li, Zhengguang Li, Binghong Wang, Chunguo Wu. An Efficient Genetic Algorithm with Fitness Estimation. International Journal of Computational Methods. In International Journal of Computational Methods, PP 213-220,2012

[11]Farina M, A minimal cost hybrid strategy for Pateto optimal front approximation.

Evolutionary Optimization,3(1):41-52, 2001

[12]周春光, 梁艳春. 计算智能优化算法理论与应用 [M]. 长春: 吉林大学出版社, 2009.

[13]Hotelling H, ‘Analysis of A Complex of Statistical Variables into Prindpal Components,”Journal of Educational Psychology,vol.24,pp.417-441,1933.

[14][Fisher RA.“The Use of Multiple Measurements in Taxonomic Problems,”Annals of Eugenics,vol.7,pp.179-188,1936

[15]Thomas L.Griffiths and Michael L.Kalish,A Multidimensional scaling approach to mental multiplication,Memory&Cognition,vol.30(1),pp.97-106,2002

[16]Huber P J. Projection pursuit.Annals of Statistics,13(2):435 475 (with comments, pp. 475 525), June 1985.

[17]TENENBAUM J,SILVADD,LANGFORD J. Aglobal geometric framework for nonlinear dimensionality reduction[J].Science,2000,290(5500):2319-2323.

[18]ROWEISS,SAULL. Nonlinear dimensionality reduction by locally linear embedding [J].Science,2000,290(5500):2323-2326.

[19]Brendan JF, Delbert D. Clustering by passing messages between data points [J]. Science. 2007, 315(5814): 972-976.

# 致　　谢

感谢我吉林大学的韩霄松老师以及其他的良师益友们。

感谢父母。