统计推断在数模转换系统中的应用

组号：62　　姓名:吴思源　　学号:5140309109　　姓名:安积善　　学号:5140309119

摘要：本报告是上海大学电子信息与电气工程学院课程设计《统计推断在数模、模数转换系统中的应用》 的课程论文初稿，主要展示了统计推断在模数、数模转换系统中的应用，探究针对某一检测模块中输入输出特性呈明显的非线性关系，研究和分析一组电压信号 X 和被测物理量 Y 的关系，运用一定的数理统计方法，经过特征点选取、算法研究、拟合比较等一系列过程，借助 MATLAB 建立电压 X 和被测物理量 Y 的关系曲线模型。最终实现以少量数据反映整体系统特性的效果，从而有效降低工程设计上的成本，力求为该模块的批量生产设计一种成本合理的传感特性校准（定标工序）方案。
关键词：样本，特征点，曲线拟合，遗传算法，MATLAB，优化采样

ABSTRACT: This article is for Course Design-Application of Statistical Inference inAD&DAInverting System， School of Electronic Information and Electrical Engineering,Shanghai Jiao Tong University, and itmainly shows the application of the statistical inference in ADC and DAC system.……
Key words： sample, feature points, curve-fitting, Genetic Algorithm,MATLAB, optimal sampling

1 引言

在生产和研究中，对物理现象的研究可以通过确定其输入输出的关系来确定其性质，这就有必要提出一种解决方案来确定这二者的关系。但是实际研究生产中发现，我们很难找到一个能够精准描述二者关系的函数，这就要求我们运用实际测得的数据，通过曲线拟合来找到一个最符合输入与输出关系的函数。为了保证准确找到最优解，我们就需要运用统计推断的知识找到合适的特征点，对这些特征点进行拟合，计算得到的曲线与实际输出的残差，残差最小的解围最优解。

2 数据统计与初步分析

2.1 课题概述

假定有某型投入批量试生产的电子产品，其内部有一个模块，功能是监测某项与外部环境有关的物理量（可能是温度、压力、光强等）。该监测模块中传感器部件的输入输出特性呈明显的非线性。本课题要求为该模块的批量生产设计一种成本合理的传感特性校准（定标工序）方案。

2.2 设计模型

监测模块的组成框图如图 1-1，

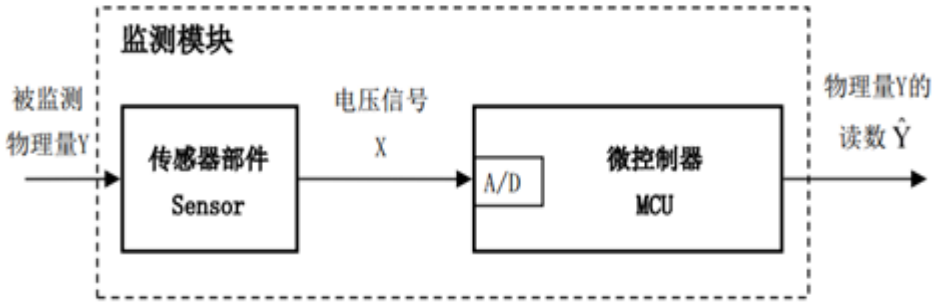

图 1　监测模块的组成框图

其中，传感器部件（包含传感器元件及必要的放大电路、调理电路等）的特性是我们关注的重点。传感器部件监测的对象物理量以符号 Y 表示；传感部件的输出电压信号用符号 X 表示，该电压经模数转换器（ADC）成为数字编码，并能被微处理器程序所读取和处理，获得信号 $\hat{Y}$ 作为 Y 的读数（监测模块对 Y 的估测值）。

所谓传感特性校准，就是针对某一特定传感部件个体，通过有限次测定，估计其 Y 值与 X 值间一一对应的特性关系的过程。

一个传感部件个体的输入输出特性大致如图 2 所示。

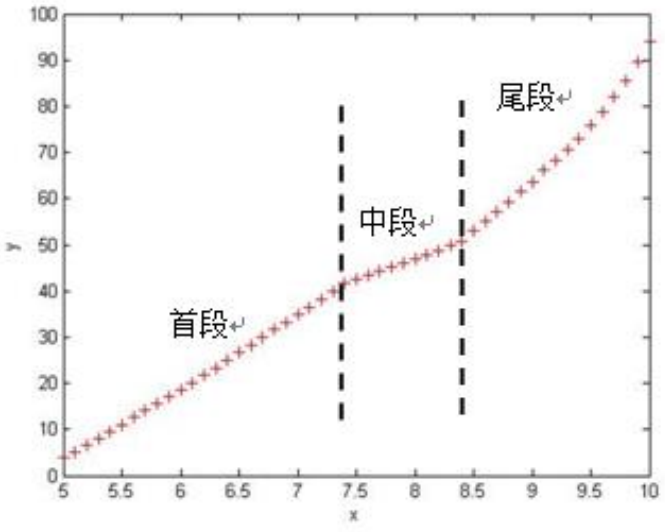

图 2　传感特性图示

2.3　成本计算

为评估和比较不同的校准方案，特制定以下成本计算规则。

1).单点定标误差成本

$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.4\\ 0.1 & \text{if } 0.4<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.6\\ 0.7 & \text{if } 0.6<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.8\\ 0.9 & \text{if } 0.8<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases} \qquad (1)$$

单点定标误差的成本按式（1）计算，其中 $y_{i,j}$ 表示第 i 个样本之第 j 点 Y 的实测值，$\hat{y}_{i,j}$ 表示定标后得到的估测值（读数），该点的相应误差成本以符号 $s_{i,j}$ 记。

2). 对某一样本 i 的定标成本

$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i \qquad (2)$$

对样本 i 总的定标成本按式（2）计算，式中 n_i 表示对该样本个体定标过程中的单点测定次数。

3）。校准方案总成本

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \qquad (3)$$

按式（3）计算评估校准方案的总成本，即使用该校准方案对标准样本库中每个样本个体逐一定标，取所有样本个体的定标成本的统计平均。

注：我们求取统计平均意义上的最好方法。总成本最低的方案，对特定样本的定标成本不一定最低。

3 选择拟合方式

3.1 了解拟合算法

数据样本实验数据的正确处理, 关系到是否能达到实验目的, 得出明确结论。传统的数据处理方法, 很难得到一条很好地适应所有点的曲线, 同时也无法估计所得曲线的精度, 由此所确定的特征值就可能有较大的误差, 且没有建立起由这些点构成曲线的数学模型,这些都将直接影响利用数学方法进行解析分析。通常的做法是在进行实验数据分析时，采用某种拟合方式，并在某种最佳准则下找出最合适的曲线。常见的拟合方式有基于最小二乘原理的多项式拟合、指数函数拟合、傅里叶级数拟合和插值计算等。由于这一部分仅仅讨论采取哪种拟合方式更好，所以可以先确定若干组特征点，然后做拟合。选择一种拟合（或插值）方法，也就是选择表达式。

3.2 具体拟合方法的选择

查阅相关资料及翻看概率统计书籍，了解了相关知识后。主要可以采取的方法有如下两种：

1. 插值

在特征点位置上，表达式与实验值无误差。

2 拟合

在特征点位置上，表达式与实验值可以有误差（残差）。

通过讲座我们了解到由于线性插值误差较大，而三次样条插值计算量较大，基于准确性和成本的考虑，应采用多项式拟合，由于不可能为线性，故从二次多项式开始考虑当阶数增加时，标准差降低，所得精度越来越高，但计算量也随之加大。除二阶拟合外，三阶四阶以至于更高阶相差不大。

插值拟合代码见附页。拟合

4 遗传算法

4.1 了解遗传算法

1.遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

2.遗传算法的基本运算程如下：

1).初始化：设置进化代数计数器 t=0，设置最大进化代数 T，随机生成 M 个个体作为初始群体 P(0)。

2).个体评价：计算群体 P(t)中各个个体的适应度。

3).选择运算：将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

4).交叉运算：将交叉算子作用于群体。遗传算法中起核心作用的就是交叉算子。

5).变异运算：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。群体 P(t)经过选择、交叉、变异运算之后得到下一代群体 P(t+1)。

6).终止条件判断:若 t=T,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

3. 遗传算法流程图如图三。

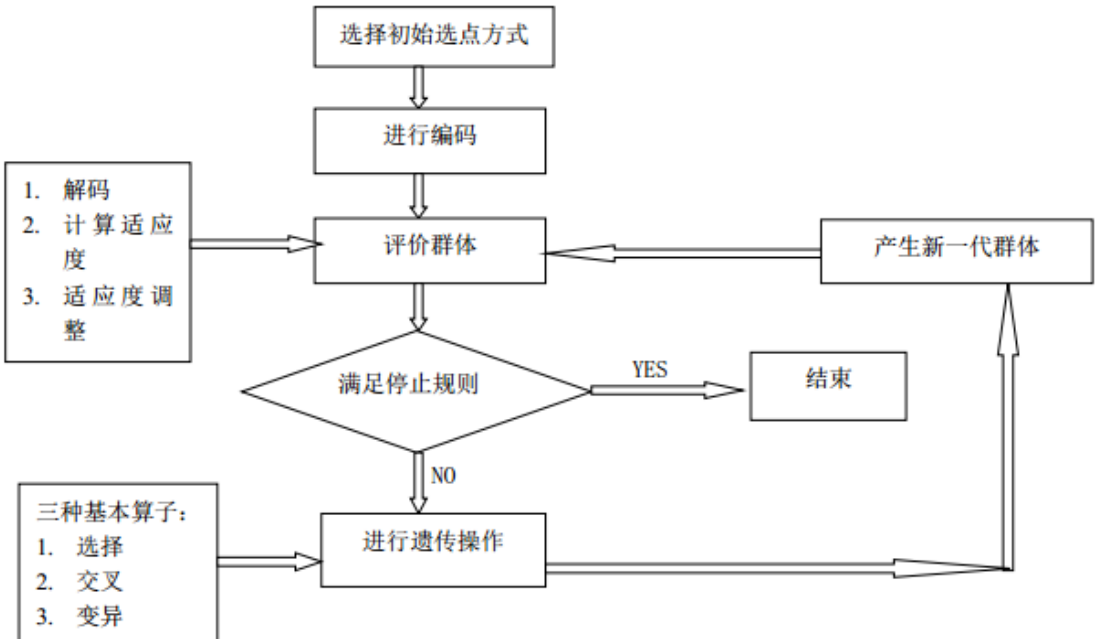

图 3　　遗传算法流程图

4.2 借助 MATLAB 对遗传算法在本问题的运用

1. 编码

采用二进制编码，即用一个二进制串来表示一个个体。

在本问题中，方法如下：

1.确定方法实现

（1）读入老师提供的 400 组数据。

（2）所要选择的位点用一串 51 位二进制数字表示（即基因型），对应位置为 1 表示该位置的数据在拟合中需要取到，为 0 表示该位置额数据在不参与拟合。如：100100110……1 这串数字，表示：所读入的数据中，每行的第 1、4、7、8……51 个位置（就是 x 为 5.0, 5.3, 5.6, 5.7 …… 10.0 的点）的数据要参与拟合。另外规定第一个和最后一个数据必须取到。总共取 100 个这样的二进制数来表示 100 个个体。

（3）用遗传算法作为大循环。

（4）进行 100 次循环后得到最终剩下的点，作为最优解。

2.遗传算法实现

（1）随机生成初始基因型（createPoint）

由于用二进制数表示基因型的遗传算法在基因重组的过程中无法确保所取点数（即基因型中 1 的个数）为定值，故令所有个体初始基因型中 1 的个数都相同并没有很大的意义。所以在生成初始基因型时我们设置了 0.2 的概率，每个个体的每个基因型有 0.2 的概率是 1。从而产生了初始基因型。

（2）成本计算（cost）

每种基因型对应一种校准方案，每种校准方案对应一个成本，成本计算规则如下：

（把成本计算规则放在这里）

通过利用每个基因型对应点进行相应的拟合（三次多项式拟合或插值拟合）得到对应的函数，再利用上面的成本计算公式算得每个基因型对应的成本。其中 errorcost 用于计算所有样本个体的定标成本。

另：由于交叉过程和初始基因产生的随机性，可能产生某些基因型中 1 的个数小于 4

的情况，此时无法正确拟合出所需要的函数（三次多项式拟合至少需要 4 个数据），故需要检查这种情况并予以更正，本例采用加点的方式来更正。即基因中 1 的个数不够时，在随机位置补上 1，使基因型中 1 的个数至少为 4 个，对应代码中 addPoint 函数。

（3）交叉的实现（select、newPoint）

（a）交叉的实现第一步就是要选择所要进行交叉的父辈基因型，本例采用 select 函数来确定。为更加真实模拟自然界中交叉过程，也更符合我们的认知，我们采用双随机的方式进行父辈的选择。就是所要参加交叉过程的双亲基因型都通过随机过程来产生，并把结果记录在一个 100x2 的数组中，每行的两个数据分别表示双亲的基因型（就是在用于记录基因型的数组 point 中的位置）。允许双亲的基因型来自同个位置。

（b）之后进行交叉（newPoint）。运用随机数选择一个基因断裂的位点（位点 position 取值为 1-50，数字表示在每个基因型的第 position 个基因后断裂）。双亲其中一个提供第 1 到第 position 个位点的基因，另一个提供第 position+1 到第 51 个位点的基因。此时产生新的一代。

（4）变异的实现（mutate）

变异就是指将基因中 0、1 进行互换。首先我们给定基因变异的概率，因为变异概率只是提供多样化的选择，不能因为变异而改变基因整体的优化性，因此变异的概率应该较小，定为 0.01。由于变异不一大规模发生，故我们假定，若一个基因型产生变异，则只有其中的一个基因位点产生变异。另外我们规定第一个和最后一个位点不允许产生变异。

4.3 在 MATLAB 运行结果及其分析

1 [1 5 16 24 37 41 47 51] 114.289
2 [1 13 23 24 31 33 41 51] 116.968
3 [1 9 17 21 25 31 37 43 51] 113.882
4 [1 13 19 24 28 30 44 51] 115.4188
5 [1 8 22 34 43 51] 102.736
6 [1 11 19 30 37 38 44 51] 111.7243
7 [1 4 16 24 29 36 49 51] 109.7978
8 [1 8 24 38 44 51] 113.6813
9 [1 10 19 30 39 43 51] 103.193
10 [1 10 20 24 32 38 44 51] 104.25
11 [1 8 19 25 29 36 48 51] 106.164
12 [1 4 18 24 30 36 44 51] 105.3353
13 [1 8 19 25 36 44 51] 99.568
14 [1 10 20 24 30 36 44 51] 104.092
15 [1 13 20 24 34 44 51] 103.7545
16 [1 8 19 24 32 44 51] 98.4185
17 [1 10 19 24 32 44 51] 98.7618
18 [1 10 19 24 32 44 51] 98.7618
19 [1 10 19 24 32 44 51] 98.7618
20 [1 10 18 24 30 43 51] 100.9315
21 [1 10 20 24 32 44 51] 98.8205
22 [1 10 18 29 32 44 51] 100.4648
23 [1 10 19 29 32 44 51] 99.3012
24 [1 9 18 29 32 44 51] 100.0203

25	[1	9	18	29	32	44	51	]	100.0203
26	[1	10	19	24	30	44	51	]	101.6708
27	[1	6	13	24	34	44	51	]	102.8298
28	[1	9	18	24	29	44	51	]	103.6688
29	[1	9	18	24	29	44	51	]	103.6688
30	[1	9	18	24	34	44	51	]	98.6363
31	[1	6	16	24	30	44	51	]	102.1555
32	[1	9	19	29	36	43	51	]	98.7055
33	[1	9	19	29	36	44	51	]	98.0837
34	[1	9	19	29	36	44	51	]	98.0837
35	[1	9	19	29	36	44	51	]	98.0837
36	[1	4	18	24	32	44	51	]	100.5075
37	[1	9	18	25	30	44	51	]	100.3815
38	[1	9	18	25	30	43	51	]	99.5007
39	[1	9	18	25	30	43	51	]	99.5007
40	[1	9	18	25	29	43	51	]	101.3178
41	[1	9	19	25	36	44	51	]	99.4188
42	[1	9	19	25	36	44	51	]	99.4188
43	[1	9	19	25	36	44	51	]	99.4188
44	[1	9	19	25	36	44	51	]	99.4188
45	[1	9	19	25	36	44	51	]	99.4188
46	[1	9	19	25	34	44	51	]	97.2703
47	[1	9	19	29	36	44	51	]	98.0837
48	[1	9	19	29	36	44	51	]	98.0837
49	[1	9	19	29	36	43	51	]	98.7055
50	[1	9	19	29	36	44	51	]	98.0837
51	[1	9	19	25	36	44	51	]	99.4188
52	[1	9	19	25	36	43	51	]	99.8095
53	[1	9	19	30	44	51		]	101.3105
54	[1	9	20	25	36	44	51	]	99.517
55	[1	9	20	29	36	44	51	]	97.5585
56	[1	9	20	29	36	43	51	]	98.1887
57	[1	9	20	29	36	43	51	]	98.1887
58	[1	9	20	29	36	43	51	]	98.1887
59	[1	9	20	29	38	43	51	]	99.5332
60	[1	9	19	29	36	43	51	]	98.0837
62	[1	9	20	29	36	43	51	]	98.1887
63	[1	9	20	29	36	44	51	]	97.5585
64	[1	9	20	29	36	44	51	]	97.5585
65	[1	9	20	29	36	44	51	]	97.5585
66	[1	9	19	29	36	44	51	]	98.0837
67	[1	9	19	29	36	44	51	]	98.0837
68	[1	9	19	29	36	44	51	]	98.0837
69	[1	9	19	29	36	44	51	]	98.0837

70 [1 9 19 29 36 44 51] 98.0837
71 [1 9 19 29 36 43 51] 98.7055
72 [1 9 20 29 36 44 51] 97.5585
73 [1 9 20 29 36 44 51] 97.5585
74 [1 9 20 29 36 43 51] 98.1887
75 [1 9 19 29 36 44 51] 98.0837
76 [1 9 19 29 36 44 51] 98.0837
77 [1 9 19 24 36 44 51] 101.636
78 [1 9 19 30 36 44 51] 100.3243
79 [1 9 20 30 36 44 51] 99.182
80 [1 9 20 30 36 44 51] 99.182
81 [1 9 20 30 36 43 51] 99.8855
82 [1 9 20 29 36 44 51] 97.5585
83 [1 9 20 29 36 44 51] 97.5585
84 [1 9 20 29 36 44 51] 97.5585
85 [1 9 20 29 36 44 51] 97.5585
86 [1 9 20 29 36 44 51] 97.5585
87 [1 9 19 29 36 44 51] 98.0837
88 [1 4 20 29 36 44 51] 101.4778
89 [1 9 20 30 36 44 51] 99.182
90 [1 9 20 29 36 43 51] 98.1887
91 [1 9 20 29 36 44 51] 97.5585
92 [1 9 20 29 36 44 51] 97.5585
93 [1 9 20 29 36 44 51] 97.5585
94 [1 9 20 29 36 44 51] 97.5585
95 [1 9 20 29 36 44 51] 97.5585
96 [1 9 20 29 36 44 51] 97.5585
97 [1 9 20 29 36 44 51] 97.5585
98 [1 9 20 29 36 44 51] 97.5585
99 [1 9 20 29 36 44 51] 97.5585
100 [1 9 20 29 36 44 51] 97.5585

结果分析：由运行结果可以看出，当运行到 91 时最后的成本就基本保持不变了。整个运行结果可以看出，所取的点数的是差不多的，到了 7 个点左右维持稳定。所取的点数个数的下降也是遗传算法选择的结果。由此可见，第一代基因型中所取的基因个数的多少并不重要，因为通过足够多代的遗传总可以得到一个较好的结果。最终我们得出，所取点为[1,9,20,29,36,44,51]时，得到的结果的成本较低，为 97.5585

5 方案总结

我们组本来采用十进制遗传算法，后经过改进改为二进制的遗传算法。其中十进制算法总体思路和前面所说的二进制的遗传算法差不多，最大区别在于：十进制的遗传算法中各个基因型中的 1 个数可以保持不变（就是方案中所选择用于拟合的点的个数不变，本例中我们选取拟合点的个数为 10），具体表现在一下过程中：

（1） 在产生初始基因型（createPoint）后基因型用一个 10 位数组保存而非一个 51 位二

进制数

（2） 计算成本时不用 addPoint 函数，因为选取的拟合点个数不变，故可以确保一定大于 3 个。

（3） 变异的实现不再是 1 和 0 转换，而是把一个用于拟合的点替换成另一个原先不用于拟合的点。

该方法和二进制比的缺点如下：

（1） 由于固定了用于拟合的点的个数，因而无法比较点的个数不同对成本的影响（当然有特殊需要时要求固定拟合点的个数时并无该缺点）

（2） 随着模拟的进行，可能会产生某个点在一个基因型中产生多次的情况。我们曾经考虑过修改代码来排除此种情况，然而此时就相当于强制基因产生变异，在模拟次数较大的时候很可能会使变异概率大大提高。故权衡之下还是放弃了该处改动，带来的后果便是程序运行过程中可能所包含的重复点的个数实在太多，而使得独立点的个数小于 4 个，无法进行正确拟合。

（3） 本质上来讲二进制和十进制的遗传算法是相同的，而二进制的遗传算法的代码相比于十进制的遗传算法代码来说更容易书写。

（4） 由运行结果可以看出，当所取的点在 5 个左右的时候，成本会较少，若用十进制的遗传算法无法得到成本这么小的结果。

本次探究，首先是使用 MATLAB 编程的方法研究统计推断在模数、数模转换系统中的应用，探究针对某一检测模块中输入输出特性呈明显的非线性关系，遗传算法很好地例证了这个问题，认识到理论方法在实践探究中的运用，课程中的方法在实际生产中有极大的运用价值和成本价值。另外，也让我们通过这个问题更加深切地体会到了工程中 MATLAB 的使用，收益颇大。

6 致谢

感谢课程老师在讲座中的细心讲解，感谢同学在我们困难时的帮助。

7 参考文献

[1]上海交通大学电子工程系.统计推断在数模转换系统中的应用课程讲义[EB/OL].ftp://202.120.39.248

[2]2014-2015 秋季统计推断在数模转换系统中的应用第 63 组彭诗奇组课程设计报告

[3]百度文库《MATLAB 数据拟合使用教程》

[4]百度百科.遗传算法

http://baike.baidu.com/link?url=ZY6JWAcyXeqVobKE1gvD6F0rbWl8PprRCN8ATpHIKeOxx4soZ4S1eD86h60w1lFRCA-ZvfO-hazxhasij-iWzq#3

8 附录

十进制多项式遗传算法：

```
data0=csvread('20150915dataform.csv'); %读取数据
popSize=100;                           %种群中个体数目
mutationRate=0.01;                     %变异概率
generationSize=100;                    %遗传代数
```

```
data=zero(400,51);
data(1:400,:)=data0(2:2:800,:);
point=createPoint(popSize);                %种群中个体的表现型(用于拟合的点的位置)
fori=1:generationSize
display(i);
display('position ')
display(point);
    cost=cost(popSize);                    %计算种群中个体对应的费用
display('the cost is ');
display(min(cost));
    select=select(cost,popSize);               %选择要交配的个体
    point=newPoint(select,point,popSize);     %交配后的结果
    point=mutate(point,popSize,mutationRate);%变异后的结果
end

function out=createPoint(popSize)                    %产生第一代个体的函数
tem=zeros(popSize,51);
fori=1:popSize
tem(i,1)=1;
tem(i,51)=1;
    j=1;
while (j<=8)
        g=round(49*rand()+0.5);
if tem(i,g+1)==0
tem(i,g+1)=1;
            j=j+1;
end
end
out=find(tem(i,1:51));
end
```

```
end

function out=cost(point,data,popSize)                %平均花费的计算函数
out=zeros(popSize,1);
xx=5:0.1:10
fori=1:popSize
position=point(i,:);
    x=5+0.1*position-0,1;
    y=data(:,position);
    f=polyfit(x,y,3)
difference=polyval(f,xx)-data;
end
out(i)=120+errorCost(difference)/400;
end

function out=errorCost(difference)              %计算每个原件因为误差产生的花费
dif=abs(diffence);
dif0=sum(sum(dif<=0.4));
dif1=sum(sum(dif<=0.6))-dif0;
dif2=sum(sum(dif<=0.8))-dif0-dif1;
dif3=sum(sum(dif<=1))-dif0-dif1-dif2;
dif4=sum(sum(dif<=2))-dif0-dif1-dif2-dif3;
dif5=sum(sum(dif<=3))-dif0-dif1-dif2-dif3-dif4;
dif6=sum(sum(dif<=5))-dif0-dif1-dif2-dif3-dif4-dif5;
dif7=sum(sum(dif))-dif0-dif1-dif2-dif3-dif4-dif5-dif6;
out=0.1*dif1+0.7*dif2+0.9*dif3+1.5*dif4+6*dif5+12*dif6+25*dif7
end


function out=select(cost,popSize)                                %选择交配的两个个体
```

```
out=zeros(popSize,2);
cost=1./cost;
total=sum(cost);
pro=cost/total;
new=zeros(popSize+1,1);
for j=2:popSize+1
new(j)=sum(pro(1:j-1));
end
fori=1:popSize
    rand1=rand();
    rand2=rand();
    j=1;
    k=1;
while (rand1>new(j+1))
        j=j+1;
end
while (rand2>new(k+1))
        k=k+1;
end
out(i,1)=j;
out(i,2)=k;
end
end

function out=newPoint(select,point,popSize)                    %产生新个体
out=zeros(popSize,10);
fori=1:popSize
position=round(9*rand()+0.5);
    out(i,1:position)=point(select(i,1),1:position);
out(i,position+1:10)=point(select(i,2),position+1:10);
```

```
end
end


function point=mutate(point,popSize,mutationRate)            %遗传
out=point;
fori=1:popSize
if rand()<mutationRate
position=round(8*rand()+0.5)+1;
replace=round(49*rand()+0.5)+1;
out(i,position)=replace;
length=length(find(out(i,:))
while (length~=1)
replace=round(49*rand()+0.5)+1;
out(i,position)=replace;
length=length(find(out(i,:))
end
end
end
```

二进制多项式拟合遗传算法　（注释与十进制相同）

```
data0=csvread('20150915dataform.csv');
popSize=100;
mutationRate=0.01;
generationSize=100;
data=zeros(400,51);
data(1:400,:)=data0(2:2:800,:);
```

```
point=createPoint1(popSize);
for i=1:generationSize
    display(i);
    cost0=cost1(point,data,popSize);

    for j=1:popSize
        if cost0(j)==min(cost0)
            display(find(point(j,:)));
            break
        end
    end
    display('the cost is ');
    display(min(cost0));
    s=select1(cost0,popSize);
    point=newPoint1(s,point,popSize);
    point=mutate1(point,popSize,mutationRate);
end

function out=createPoint1(popSize)

out=zeros(popSize,51);
for i=1:popSize
    out(i,1)=1;
    out(i,51)=1;
    for j=1:51
        if rand()<0.2
        out(i,j)=1;
        end
    end
end
```

```
end


function out=cost1(point,data,popSize)
out=zeros(popSize,1);
xx=5:0.1:10;
for i=1:popSize
    tmp=point(i,:);
    su=sum(tmp>0);
        if su<=3
            tmp=addPoint(tmp,su);
            su=4;
        end
    point(i,:)=tmp;
    position=find(tmp);
    x=5+0.1*position-0.1;
    for j=1:1:400

        y=zeros(1,su);

        for k=1:1:su
            c=position(k);
            y(k)=data(j,c);
        end
        f=polyfit(x,y,3);
        difference(j,:)=polyval(f,xx)-data(j,:);

    end
    out(i)=12*sum(position>0)+errorCost1(difference)./400;
end
end
```

```
function out=errorCost1(difference)
dif=abs(difference);
dif0=sum(sum(dif<=0.4));
dif1=sum(sum(dif<=0.6))-dif0;
dif2=sum(sum(dif<=0.8))-dif0-dif1;
dif3=sum(sum(dif<=1))-dif0-dif1-dif2;
dif4=sum(sum(dif<=2))-dif0-dif1-dif2-dif3;
dif5=sum(sum(dif<=3))-dif0-dif1-dif2-dif3-dif4;
dif6=sum(sum(dif<=5))-dif0-dif1-dif2-dif3-dif4-dif5;
dif7=sum(sum(dif>=0))-dif0-dif1-dif2-dif3-dif4-dif5-dif6;
out=0.1*dif1+0.7*dif2+0.9*dif3+1.5*dif4+6*dif5+12*dif6+25*dif7;
end

function out=select1(cost0,popSize)
out=zeros(popSize,2);
cost0=abs(1./cost0);
total=sum(cost0);
pro=cost0/total;
new=zeros(popSize+1,1);
for j=2:popSize+1
    new(j)=sum(pro(1:j-1));
end
for i=1:popSize
    rand1=rand();
    rand2=rand();
    j=1;
    k=1;
    while (rand1>new(j+1))
        j=j+1;
```

```
    end
    while (rand2>new(k+1))
        k=k+1;
    end
    out(i,1)=j;
    out(i,2)=k;
end
end

function out=newPoint1(s,point,popSize)
out=zeros(popSize,51);
for i=1:popSize
    position=round(50*rand()+0.5);
    out(i,1:position)=point(s(i,1),1:position);
    out(i,position+1:51)=point(s(i,2),position+1:51);
end
end

unction point=mutate1(point,popSize,mutationRate)
out=point;
for i=1:popSize
    flag=0;
    if rand()<mutationRate
        position=round(49*rand()+0.5)+1;
        if out(i,position)==1
            out(i,position)=0;
        else out(i,position)=1;
        end
    end
end
```

二进制插值拟合（注释与十进制相同）

```
function out=cost2(point,data,popSize)
out=zeros(popSize,1);
xx=5:0.1:10;
for i=1:popSize
    tmp=point(i,:);
    su=sum(tmp>0);
        if su<=3
            tmp=addPoint2(tmp,su);
            su=4;
        end
    point(i,:)=tmp;
    position=find(tmp);
    x=5+0.1*position-0.1;
    for j=1:1:400

        y=zeros(1,su);

        for k=1:1:su
            c=position(k);
            y(k)=data(j,c);
        end
        f=spline(x,y);
        difference(j,:)=ppval(f,xx)-data(j,:);

    end
    out(i)=12*sum(position>0)+errorCost2(difference)./400;
end
end
```