机器学习
Machine Learning

鲍军鹏

2019年2月（V1.2）

西安交通大学计算机学院
Email: baojp@xjtu.edu.cn

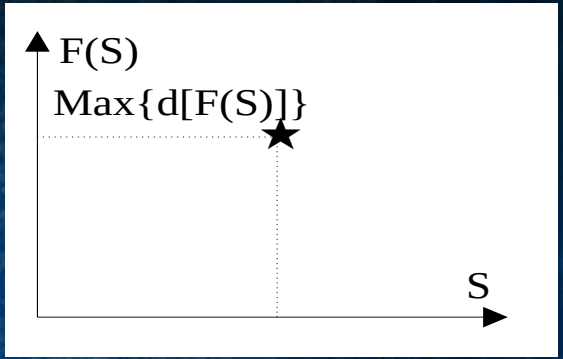
第六章 智能优化方法

- ★ 什么是智能优化方法
- ★ 遗传算法
- ★ 蚁群算法
- ★ 粒子群算法
- ★ 自适应协方差矩阵进化策略

优化问题

★求取极值的问题

$$\begin{cases} min/max\{F(x)\} \\ subject\ to\ x \in \Omega \end{cases}$$



★传统的方法

- 无约束优化问题
  - 求解导数为0的点
  - 最速下降法（梯度下降法）、牛顿法、共轭梯度法、单纯形法
- 有约束优化问题
  - 想办法将其转换为无约束优化问题
  - Monte Carlo法、线性规划法、二次规划法

进化计算

- ★Evolutionary Computation
  - 进化计算是一类模拟生物进化、自然选择过程与机制求解问题的自组织、自适应人工智能技术。
- ★也称为
  - 仿生计算（ Bio-inspired Computation）
  - 群体计算（Swarm Computation）
- ★典型代表
  - 遗传算法（Genetic Algorithm）
  - 蚁群算法（Ant Colony Optimization）
  - 粒子群算法（Particle Swarm Optimization）

进化计算

★ 基本思想

- 如果把待解决的问题理解为对某个目标函数的全局优化，则进化计算就是建立在模拟生物进化过程基础上的随机搜索优化技术。

★ 特点

- 进化计算具有隐含并行性。
- 进化计算不需对待解问题进行完整的解析分析，只需要对求解目标进行合理表示，然后就可以自动获得较优解。
- 进化计算原则上适用于任意函数类。

进化计算的组成部分

- 进化计算方法一般包括
  - 编码策略、适应函数、变异算子、交叉算子、选择算子等几部分。
  - 编码策略就是决定如何用一个字符串来表示一个个体。
  - 适应函数就是从编码字符串到表现型的映射，也就是评价一个具体编码串是否优劣的函数。适应函数的值称为适应度。
  - 变异（即突变）算子随机改变一个编码串中的某几位，得到另一个新的编码串。这个算子模拟生物基因突变现象。
  - 交叉算子把两个编码串进行混合得到新的编码串。
  - 选择算子从一个群体中（即多个编码串）中取出多个较优个体，用于繁衍下一代。

进化计算的一般过程

★进化计算的一般过程如下：

1. 确定编码策略，对待解决问题进行编码；
2. 随机生成n个不同编码串构成初始种群$X(0)=\{ x_1, x_2, \ldots x_n \}$；
3. 计算当前群体$X(t)$中每个个体x_i的适应度$F(x_i)$；
4. 应用选择算子产生中间代$Xr(t)$；
5. 对$Xr(t)$应用其它进化算子，产生新一代群体$X(t+1)$；
6. 进化代数增一，即$t=t+1$；
7. 如果不满足终止条件则转至第3步，否则结束。

第六章 智能优化方法

- ★ 什么是智能优化方法
- ★ 遗传算法
- ★ 蚁群算法
- ★ 粒子群算法
- ★ 自适应协方差矩阵进化策略

遗传算法

★Genetic Algorithm

- ★由美国Michigan大学J.H.Holland于60年代提出，
- ★模仿生物遗传学和自然选择机理，通过人工方式所构造的一类优化算法，
- ★对生物进化过程进行的一种数学仿真，
- ★进化计算的最重要的形式。

★进化计算和遗传算法借鉴了生物科学中的某些知识

- ★生物通过个体间的选择、交叉、变异来适应大自然环境。
- ★生物染色体用数学方式或计算机方式来体现就是一串数码，仍叫染色体，也叫个体。

遗传算法

★基本思想

- 把问题的可行解表示成“染色体”，即以一定方式编码的字符串。
- 在执行遗传算法之前，给出一组“染色体”，称为初始种群。
- 然后按适者生存的原则，从中选择出比较适应环境的“染色体”进行复制，再通过交叉，变异过程产生更适应环境的新一代“染色体”种群。
- 这样，经过一代一代地进化，最后可找到一个最适应环境的“染色体”，它就是问题的最优解。

★遗传算法是一种最优化方法

- 它通过进化和遗传机理，从给出的原始种群中，不断进化产生新的染色体，最终求得问题的最优解。

基本概念

- 串(String)
  - 它是个体(Individual)的形式，在算法中为二进制串或者其它字符串，对应于遗传学中的染色体(Chromosome)。
- 群体(Population)和群体大小(Population Size)
  - 个体的集合称为群体（种群），串是群体的元素。
  - 在群体中个体的数量称为群体的大小。
- 基因(Gene)
  - 基因是串中的元素。例如有一个串S＝1011，则其中的1，0，1，1这4个元素分别称为基因。
- 基因位置(Gene Position)
  - 一个基因在串中的位置称为基因位置，有时也简称基因位。
  - 基因位置由串的左向右计算，例如在串S＝1101中，0的基因位置是3。
- 适应度(Fitness)
  - 表示某一个体对于环境的适应程度，用一个函数表示。

遗传算法的组成部分

遗传算法由四个部分组成：

编码机制、适应度函数、控制参数、遗传算子

（1）编码机制(encoding mechanism)

根据具体问题的具体特点，将问题的可行解用染色体表示。染色体对应的字符串可以是二进制字符串，也可以是其它的字符串。

（2）适应度函数(fitness function)

- 优胜劣败是自然进化的原则。优、劣要有标准。在GA,用适应度函数f(x)描述每一个体x的适应程度。对优化问题,适应度函数往往就是优化目标函数。
- 在遗传算法的执行过程中,每一代都有多个个体(染色体)同时存在。这些染色体中哪个保留(生存)、哪个淘汰(死亡),是根据其适应度函数f(x)的值来决定的,适应性强的有更多的机会保留下来。

（3）控制参数(control parameters)

- 在GA算法中,需适当确定某些参数的值以提高选优的效果。这些参数包含：
  - 字符串所含字符的个数,即串长。这一长度为常数, 记为L。
  - 每一代群体的大小, 也称群体的容量,记为n。
  - 交换率(crossover rate),即施行交换算子的概率,记为Pc。
  - 突变率(mutation rate),即施行突变算子的概率,记为Pm。
- 系统参数对算法的收敛速度及结果有很大的影响，应视具体问题选取不同的值。

（4）遗传算子(genetic operator)

GA中最常用的算子有如下几种：

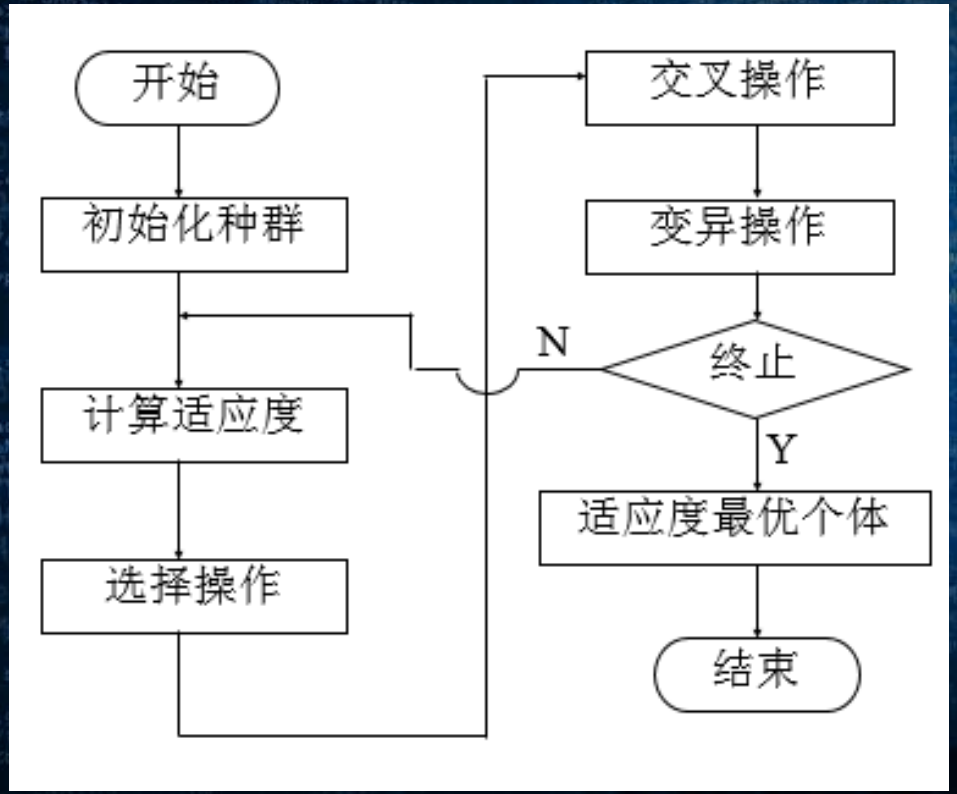
- 选择算子(Selection/Reproduction): 选择算子从群体中按某一概率选择个体，某个体xi被选中的概率Pi与其适应度值f(xi)成正比。最通常的实现方法是轮盘赌(roulette wheel)模型。
- 交叉算子(Crossover): 交叉算子将被选中的两个个体的基因链按概率Pc进行交叉，生成两个新的个体，交叉位置是随机的。其中Pc是一个系统参数。
- 变异算子(Mutation): 变异算子将个体基因链的某位按概率Pm进行变异，对二值基因链(0,1编码)来说即是取反。

遗传算法的求解步骤

一般过程：

(1) 对待解决问题进行编码；
(2) 随机产生一个初始群体X(0):=(x1, x2, … xn)；
(3) 对当前群体X(t)中每个个体xi计算其适应度f(xi)，适应度表示了该个体的性能好坏；
(4) 应用选择算子产生中间代Y(t)；
(5) 对Y(t)应用交叉和变异算子，产生新一代群体X(t+1)
(6) t:=t+1；如果不满足终止条件继续(3)。

基本遗传算法的框图如下：



操作方法

(1) 初始化

- 选择一个群体，即选择一个串或个体的集合。这个初始的群体也就是问题可行解的集合。
- 通常以随机方法产生串或个体的集合。

(2) 选择

- 根据适者生存原则选择下一代的个体。在选择时，以适应度为选择原则。适应度准则体现了适者生存，不适应者淘汰的自然法则。
- 给定适应度函数f，则f(bi)称为个体bi的适应度。通常选中bi进入下一代的概率为

$$\frac{f(b_i)}{\sum_{j=1}^{n} f(b_j)}$$

(3) 交叉

- 在被选中进入下一代的个体中，随机选择两个个体及交叉位置，按交叉概率Pc在选中的位置实行交换。
- 目的在于产生新的基因组合，也即产生新的个体。交叉时，可实行单点交叉或多点交叉。

例如有个体

S1=100101

S2=010111

选择它们的左边3位进行交叉操作，则有

S1=010101

S2=100111

(4) 变异

- 根据生物遗传中基因变异的原理，以变异概率P_m对被选中个体的某些位执行变异。
- 对二进制串，变异就是在串的某个位置求反，即把1变为0，把0变为1。
- 变异概率P_m与生物界变异概率极小的情况一致，一般取较小的值，如0.0001-0.1左右。

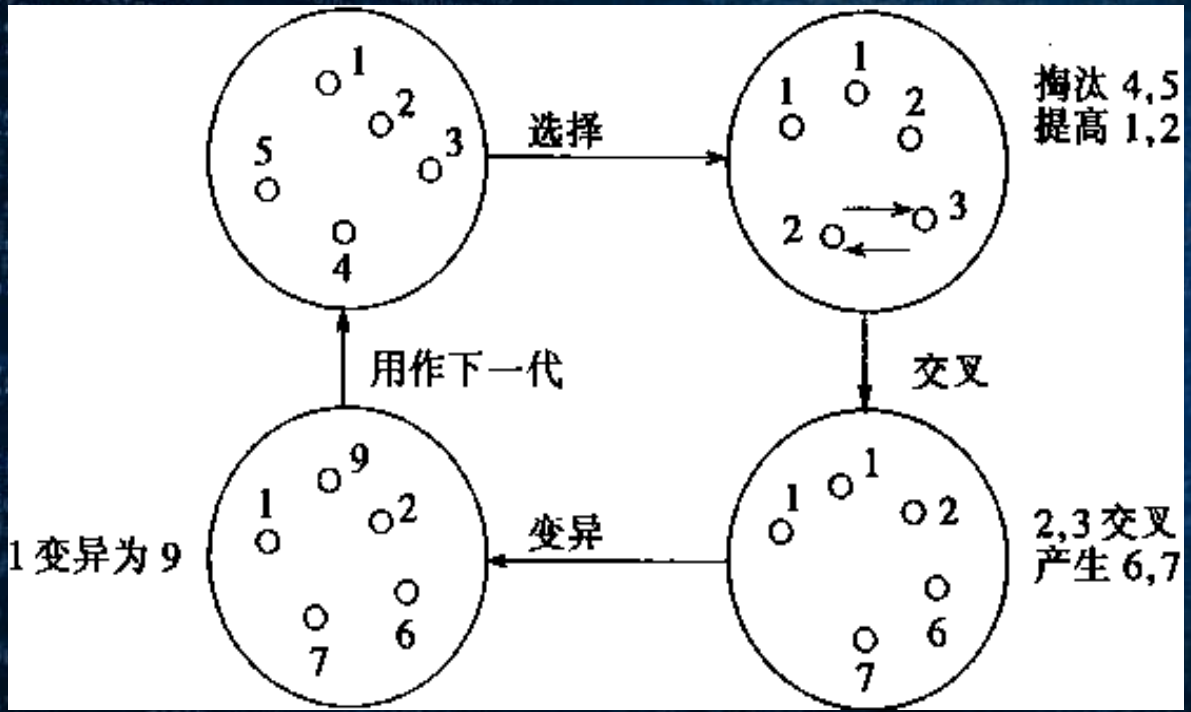
例如有个体S = 101011。对其的第1，4位置的基因进行变异，则有

S'=001111

(5) 终止条件（全局最优收敛）

- 当最优个体的适应度达到给定的阈值，或者最优个体的适应度和群体平均适应度不再上升时，则算法的迭代过程收敛，算法结束。
- 否则，用经过选择、交叉、变异所得到的新一代群体取代上一代群体，继续进行进化。

遗传算法原理



遗传算法举例

例1　求函数f(x)=x2的最大值，变量x为0~31之间的整数。

解：

- 为用GA解此问题, 将变量x的取值以二进位数表示, 从而得到一种自然的编码;
- 每一个体均为长度是5的二进制位串。初始群体的容量取4。随机选取4个个体组成第一代群体，即初始群体。
- 具体操作可通过掷硬币确定。
- 例如,将一枚硬币连续掷20次,从而得到4个长度为5的二进制字符串,如(01101)、(11000)、(01000)、(10011)。

- GA按适应度大小选择个体，从而决定第一代群体中哪个个体能被保留。结果如下表所示。

表1　第一代群体的选择

串NO.	初始群体 (随机生成)	x	$f(x)=x^2$	$\frac{f}{\sum f}$	$\frac{nf}{\sum f}$	实际生存数 (由轮盘决定)
1	01101	13	169	0.144	0.576	1
2	11000	24	576	0.492	1.968	2
3	01000	8	64	0.055	0.220	0
4	10011	19	361	0.309	1.236	1
和			1170	1.000	4.000	
平均			293	0.250	1.000	
max			576	0.492	1.968	

- 设交叉概率Pc取为1，即肯定施行交换算子。

同样,可通过掷硬币的方法将前面复制出的4个串配成两对,并在随机确定的交叉点进行交换,结果如表2。

表2 交叉的进行

串NO.	种群	配对 (随机选择)	交叉点 (随机选择)	新种群	x	$f(x)=x^2$
1	01101	2	4	01100	12	144
2	11000	1	4	11001	25	625
3	11000	4	2	11011	27	729
4	10011	3	2	10000	16	256
和 平均 max						1754 439 729

- 设突变概率Pm取为0，因此本例没有基因突变。
- 于是,经过选择、交换完成了一代的遗传，得到了一个新的群体。
- 事实证明,第二代群体的质量有了明显的提高,平均适应度由293增加为439,最大适应度由576增加到729。

遗传算法的特点（1）

(1) 遗传算法从问题的解集中开始嫂索，是群体搜索，而不是从单个解开始。

这是遗传算法与传统优化算法的极大区别。传统优化算法是从单个初始值迭代求最优解的；容易误入局部最优解。遗传算法从串集开始搜索，复盖面大，利于全局择优。

(2) 遗传算法求解时使用特定问题的信息极少，容易形成通用算法程序。

由于遗传算法使用适应值这一信息进行搜索，并不需要目标函数的导数等与问题直接相关的信息。遗传算法只需适应值和串编码等通用信息，故可处理很多问题。

遗传算法的特点（2）

(3) 遗传算法有极强的容错能力。

遗传算法的初始串集本身就带有大量与最优解甚远的信息；通过选择、交叉、变异操作能迅速排除与最优解相差极大的串；这是一个强烈的滤波过程；并且是一个并行滤波机制。故而，遗传算法有很高的容错能力。

(4) 遗传算法中的选择、交叉和变异都是随机操作，执行概率转移准则，而不是确定的精确规则。

(5) 遗传算法具有隐含的并行性。

第六章 智能优化方法

- ★ 什么是智能优化方法
- ★ 遗传算法
- ★ 蚁群算法
- ★ 粒子群算法
- ★ 自适应协方差矩阵进化策略

蚁群算法

蚁群算法(ant colony algorithm)是一种模拟进化算法。是由意大利学者M. Dorigo等人对自然界中真实蚁群集体行为研究基础上，于1991年首先提出的。

蚁群算法模拟了自然蚂蚁的协作过程，用一定数目的蚂蚁共同求解，用蚂蚁的移动线路表示所求问题的可行解集，通过正反馈、分布式协作和隐并行性找最优解。

蚁群算法已成功应用于求解TSP 问题、任务分配问题、调度问题等组合优化问题，并取得了较好的实验结果。受其影响，蚁群系统模型逐渐引起了其它研究者的注意，并用该算法来解决一些实际问题。

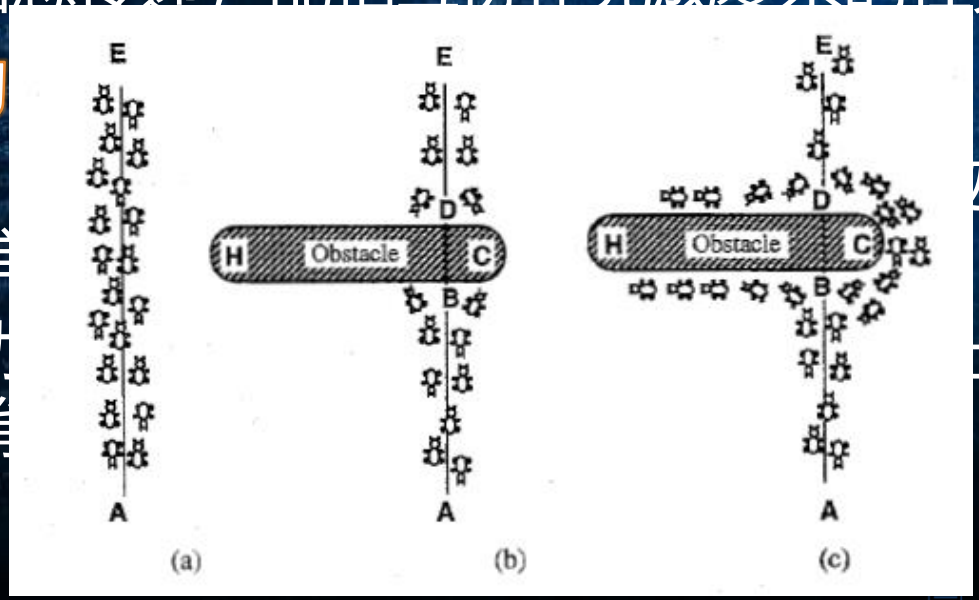
蚁群算法原理（1）

★蚂蚁

- 一种群居昆虫，个体行为极其简单，而群体行为却相当复杂。
- 蚂蚁个体之间通过外激素(pheromone，也称为信息素) 进行信息传递，能相互协作完成复杂的任务。

★蚁群的能力

- 协作能力：
- 自适应能力



蚁群算法原理（2）

★信息素

- 蚂蚁碰到一个还未走过路口时就随机选择一条路径前行，并释放出信息素。信息素浓度会随着时间衰减。

★正反馈机制

- 越短的路上积累的信息素越多，越长的路上积累的信息素越少。当后来的蚂蚁再次碰到这个路口时，选择信息素较多的路径的概率相对较大。这样便形成了一个正反馈机制。
- 最优路径上的信息素越来越多，而其他路径上的信息素则随着时间逐渐减少。最终整个蚁群会找出最优路径。

蚁群系统模型

★生物界中蚂蚁群体行为的显著特征

- 能察觉其它蚂蚁遗留的信息素；
- 能释放自己的信息素；
- 所遗留的信息素数量会随时间而逐步减少。

★蚁群算法的核心

- 选择机制：信息素越多的路径，被选中的概率越大；
- 更新机制：路径越短，信息素增加越快；
- 协作机制：个体之间通过信息素进行交流；
- 随机性：单个蚂蚁个体在周围没有信息素指引时按照概率随机选择方向。

蚁群算法的核心公式

★转移概率

- 蚂蚁k在t时刻从位置i移动到位置j的概率

$$p_{ij}^{k}(t)=\frac{\tau_{ij}(t)^{\alpha}\times\eta_{ij}(t)^{\beta}}{\sum_{s\in Jk(i)}\tau_{is}(t)^{\alpha}\times\eta_{is}(t)^{\beta}}$$

其中，

$s\in Jk(i)$表示从当前位置i能够到达的所有位置；

α和β分别表示信息素和启发式各自的权重；

η_{ij}表示启发式，或者能见度。例如：$\frac{1}{d_{ij}}$，即距离倒数；

τ_{ij}表示从位置i到位置j的信息素数量。

蚁群算法的核心公式

★信息素更新公式

$$\tau_{ij}(t+1)=\rho\times\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k}=\frac{Q}{L_k}$$

其中，

ρ表示信息素挥发系数，$0<\rho<1$；

m表示蚂蚁的数量；

Q表示信息素总量，为正常数，影响算法收敛速度；

L_k表示蚂蚁k从i到j走过路径的长度；

$\tau_{ij}(0)=C,\qquad \Delta\tau_{ij}(0)=0$。

蚁群算法的特点(1)

1）蚁群算法是一种自组织算法

- 算法开始时，单个蚂蚁无序地进行寻找解；
- 一段时间迭代之后，由于正反馈机制，蚂蚁群体趋向于接近最优解；
- 这是一个从无序到有序的过程。。

2）蚁群算法是一种本质上并行的算法

- 每只蚂蚁的搜索过程彼此独立，仅仅依靠信息素进行通信。

蚁群算法的特点(2)

3）蚁群算法是一种正反馈的算法

- 蚂蚁之所以可以找到最短路径，是依赖最短路径上的信息素堆积而成的，而信息素的堆积是一个正反馈过程。

4）蚁群算法具有较强的鲁棒性

- 蚁群算法的求解结果不依赖初始路线的选择，而且在搜索过程中不需要进行人工调整；
- 蚁群算法的参数数目小，设置简单。

蚁群算法存在的问题

★算法运算量较大，运行时间较长

- 由于蚂蚁个体运动过程的随机性，当群体规模设置较大时，很难在短时间内从杂乱无章的路径中找出一条较好路径。

★易收敛到局部较优解

- 搜索进行到一定程度后，所有蚂蚁发现的解完全一致。此时不能进一步搜索解空间，不利于发现全局最优解。

★信息素更新策略，路径搜索策略和最优解保留策略都带有经验性

- 求解结果具有较大的分散性

货郎担问题（TSP）

★货郎担/旅行商问题（Travelling Salesman Problem）

- 经典的组合优化难题，是NPC问题

★问题描述

- 从n个城市中的一个出发，
- 遍历其它所有城市然后回到出发城市，
- 每个城市必须走过并且只能走过一次，
- 整个环路的路径之和最小。
- 穷举对称型TSP时间复杂度为

$$O(\frac{1}{2}(n-1)!)$$

用蚁群算法解决TSP

★假设

- 将m只蚂蚁随机放入到n个城市中。
- d_{ij}表示城市i和城市j之间的距离，$i,j= 1,2,\cdots,n$
- $\tau_{ij}(t)$：表示t时刻在(i,j)连线上残留的信息素数量。初始时刻，各条路径上信息量相等，即$\tau_{ij}(0) = C$;
- α：表示路径上信息量的相对重要性($\alpha \geq 0$);
- η_{ij}：表示由城市i转移到城市j的启发式。可根据某种启发式算法具体确定。对于TSP问题一般可取$\eta_{ij}=1/d_{ij}$;
- β：表示启发式的相对重要性($\beta \geq 0$)。

$p_{ij}^{k}(t)$：表示在t时刻蚂蚁k由位置i转移到位置j的概率：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum_{s\in allowed_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}} & j\in allowed_k \\ 0 & \text{其它}\end{cases}$$

其中，$allowed_k$表示蚂蚁k下一步允许选择的城市集合。

人工蚁群系统需要记忆已经走过的城市序列。
用ta_k(k= 1, 2, …,m)记录蚂蚁k目前已走过的城市序列。

随着时间的推移，以前留下的信息素逐渐挥发，用1-ρ表示信息挥发的程度，ρ可理解为信息的持久性。经过n 个时刻，蚂蚁完成一次循环，各路径上信息素要根据下式作调整：

$$\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\Delta\tau_{ij}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

$\Delta\tau^{k}_{ij}$表示第k 只蚂蚁在本次循环中留在路径(i, j)上的信息量：

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/L_k & \text{若第}k\text{只蚂蚁在本次循环中经过}(i,j)\\ 0 & \text{其它}\end{cases}$$

其中Q为常数，L_k表示第k只蚂蚁在本次循环中走过的路径总长度。

采用蚁群算法进行求解TSP的主要步骤可叙述如下：

① nc←0 (nc 为迭代次数或搜索次数)；初始化τ_{ij}；将m个蚂蚁置于n 个顶点上；

② 将每个蚂蚁k的初始出发点置于该蚂蚁的ta_k中；

将每个蚂蚁k按概率$p_{ij}^k(t)$ 移至下一顶点j，并将顶点j置于ta_k 中，重复该过程直至每个蚂蚁遍历了所有的城市。

③ 计算各蚂蚁的目标函数值Z_k(k =1, …,m)；

记录当前的最好解；

④ 按更新方程修改信息素强度；

⑤ nc←nc+1；

⑥ 若nc < 预定的迭代次数，则转步骤②； 否则退出。

实验结果表明，当m大致等于n 时，效果较佳。

第六章 智能优化方法

- ★ 什么是智能优化方法
- ★ 遗传算法
- ★ 蚁群算法
- ★ 粒子群算法
- ★ 自适应协方差矩阵进化策略

粒子群算法

★群体智能

- 生物群体普遍能够表现出一定智能行为，即群体智能（Swarm Intelligence）。
- 群(Swarm)：某种具有交互作用的组织或智能体的集合。
- 在群体中，个体的结构很简单，而它们的群体行为却会相当复杂。个体行为和群行为之间存在着某种紧密联系。
- 个体行为构成和支配了群行为；同时，群行为又影响和改变个体的自身行为。个体之间的交互在构建群行为中起到重要作用。它帮助群体改善了对环境的经验知识。
- 对不同群的研究得到了不同算法。其中最引人注目的是对鸟群和蚁群的研究而建立的粒子群算法和蚁群算法。

粒子群算法

★Particle Swarm Optimization，PSO

- 由Kennedy和Eberhart在1995年提出。
- 该算法模拟鸟集群飞行觅食的行为，鸟之间通过集体的协作使群体达到最优目的。
- 群体智能（Swarm Intelligence）优化方法的典型代表之一。

粒子群算法

- ★ 源于对鸟类捕食行为的模拟
- ★ 设想场景：
  - 一群鸟在随机搜索食物。
  - 在这个区域里只有一块食物。
  - 所有的鸟都不知道食物在那里，但是它们知道当前的位置离食物还有多远。
  - 那么找到食物最简单有效的方法就是搜寻目前离食物最近鸟的周围区域。
  - 在搜寻过程中，每只鸟的位置变化以成功超越其它个体的社会心理意向为基础。
  - 因此一只鸟的搜寻行为受到其它鸟搜寻行为的影响。

粒子群算法

★粒子（Particle）

- PSO中每个优化问题的解都是搜索空间中的一只鸟，称之为“粒子”。
- 每个粒子都有一个由优化函数决定的适应值，
- 每个粒子还有一个速度决定其飞翔的方向和距离，
- 所有粒子都追随当前的最优粒子在解空间中搜索。

粒子群算法

★基本搜索过程

- PSO初始化为一群随机粒子，然后叠代找到最优解。
- 每一次叠代中，粒子跟踪两个"极值"来更新自己。
  - 一个极值是粒子本身所找到的最优解。这个解叫做个体极值pBest;
  - 一个极值是整个种群当前找到的最优解。这个极值是全局极值gBest。即，全局PSO算法。
  - 也可以不用整个种群而只用其中一部分近邻的最优位置。即，局部PSO算法。
- 粒子通过不断学习和更新，最终飞至空间中最优解所在的位置。

粒子群算法

假设在一个n维搜索空间中，有m个粒子组成一个群体，在某一时刻t：
第i个粒子的位置为一个n维向量：$Xi(t)=(x_{i1},x_{i2},\ldots,x_{in})$
第i个粒子的飞翔速度也是一个n维向量：$Vi(t)=(v_{i1},v_{i2},\ldots,v_{in})$
第i个粒子迄今为止搜索到的最优位置为：$Pi=(p_{i1},p_{i2},..,p_{in})$
整个粒子群迄今为止搜索到的最优位置为：$Pg=(p_{g1},p_{g2},..,p_{gn})$
目标函数f计算粒子的适应度$f(Xi(t))$，根据适应度大小衡量粒子的优劣。

Kennedy和Eberhart最早提出用下列公式对粒子更新：

其中，

$$\begin{cases} V_i(t+1)=V_i(t)+c_1r_1(P_i-X_i(t))+c_2r_2(Pg-X_i(t)) \\ X_i(t+1)=X_i(t)+V_i(t+1) \end{cases}$$

c_1和c_2是非负常数，称为学习因子；

r_1和r_2是介于[0,1]之间的随机数。

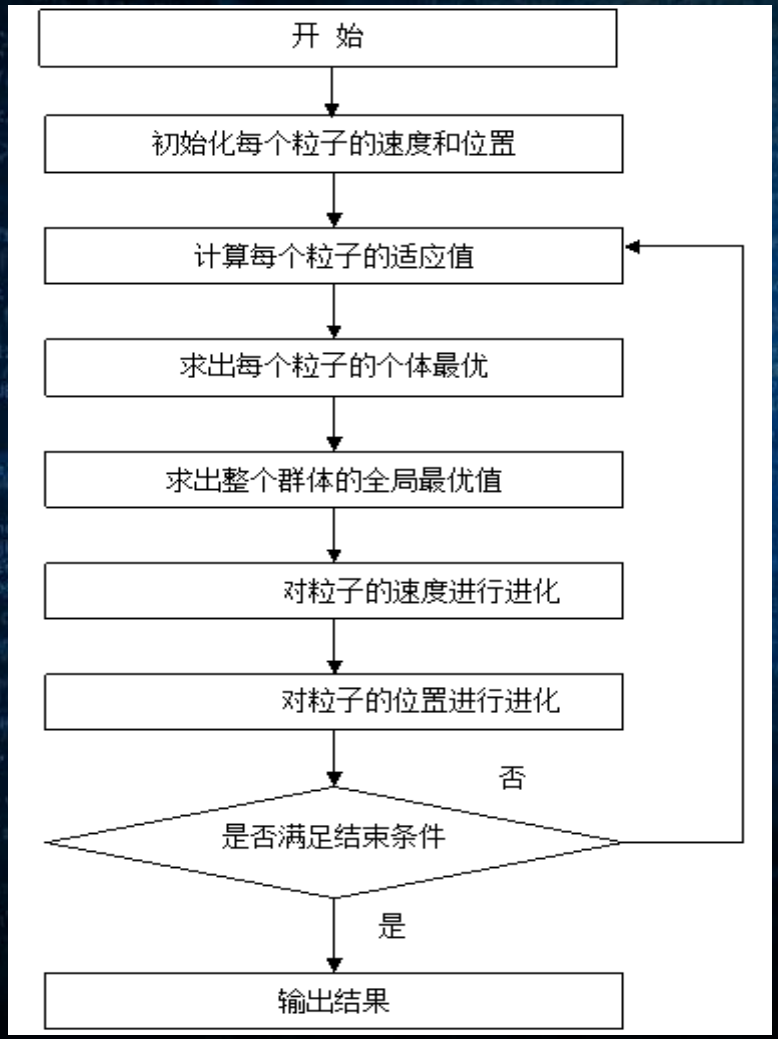
- 由于在早期的粒子群算法中，粒子速度主要是根据粒子的当前位置和个体最优值及全局最优值进行更新，因此很容易在算法运行后期出现“振荡”现象。
- 所谓“振荡”现象是指随着粒子群中的粒子聚集在目标的周围，在全局最优解的附近振荡，导致在较大的迭代次数内才能收敛于全局最优解。
- Eberhart和Shi在1998年将惯性因子引入粒子速度的更新公式中，即

$$V_i(t+1) = wV_i(t) + c_1r_1(P_i - X_i(t)) + c_2r_2(Pg - X_i(t))$$

算法中惯性因子w随着迭代的进行由最大加权因子w_{max}线性减小到最小加权因子w_{min}。

一般是将w_{max}设置为0.9，w_{min}设置为0.4。

粒子群算法流程图


开 始
初始化每个粒子的速度和位置
计算每个粒子的适应值
求出每个粒子的个体最优
求出整个群体的全局最优值
对粒子的速度进行进化
对粒子的位置进行进化
否
是否满足结束条件
是
输出结果

全局粒子群算法过程

(1) 随机初始化粒子群，即t=0时随机为每个粒子指定一个位置Xi(0)及速度Vi(0)；

(2) 计算每个粒子的适应度值f(Xi(t))；

(3) 比较每个粒子的当前适应度值f(Xi(t))和个体最优值f(Pi)，如果f(Xi(t))<f(Pi)，那么Pi=Xi(t)；

(4) 比较每个粒子的当前适应度值f(Xi(t))和全局最优值f(Pg)，如果f(Xi(t))<f(Pg)，那么Pg=Xi(t)；

(5) 按下列公式更改每个粒子的速度矢量和位置：

$$\begin{cases} V_i(t+1) = wV_i(t) + c_1 r_1 (P_i - X_i(t)) + c_2 r_2 (Pg - X_i(t)) \\ X_i(t+1) = X_i(t) + V_i(t+1) \end{cases}$$

(6) 如果满足终止条件，则输出Pg；否则，t =t+1，转(2)。

粒子群算法的参数

PSO算法中需要调节的参数，以及经验设置：

① 粒子数m(种群大小)：一般取20-40。对于比较难的问题或者特定类别的问题，粒子数可以取到100或200。

② 粒子的长度n(空间维数)：这是由优化问题决定的，就是问题解的长度。

③ 粒子的坐标范围：由优化问题决定，每一维可设定不同的范围。

④ 学习因子：c_1和c_2通常等于2，在一些文献中也有其它取值。但是一般c_1和c_2相等，并且范围在0和4之间。

⑤ 中止条件：最大循环数以及最小偏差要求，由具体问题确定。

粒子群算法特点

★基于群体叠代的随机搜索算法

- 同遗传算法类似，但没有遗传算法中的交叉和变异操作，
- 而是粒子在解空间追随最优粒子进行搜索。

★PSO的优势在于

- 简单容易实现，
- 既适合科学研究，又特别适合工程应用，
- 没有过多参数需要调整。

第六章 智能优化方法

- ★ 什么是智能优化方法
- ★ 遗传算法
- ★ 蚁群算法
- ★ 粒子群算法
- ★ 自适应协方差矩阵进化策略

传统进化策略算法

传统的进化策略(Evolution Strategy)借助一维正态分布实现个体突变。

首先，作用于步长σ来调整突变强度，其次，作用于旋转角度α来调整突变方向。

在Es中，有两种常用的变异强度调整方法，一种为Rechenberg提出的“1 / 5成功法则”，但其多用于“+”选择方案，而且仅能调整一个通用步长；另一种为“变异步长控制”，

步长σ被加入染色体参与进化过程的变异和选择，具有较大的随机波动性。对于旋转角度 α ，一般做小范围的随机扰动来实现调整，但这样会产生大量的无效突变，增加无用的计算开销。

自适应协方差矩阵进化策略算法

针对ES中策略参数调整方法的局限和不足， Nikolaus Hansen等人于1996年提出一种新的进化算法：自适应协方差矩阵进化策略（Covariance Matrix Adaptation Evolution Strategy, CMA-ES）算法，通过模拟自然界生物进化过程，从而达到寻优目的。

CMA-ES通过协方差矩阵来直接描述种群的突变尺度和方向，最终得到协方差矩阵自适应进化策略。

CMA-ES是在进化策略ES的基础上发展起来的一种高效的搜索算法，它将ES的可靠性、全局性与自适应协方差矩阵的高引导性结合起来，对求解非凸非线性优化问题具有较强的适应性。

CMAES

为了克服ES的局限性，CMAES采用多维正态分布N(x,σ)进行采样产生种群分布。

协方差矩阵C直接描述群体突变分布的形状，一个附加的参数全局步长σ当做协方差矩阵C的一个全局尺度因子。

旋转和尺度，将当代最优子群与前一代群体均值m之间的关系更新协方差矩阵来实现这个群体突变方向的调整，个体则由N（m,C）抽样获得。

CMAES算法步骤

步骤1：参数设置。

静态参数：子代个体数为λ，父代个体数为μ，重组权值w,以及自适应调整时所需的常量量Cd、cc、Cl等，最大迭代次数为G。

动态参数：初始分布均值m，初始化全局步长σ，初始进化路径，初始协方差矩阵C，初始化进化代数g=0

CMAES算法步骤

步骤2：种群采样。

采样公式如下：

$$x_k^{(g+1)} = m^{(g)} + \sigma^{(g)} N_k\left(0, C^{(g)}\right),\ k = 1, 2, \ldots, \lambda$$

步骤3：评价与选择。对子代个体进行逐个适应度评价并排序，进行（μ，λ）截断选择组成当前最优子群。

步骤4：参数更新。

CMAES算法步骤

步骤4.1：均值移动。当前最优子群加权重组得到新的分布均值。

更新公式如下：

$$m^{(g+1)} = m^{(g)} + \sigma^{(g)} \langle y \rangle_w = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g)}$$

步骤4.2：协方差矩阵自适应调整（CMA）。

$$p_c^{(g+1)} = (1-c_c) \cdot p_c^{(g)} + h_\sigma^{(g+1)} \sqrt{c_c(2-c_c)} \cdot \sqrt{\mu_{eff}} \cdot \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

$$C^{(g+1)} = (1-c_1-c_\mu) C^{(g)} + c_1 \left(p_c^{(g+1)} p_c^{(g+1)^T} + \delta(h_\sigma^{(g+1)}) C^{(g)} \right) + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g+1)} y_{i:\lambda}^{(g+1)^T}$$

CMAES算法步骤

步骤4.3：全局步长控制（CSA）。

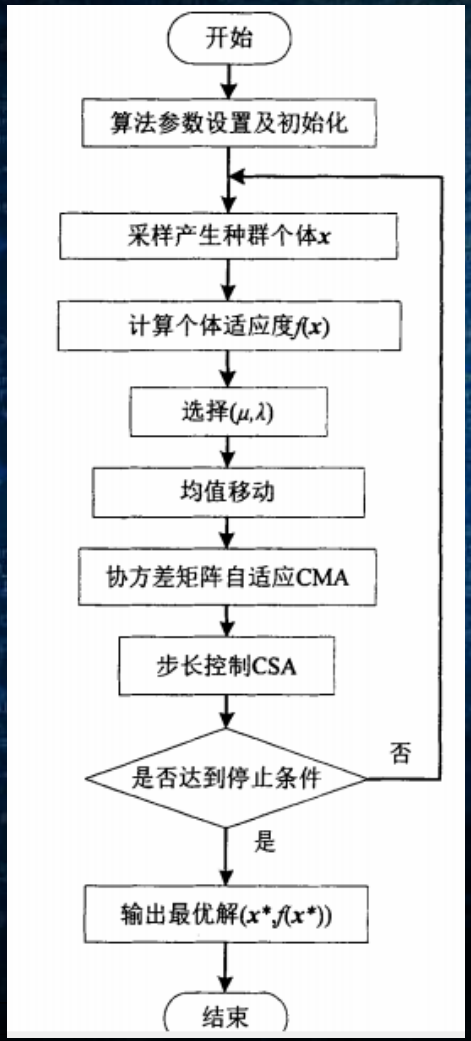
更新公式如下：

$$p_\sigma^{(g+1)} = (1-c_\sigma)\cdot p_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\cdot\sqrt{\mu_{eff}}\cdot C^{-\frac{1}{2}}\cdot\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}}$$

$$\sigma^{(g+1)} = \sigma^{(g)}\cdot\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|p_\sigma^{(g+1)}\right\|}{E\left(\left\|N(0,I)\right\|\right)}-1\right)\right)$$

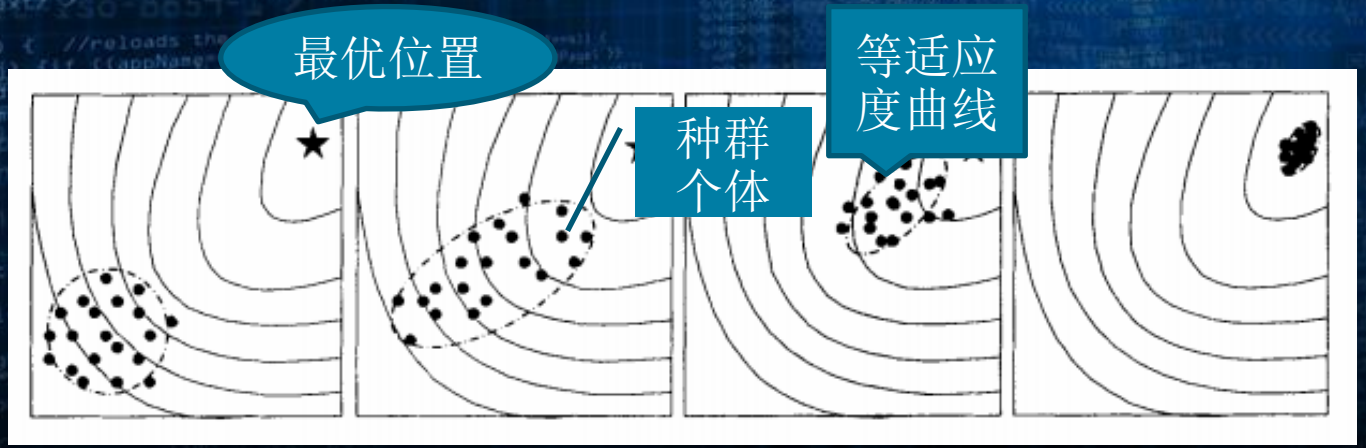
步骤5：是否达到停止条件?若是，则停止，输出最优解和最优值，否则返回步骤2.

CMAES算法流程图



$N(x, \sigma)$

CMA-ES的参数更新利用了所谓的“进化路径”，其是协方差矩阵的更新结合“秩1”和“秩μ”两种机制，可以很好地适应大小种群的变化。算法通过具有高引导性的协方差矩阵和有效的全局步长，使得演化过程具有很高的效率。下图表示CMA．ES算法的典型的迭代寻优过程：



CMAES算法主要特点

- 使用多变量的正态分布产生新的搜索点；（遵循最大熵原理）
- 基于排序的选择过程；
- 步长控制使得快速收敛更加便捷
- 协方差矩阵自适应算法增加了成功步长的似然性，可以根据问题规模的数量级改善性能

CMAES算法优点

- 高效的寻优性能，种群通过“采样一选择一更新一再采样”的循环过程，逐步向最优解逼近。
- CMA方法具有很高的效率，可以在一定程度上避免种群

过早收敛，并且CMA方法具有的高引导性可同时适应大小种群的变化，小种群可实现快速收敛，大种群可增强全局搜索能力

- 采用严格的择优截断选择策略，使得种群严格按照“优胜劣汰“的规则进行演化，这样有助于算法加速收敛。

CMAES算法缺点

- 参数较多，自适应过程比较复杂；
- 搜索过程缺乏种群多样性；
- 方差矩阵的频繁分解需要大量的计算开销。
- 均值移动过程仅仅使用了当前最优信息，忽视了历史最优信息；
- 确定性函数映射的步长控制过程未能充分体现进化过程的不确定性影响；

感谢聆听
欢迎提问