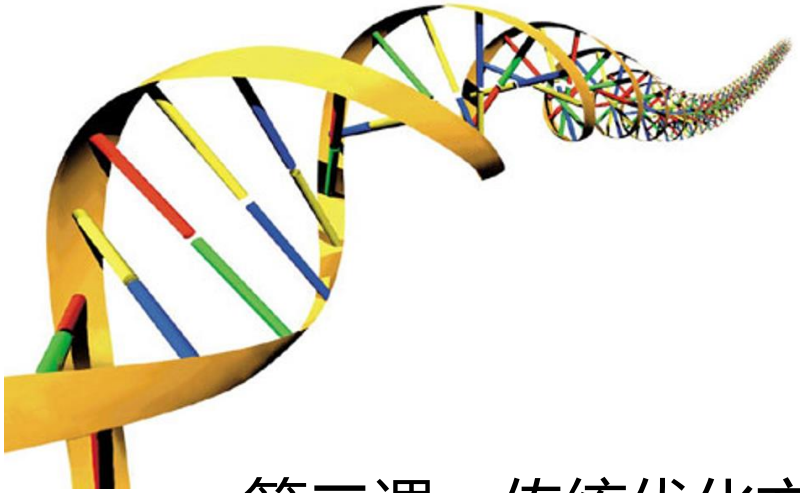
进化优化算法

基于仿生和种群的计算机智能方法

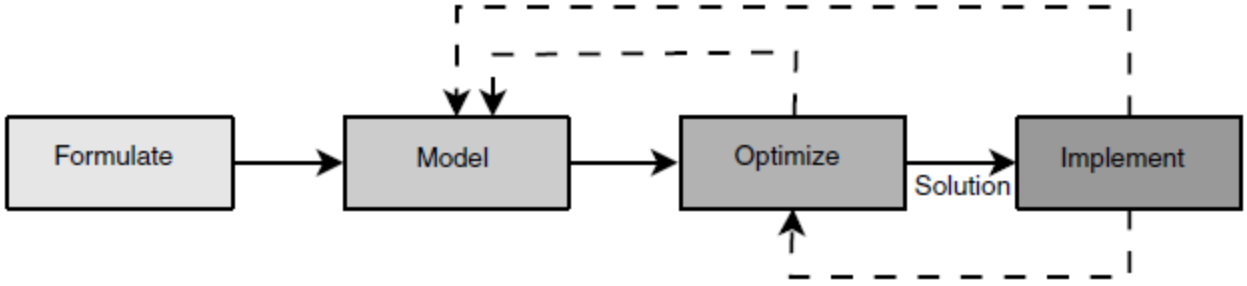


第二课：传统优化方法及遗传算法

概述

- 引言
- 进化优化算法的分类
- 一个应用优化算法的例子
- 最优化问题的数学模型
- 最优化问题的分类
- 最优化方法的分类
- 进化优化算法通用框架
  - ✓流程图及通用框架
  - ✓优点及局限

优化模型



- 提出问题：指出研究问题的目标，问题的内在和外在因素。对问题进行初步的描述及说明
- 为问题建模：为问题构建数学模型，通常将复杂的实际问题简化为典型的优化问题模型
- 优化问题：选择优化算法，求解得到问题的优质解。优质解可能是最优解或次优解
- 完成解决方案：将得到的优质解在实际问题中进行测试，如果该解不理想，对模型和优化算法进行改进，重复优化过程

最优化问题

- E.g. 开车上班怎样选择路线才能最快到达单位，旅游如何选择航班和宾馆才能既省钱又能参观更多的景点
- 最优化问题就是在一个给定的变量空间内，依据一定的判定条件，在多个已知解中选择最优解的问题
- 某人从家到单位的各条路线均包含多个中间地点，假设他有N条可以从家到达单位的路线，其中有的路线最近，有的路线用时最少，如果要从这N条路线中选择用时最少的，这个问题就是最优化问题
  - 上述例子中，中间地点称为设计变量
  - 由中间地点构成的从家到单位的路线称为已知解
  - 中间地点构成的集合称为变量空间
  - 用时最少是判定条件，在最优化问题中被称为目标函数或适应度函数

最优化问题

- 最优化问题的求解过程分为
  - ✓ 建模：将所求解的实际问题转化为最优化数学模型的过程。包括设计变量的确定、所有已知解的表达以及目标函数的构建等
  - ✓ 优化：选择相应的优化方法以对模型问题进行求解并最终给出最优解的过程，进化优化算法即可提供这样一类优化方法
- 在最优化问题中，有时候会存在多个判定条件，如在上述例子中若要寻找用时最少且路程最近的路线，则目标函数有两个，这样的最优化问题又被称为多目标优化问题
- 有的最优化问题还存在约束条件，假如某些路线有特殊情况，如道路施工封路，则“该路线无法通行”，即为约束条件，且在寻优过程中需要考虑该约束条件，此时最优化问题又被称为约束优化问题

目标函数

- 目标函数是问题优化的目标和标准
- 目标函数可以是连续函数，也可以是半连续函数
- 一个优化问题可以写成最小化问题或最大化问题
- 最大化问题与最小化问题可以通过乘以-1相互转换

$$\min_x f(x) \quad \Leftrightarrow \quad \max_x [-f(x)]$$

$$\max_x f(x) \quad \Leftrightarrow \quad \min_x [-f(x)]$$

函数$f(x)$被称为目标函数

向量x被称为独立变量或决策变量

向量x中元素的个数为问题的维数

$$\min_x f(x) \quad \Rightarrow \quad f(x)\text{被称为“费用”或“目标”}$$

$$\max_x f(x) \quad \Rightarrow \quad f(x)\text{被称为“适应度”或“目标”}$$

例题

例2.1　这个例子说明本书中用到的术语。假设我们想要最小化函数

$f(x, y, z) = (x - 1)^2 + (y + 2)^2 + (z - 5)^2 + 3$

变量x, y, z被称为独立变量、决策变量、或解的特征

$f(x, y, z)$被称为目标函数或费用函数

定义$g(x, y, z) = -f(x, y, z)$并最大化$g(x, y, z)$

函数$g(x, y, z)$被称为目标函数或适应度函数

min $f(x, y, z)$与 max $g(x, y, z)$的解相同，解为$x = 1, y = -2$和$z = 5$

限制条件

- 优化参数x可能有限制条件
  - 不等式限制条件(Inequality Constraints) （通常是资源的限制）e.g. $g(x)\leq 0$
  - 等式限制(Equality Constraints)（通常是参量守恒）e.g. $h(x)=0$
  - 可行解（Feasible Solution）：满足所有限制条件的解
  - 不可行解（Infeasible Solution）：至少不满足一个限制条件的解
  - 严格限制条件（Hard Constraints）：必须满足的限制条件
  - 非严格限制条件（Soft Constraints）：可以一定程度违反的限制条件

例题

最小化函数 $f(x)=\left(\frac{\pi x_1^2}{2}+\pi x_1 x_2\right)$

限制条件 $\frac{\pi x_1^2 x_2}{4}\geq 300 \quad \Leftrightarrow \quad g(x)\geq 300$

考虑两个解$S_1=[3 \quad 10]$和$S_2=[8 \quad 6]$

把$S_1=[3 \quad 10]$代入$f(x)$ $f_1=108.38$

但是$g_1=70.69$ 不满足限制条件$g(x)\geq 300$

把$S_2=[8 \quad 6]$代入$f(x)$ $f_2=251.32$ $g_2=301.59$

满足限制条件$g(x)\geq 300$

虽然$f_1<f_2$,但由于S_1不满足限制条件，S_1 是不可行解，S_2 是可行解

最优化问题的数学模型

最小化优化问题（求目标函数的最小值）为例，一个具有n个设计变量、m个目标函数和$p+q$个约束条件的最优化问题的数学模型如下式所示

$$\min_{X\in R^n} F(X)=[f_1(X), f_2(X),..., f_m(X)]$$

$$s.t.\begin{cases} g_i(X)\le 0, & i=0,\ 1,..., p \\ h_j(X)=0, & j=0,\ 1,..., q \end{cases}$$

式中$X=(x_1,x_2,...,x_n)^T\in R^n$称为已知解，$x_1,x_2,...,x_n$称为设计变量，$R^n$称为$n$维变量空间，$F(X)$称为目标函数

$g_i(X)\le 0$为第i个不等式约束条件，p为不等式约束条件的个数；

$h_j(X)=0$为第j个等式约束条件，q为等式约束条件的个数。

特别的，当$m=1$且$i=j=0$时，上式称为单目标无约束优化问题，是最简单的最优化问题

概述

- 引言
- 进化优化算法的分类
- 一个应用优化算法的例子
- 最优化问题的数学模型
- 最优化问题的分类
- 最优化方法的分类
- 进化优化算法通用框架
  - ✓流程图及通用框架
  - ✓优点及局限

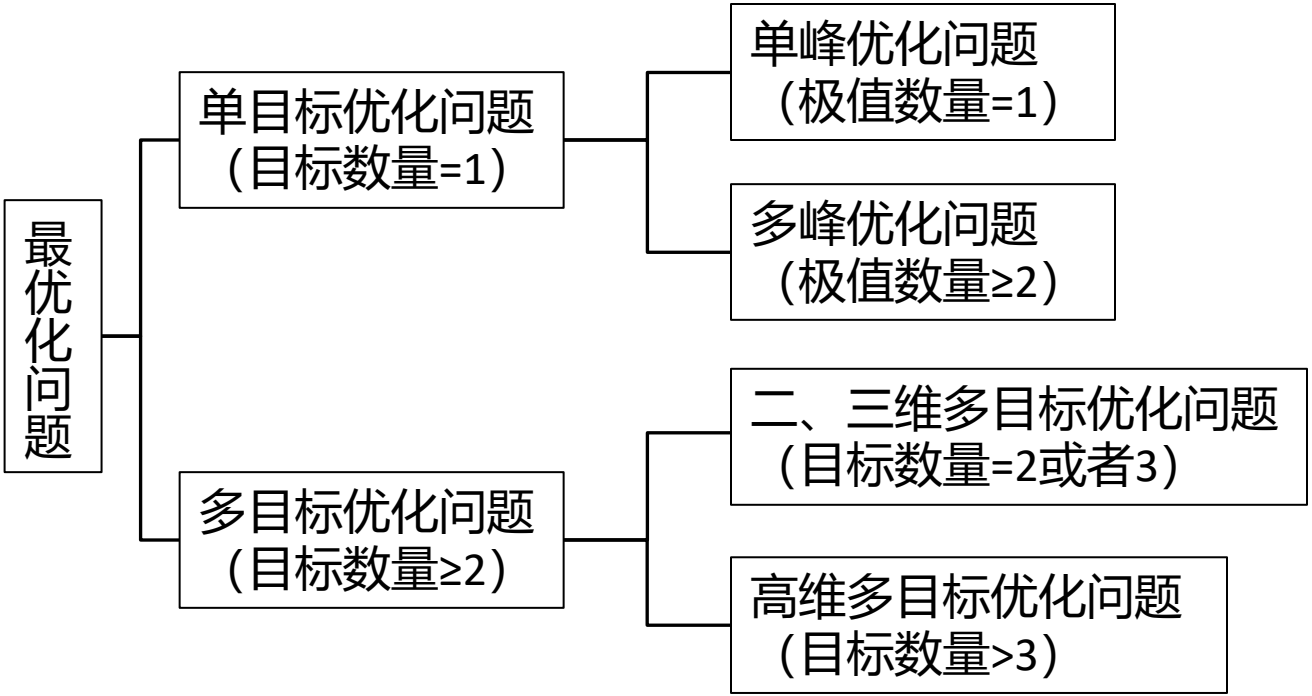
最优化问题的分类

- 根据目标的数量
  - 单目标优化问题（Single-objective Optimization Problems, SOPs）
  - 多目标优化问题(Multi-objective Optimization Problems, MOPs)

- 根据设计变量是否连续
  - 连续变量优化问题 （函数优化问题）
  - 离散变量优化问题 （组合优化问题）

- 根据是否有约束条件
  - 无约束优化问题
  - 约束优化问题

单目标优化问题和多目标优化问题



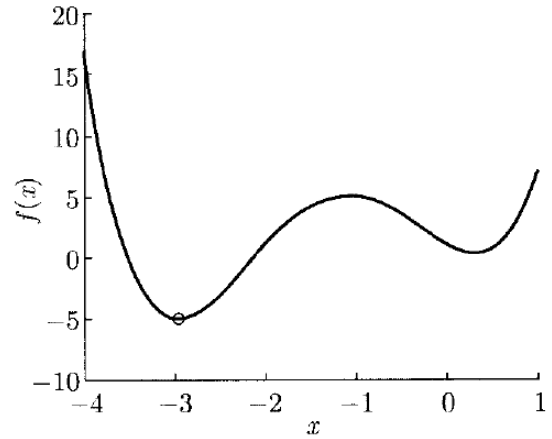
- 多目标优化问题中，多个目标之间常常存在冲突，即某个目标性能的改善可能会引起其他目标性能的降低，多目标优化问题的求解难度远远大于单目标优化问题

局部最优和全局最优

考虑问题 $\min\limits_{x} f(x)$, 其中$f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1$

$f(x)$有3个驻点, $x = -2.96$, $x = -1.10$, $x = 0.31$,

$$f''(x) = 12x^2 + 30x + 8 = \begin{cases} 24.33, & x = -2.96 \quad \text{局部最小点} \\ -10.48, & x = -1.10 \quad \text{局部最大点} \\ 18.45, & x = 0.31 \quad \text{局部最小点} \end{cases}$$



例 2.2: 简单的最小化问题. $f(x)$ 有两个局部最小值, $x = -2.96$ 为全局最小值点.

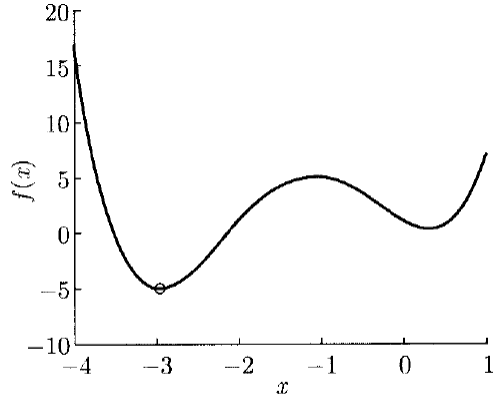
局部最小值和全局最小值

局部最小值x^*定义为

对所有满足

$\|x - x^*\| < \varepsilon$的$x$,

$f(x^*) < f(x)$



例 2.2: 简单的最小化问题. $f(x)$ 有两个局部最小值, $x = -2.96$ 为全局最小值点.

其中,$\|\cdot\|$是某个距离的度量，$\varepsilon > 0$是由用户定义的邻域大小

如果$\varepsilon = 1$，$x = 0.31$是局部最优，但是若$\varepsilon = 4$,它就不再是局部最优

全局最小值x^*定义为

对所有的x，$f(x^*) \leq f(x)$

多峰优化

考虑问题

$\min\limits_{x,y} f(x,y)$，　　其中

$$f(x,\mathrm{y}) = e - 20\exp\left(0.2\sqrt{\frac{x^2+y^2}{2}}\right) - \exp\left(\frac{\cos(2\pi x)+\cos(2\pi y)}{2}\right)$$

其中$f(x,y)$是二维*Ackley*函数

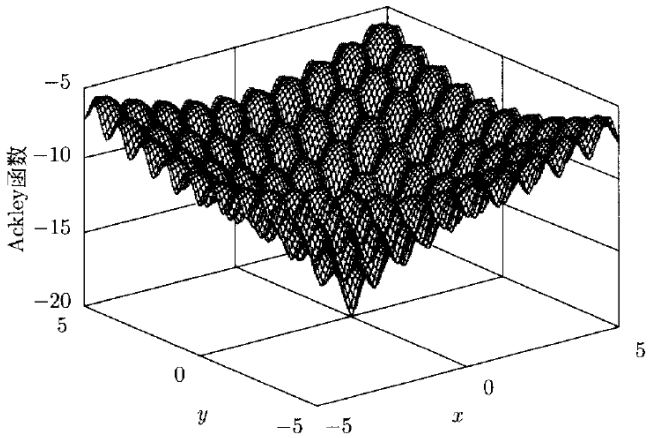


图 2.5　例 2.5: 二维 Ackley 函数.

对于这类问题，进化算法能给出很好的结果

基准测试函数

基准测试函数

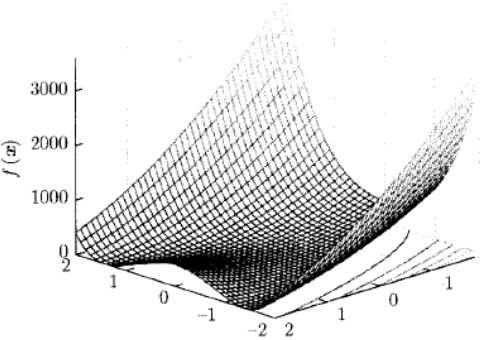


图 C.4　二维 Rosenbrock 函数.

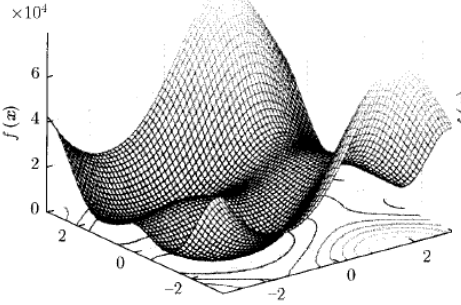


图 C.5　二维 Fletcher 函数.

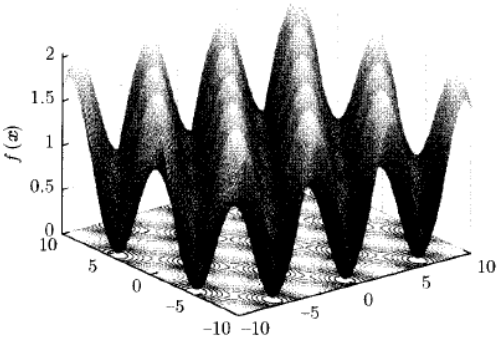


图 C.6　二维 Griewank 函数.

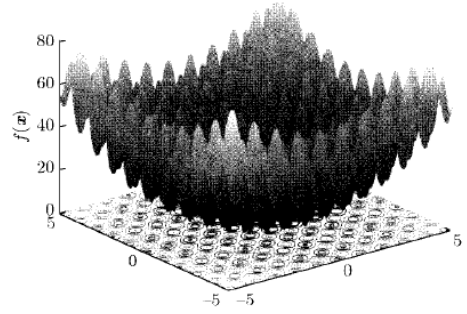


图 C.11　二维 Rastrigin 函数.

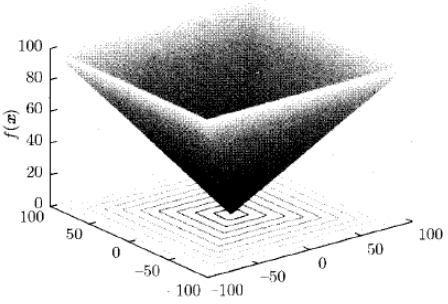


图 C.13　二维 Schwefel 最大函数.

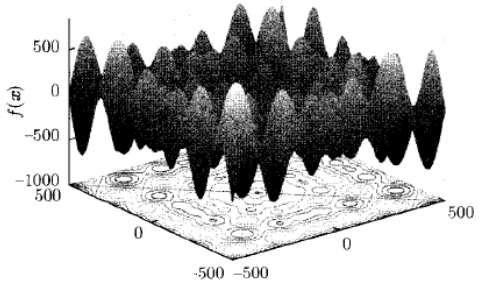
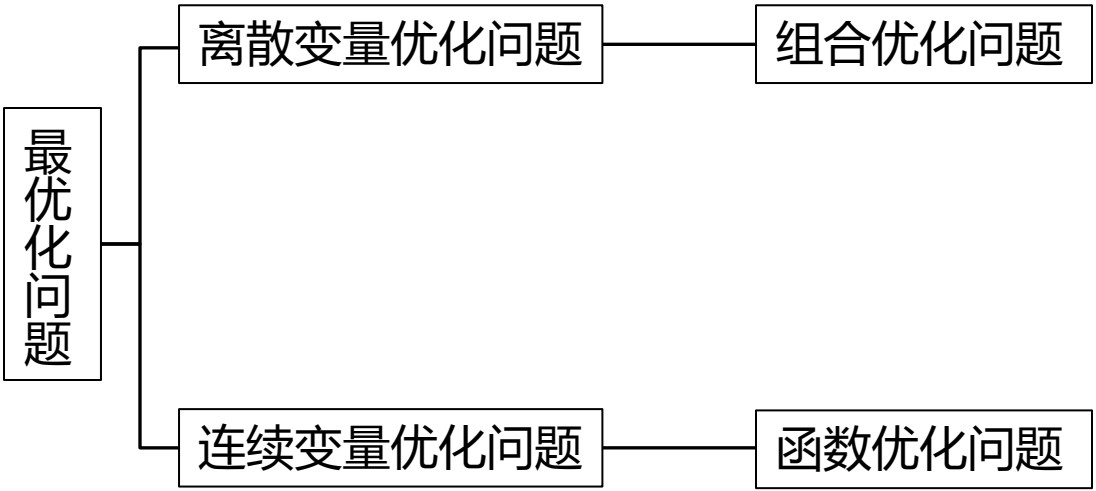


图 C.15　二维 Schwefel 正弦函数.

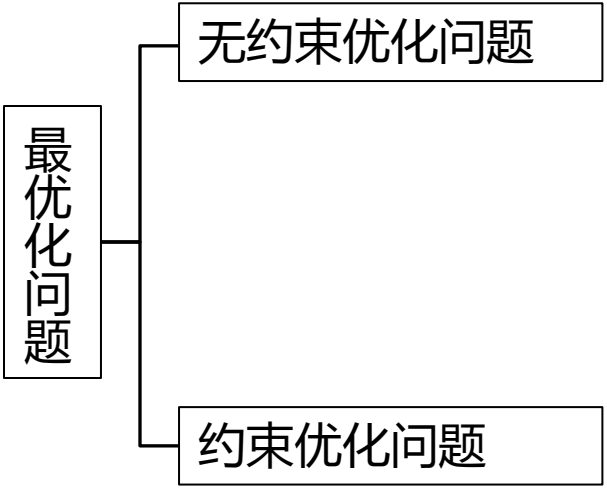
离散变量优化问题和连续变量优化问题



- 离散变量优化问题最常见的问题是组合优化问题：旅行商问题、调度问题、背包问题、装箱问题、图着色问题、聚类问题等

- 连续变量优化问题最常见的问题是函数优化问题：单峰函数优化问题、多峰函数优化问题、动态多目标函数优化问题等

无约束优化问题和有约束优化问题



- 目标函数和设计变量没有约束条件的问题称为无约束优化问题，简称优化问题

- 约束条件的存在会使变量空间（由可行域和不可行域组成）中的可行域变得非常狭小且不连贯，因此，约束优化问题的处理更为困难

约束优化

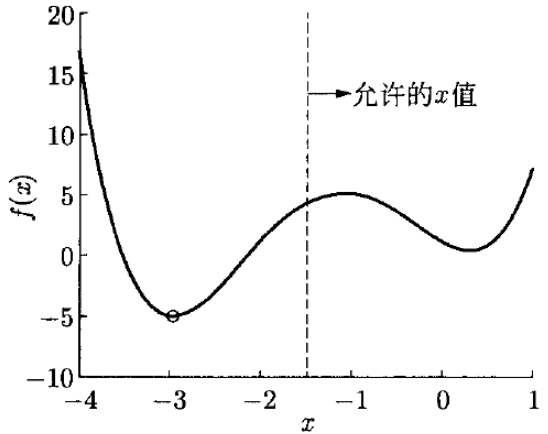
考虑问题

$\min_x f(x)$，　　其中$f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1$　　并且$x \geq -1.5$

3个驻点,$x = -2.96$,　$x = -1.10$,　$x = 0.31$、

$$f''(x) = 12x^2 + 30x + 8 = \begin{cases} 24.33, & x = -2.96 \quad \text{局部最小点} \quad \text{不满足条件} \\ -10.48, & x = -1.10 \quad \text{局部最大点} \quad \text{满足条件} \\ 18.45, & x = 0.31 \quad \text{局部最小点} \quad \text{满足条件} \end{cases}$$

实际的优化问题几乎总是带有约束。在实际的优化问题中，使目标函数最优的独立变量的值也几乎总是出现在约束的边界上



例 2.3: 一个简单的带约束最小化问题. 带约束的最小值出现在 $x = 0.31$ 处.

概述

- 引言
- 进化优化算法的分类
- 一个应用优化算法的例子
- 最优化问题的数学模型
- 最优化问题的分类
- 最优化方法的分类
- 进化优化算法通用框架
  - ✓流程图及通用框架
  - ✓优点及局限

传统优化方法

- 传统优化方法属于确定性的数学类方法，它们按照固定的搜索方式来寻找最优解。最常见的有梯度下降法(Gradient Descent)、共轭梯度法（Conjugate Gradient）、牛顿法（Newton's method）等。主要包括以下3个步骤

1. 选择一个初始解，该解通常需要为可行解
2. 改进解的移动方向，一般表现为优化梯度
3. 判断终止条件，该条件一般被设置为最大迭代次数小于某一整数或前后两次迭代求解中的目标函数值小于一个很小的实数

凹函数和凸函数

$\arg\min$ 或者 $\arg\max$：含义是当函数$F(x)$取最小（大）值时的变量值

*e.g.*函数$F(x)=(x-1)^2$的最小值为0　　$\arg\min F(x)=1$

对于严格凹/凸函数（*strictly concave/convex functions*),

$\arg\min F(x)/\arg\max F(x)$是一个点$x^*$:孤立极小/大值

严格凹函数：任意一点的二阶导数大于0，函数值位于切线上方

严格凸函数：任意一点的二阶导数小于0，函数值位于切线下方

判断函数极大值以及极小值:

当一阶导数等于0，而二阶导数大于0，为极小值点

当一阶导数等于0，而二阶导数小于0 时，为极大值点

$\arg$是“$\arg ument$"的缩写

- 凸函数和凹函数在国内和国外的一些定义是相反的

多变量微积分的基础知识

一个函数F，一个变量x

$$F(x+\Delta x) \approx F(x) + \Delta x \frac{dF}{dx}(x) + \frac{1}{2}(\Delta x)^2 \frac{d^2F}{dx^2}(x) \qquad (1)$$

*Taylor*级数展开的前几项----通常不超过二阶

$F(x) + \Delta x \frac{dF}{dx}(x)$是函数的一阶近似

$F(x) + \Delta x \frac{dF}{dx}(x) + \frac{1}{2}(\Delta x)^2 \frac{d^2F}{dx^2}(x)$是函数的二阶近似

优化方法:

一阶近似 -------- 梯度下降法

二阶近似 -------- 牛顿法

多变量微积分的基础知识

一个函数F，n个变量x_1，x_2,...x_n

$$F(x+\Delta x) \approx F(x) + (\Delta x)^T \nabla F + \frac{1}{2}(\Delta x)^T H(\Delta x) \qquad (2)$$

∇F是函数F的梯度，矩阵H是*Hessian*（黑塞）矩阵，$H_{ij} = \dfrac{\partial^2 F}{\partial x_i \partial x_j} = \dfrac{\partial^2 F}{\partial x_j \partial x_i} = H_{ji}$

$$\nabla F = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_n}\right]^T$$

$$H = \begin{bmatrix} \dfrac{\partial^2 F}{\partial x_1^2} & \dfrac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 F}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 F}{\partial x_2 \partial x_1} & \dfrac{\partial^2 F}{\partial x_2^2} & \cdots & \dfrac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 F}{\partial x_n \partial x_1} & \dfrac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

什么是梯度下降法?

- 梯度下降法是最常用的优化方法之一，常用来求解目标函数的极值
- 基本原理：从一个随机点出发，沿着目标函数梯度下降的方向搜索极小值（也可以沿着梯度上升的方向搜索极大值）

真实优化问题，可能并不容易直接得到目标函数的导数

可以采用有限差分近似求导, 函数$f(x, y)$

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + h, y) - f(x, y)}{h}$$

$$\frac{\partial f(x, y)}{\partial y} \approx \frac{f(x, y + h) - f(x, y)}{h}$$

...

梯度下降法的基本步骤

gradient descent $x_{k+1} = x_k - S_k \nabla f\left(x_k\right)$

$S_k > 0$很小的量

$$f(x_{k+1}) = f(x_k - S_k \nabla f\left(x_k\right)) \approx f(x_k) - S_k \nabla f\left(x_k\right)^T \nabla f\left(x_k\right) < f(x_k)$$

从某个初始点x_0出发

1.计算上升 / 下降的方向：d_0

2.朝着上升 / 下降的方向移动：$x_{k+1} = x_k + S_k d_k$

其中 $\begin{cases} d_k = \nabla f(x_k) & \text{上升} \\ d_k = -\nabla f(x_k) & \text{下降} \end{cases}$

3.迭代直到达到收敛条件

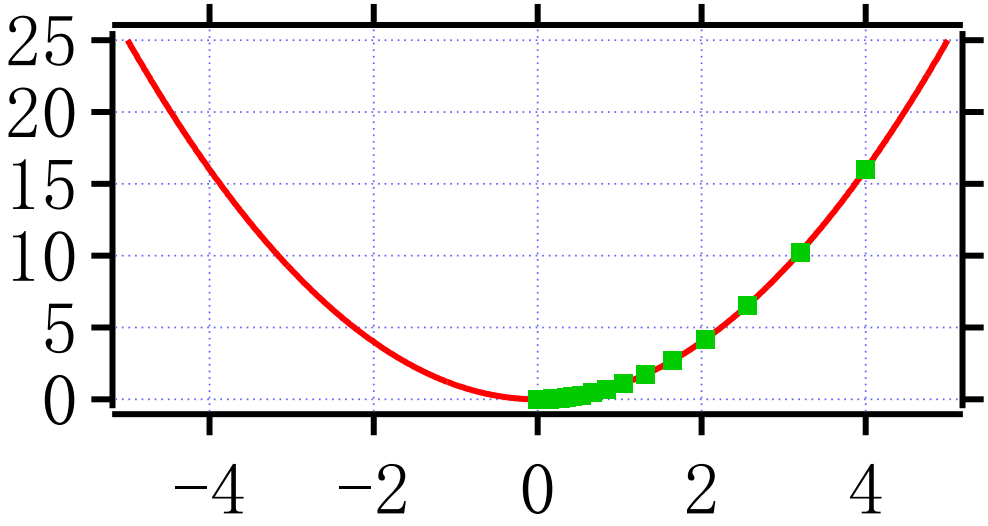
梯度下降法示例（一维）

$f(x) = x^2$

$\frac{df}{dx} = 2x$

$Step1$：随机选择一个初始点$x_0 = 4$, 然后计算函数在此点的梯度

$\frac{df}{dx}_{|x_0=4} = 2x = 8$



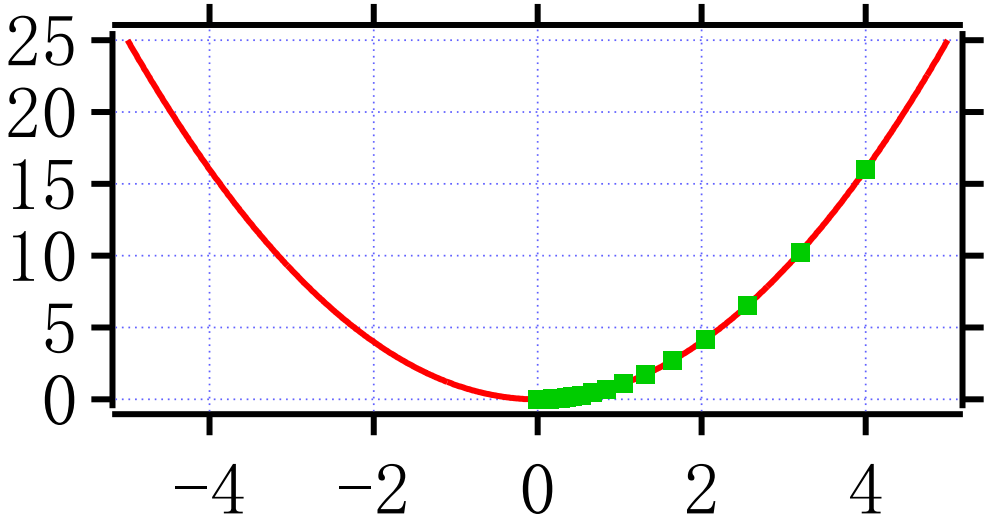
梯度下降法示例（一维）

$f(x) = x^2$

$\frac{df}{dx} = 2x$

*Step*1：随机选择一个初始点$x_0 = 4$, 然后计算函数在此点的梯度

$\frac{df}{dx}_{|x_0=4} = 2x = 8$



*Step*2：沿着梯度相反的方向移动。应该移动多少？

定义学习率$learning_rate = 0.1$

*Step*3：执行迭代操作，直至满足收敛条件(达到指定迭代次数，或$f(x)$近似收敛到最优解)

$x_1 = x_0 - (learning\_rate) \times (df/dx) = 4 - 0.1 \times 8 = 3.2$

$x_2 = x_1 - (learning\_rate) \times (df/dx) = 3.2 - 0.1 \times 6.4 = 2.56$

....

设置收敛条件$|f(x_n) - f(x_{n-1})| < 2 \times 10^{-15}$，程序迭代82步停止, $x_{82} = 5 \times 10^{-8}$

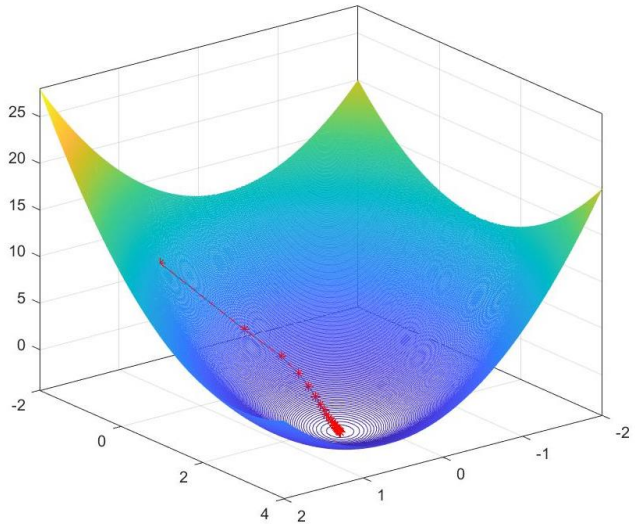
梯度下降法示例（二维）

$$f(x,y) = -xy - 4y + 3x^2 + y^2$$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} -y + 6x \\ -x - 4 + 2y \end{bmatrix}$$

*Step*1：随机选择一个初始点$x_0 = (1,-1)$，然后计算函数在此点的梯度

$$\nabla f(1,-1) = \begin{bmatrix} 1 + 6 \times 1 \\ -1 - 4 + 2 \times (-1) \end{bmatrix} = \begin{bmatrix} 7 \\ -7 \end{bmatrix}$$



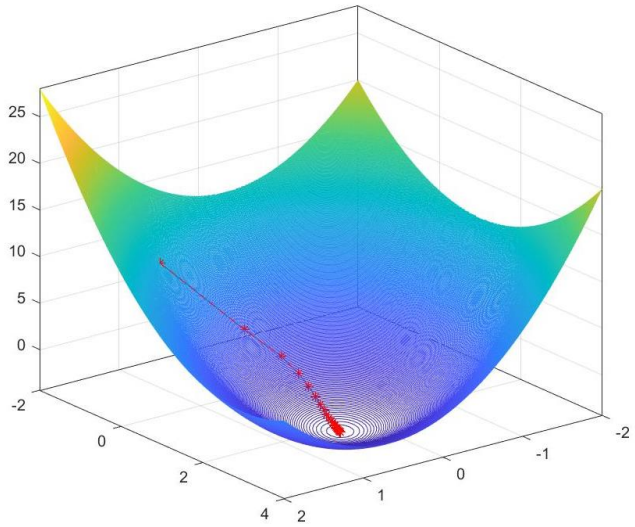
梯度下降法示例（二维）

$f(x,y) = -xy - 4y + 3x^2 + y^2$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} -y + 6x \\ -x - 4 + 2y \end{bmatrix}$$

*Step*1：随机选择一个初始点$x_0 = (1,-1)$，然后计算函数在此点的梯度

$$\nabla f(1,-1) = \begin{bmatrix} 1 + 6 \times 1 \\ -1 - 4 + 2 \times (-1) \end{bmatrix} = \begin{bmatrix} 7 \\ -7 \end{bmatrix}$$



*Step*2：沿着梯度相反的方向移动。应该移动多少？

定义学习率$learning_rate = 0.1$

*Step*3：执行迭代操作，直至满足收敛条件(达到指定迭代次数，或$f(x)$近似收敛到最优解)

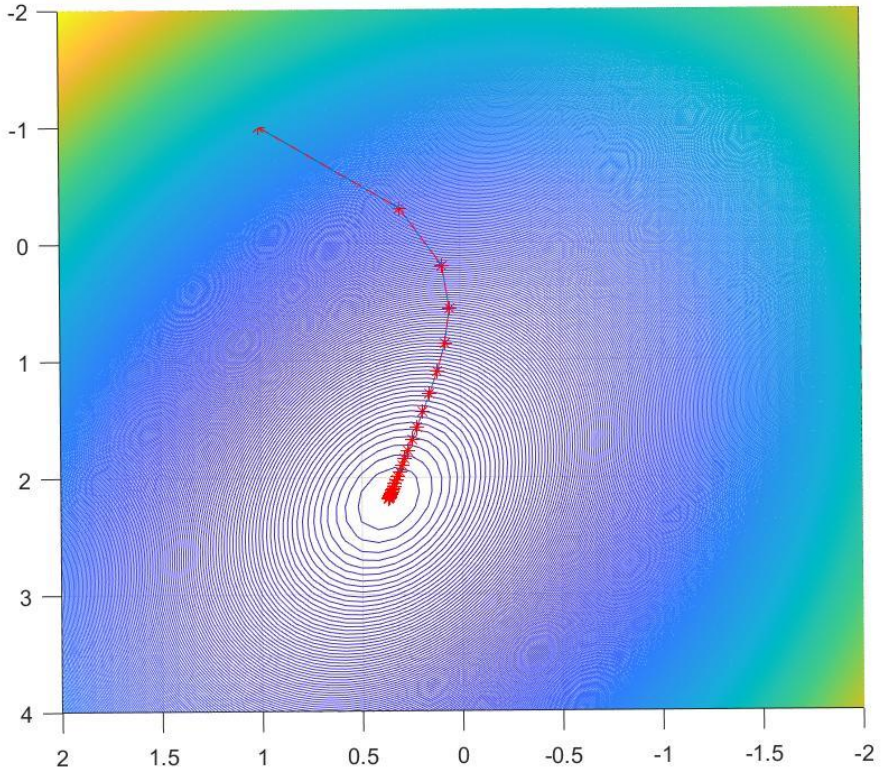
$$x_1 = x_0 - (learning\_rate) \times \nabla f(x,y)_{|x_0} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 0.1 \times \begin{bmatrix} 7 \\ -7 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.3 \end{bmatrix}$$

$$x_2 = x_1 - (learning\_rate) \times \nabla f(x,y)_{|x_1} = \begin{bmatrix} 0.3 \\ -0.3 \end{bmatrix} - 0.1 \times \begin{bmatrix} 0.3 + 6 \times 0.3 \\ -0.3 - 4 + 2 \times (-0.3) \end{bmatrix} = \begin{bmatrix} 0.09 \\ 0.19 \end{bmatrix}$$

....

设置收敛条件$\left|f(x_n) - f(x_{n-1})\right| < 2 \times 10^{-15}$，程序迭代89步停止，$x_{89} = \begin{bmatrix} 0.363636 \\ 2.18182 \end{bmatrix}$

梯度下降法示例（二维）



梯度下降法的优点和缺点

梯度下降法的优点：

- 实现简单
- 当目标函数是凸/凹函数时，梯度下降法的解是全局解

梯度下降法的局限：

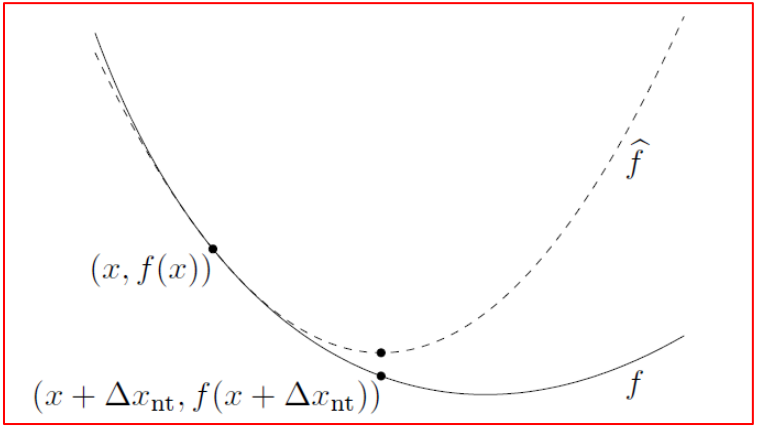
- 一般情况下不能保证得到全局最小值，可能得到的是局部最小值
- 靠近极小值时收敛速度减慢，求解需要很多次的迭代
- 可能会“之字形”地下降

牛顿法的基本思想

$$F(x+\Delta x) \approx F(x) + (\Delta x)\frac{dF}{dx} + \frac{1}{2}(\Delta x)^2 \frac{d^2F}{dx^2}$$

$$\frac{dF}{dx} + \frac{d^2F}{dx^2}\cdot(\Delta x) = 0$$

$$\Rightarrow \Delta x = -\frac{dF/\mathrm{d}x}{d^2F/dx^2}$$



牛顿迭代算法：$x_{k+1} = x_k - \dfrac{dF/\mathrm{d}x}{d^2F/dx^2}$

*Step*1：给出$x_0 \in R^n$，$0 \le \varepsilon << 1$，$k = 0$

*Step*2：计算$dF/\mathrm{d}x$，如果$\left|dF/\mathrm{d}x\right| \le \varepsilon$，停止

否则计算$\dfrac{d^2F(x_k)}{dx^2}$，并令$d_k = -\dfrac{dF/\mathrm{d}x}{d^2F/dx^2}$

*Step*3：令$x_{k+1} = x_k + d_k$，$k = k+1$，返回*Step*2

牛顿法示例（一维）

$f(x) = -0.8x^3 + 9x^2 + x + 3$

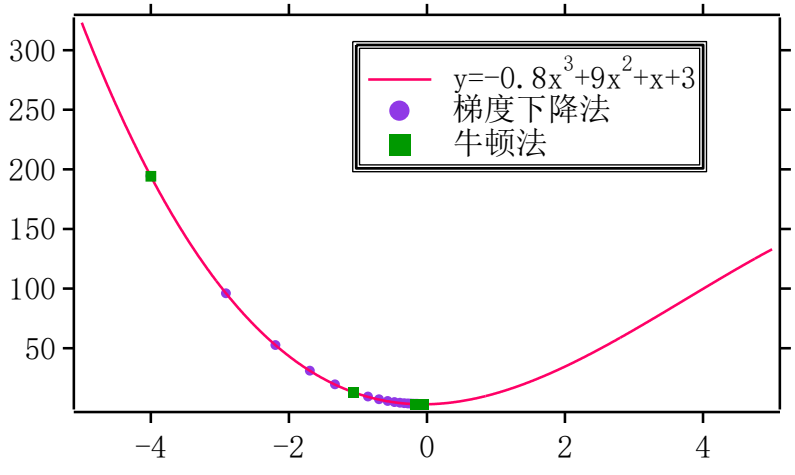
$\frac{df}{dx} = -2.4x^2 + 18x + 1$

$\frac{d^2f}{dx^2} = -4.8x + 18$

$Step1$：随机选择一个初始点$x_0 = -4$，然后计算函数在此点的梯度及二阶导数

$\frac{df}{dx}_{|x_0=-4} = -2.4x^2 + 18x + 1 = -109.4$

$\frac{d^2f}{dx^2}_{|x_0=-4} = -4.8x + 18 = 37.2$



牛顿法示例（一维）

$f(x) = -0.8x^3 + 9x^2 + x + 3$

$\frac{df}{dx} = -2.4x^2 + 18x + 1$

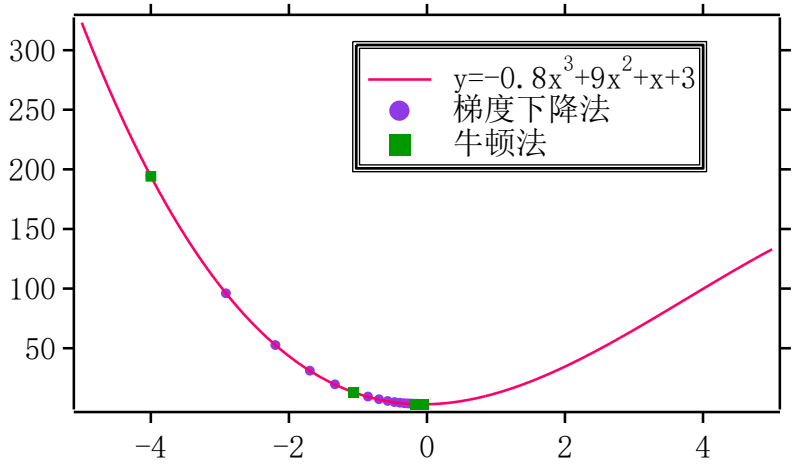
$\frac{d^2f}{dx^2} = -4.8x + 18$

*Step*1：随机选择一个初始点$x_0 = -4$，

然后计算函数在此点的梯度及二阶导数

$\frac{df}{dx}_{|x_0=-4} = -2.4x^2 + 18x + 1 = -109.4$

$\frac{d^2f}{dx^2}_{|x_0=-4} = -4.8x + 18 = 37.2$



*Step*2：执行迭代操作，直至满足收敛条件（达到指定迭代次数，或$f(x)$近似收敛到最优解）

$x_1 = x_0 - \frac{df/dx}{d^2f/dx^2} = -4 - \frac{(-109.4)}{37.2} = -1.059$

$x_2 = x_1 - \frac{df/dx}{d^2f/dx^2} = -1.059 - \frac{(-20.7536)}{23.0832} = -0.15992$

...

设置收敛条件$\left|f(x_n) - f(x_{n-1})\right| < 2 \times 10^{-15}$，梯度下降（$learning\_rate = 0.01$)91步停止，

$x_{91} = -0.05515$, 牛顿法5步停止, $x_5 = -0.05515$

牛顿法

$$F(x+\Delta x) \approx F(x) + (\Delta x)^T \nabla F + \frac{1}{2}(\Delta x)^T H(\Delta x)$$

$$\nabla F + H \cdot (\Delta x) = 0$$

$$\Rightarrow \Delta x = -H^{-1}\nabla F$$

牛顿迭代算法：$x_{k+1} = x_k - H^{-1}\nabla F$

*Step*1：

给出$x_0 \in R^n$，$0 \le \varepsilon << 1$，$k = 0$

*Step*2：

计算$\nabla F(x_k)$，如果$|\nabla F(x_k)| \le \varepsilon$，停止

否则计算$\nabla^2 F(x_k)$，并令$d_k = -[\nabla^2 F(x_k)]^{-1}\nabla F(x_k)$

*Step*3：

令$x_{k+1} = x_k + d_k$，$k = k+1$，返回*Step*2

∇F是函数F的梯度

$$\nabla F = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_n}\right]^T$$

矩阵H是*Hessian*（黑塞）矩阵，

$$H_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j} = \frac{\partial^2 F}{\partial x_j \partial x_i} = H_{ji}$$

$$H = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

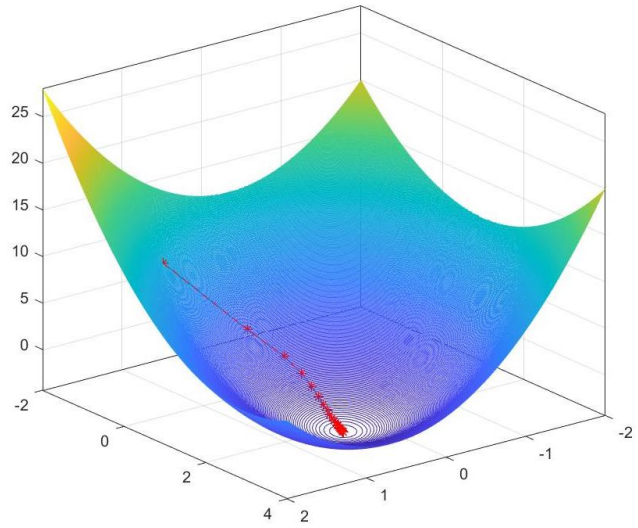
牛顿法示例（二维）

$$f(x,y) = -xy - 4y + 3x^2 + y^2$$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} -y + 6x \\ -x - 4 + 2y \end{bmatrix}$$

$$H = \begin{bmatrix} \dfrac{\partial^2 f(x,y)}{\partial x^2} & \dfrac{\partial^2 f(x,y)}{\partial x \partial y} \\ \dfrac{\partial^2 f(x,y)}{\partial y \partial x} & \dfrac{\partial^2 f(x,y)}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 6 & -1 \\ -1 & 2 \end{bmatrix}$$

$$H^{-1} = \begin{bmatrix} 0.181818 & 0.0909091 \\ 0.0909091 & 0.545455 \end{bmatrix}$$



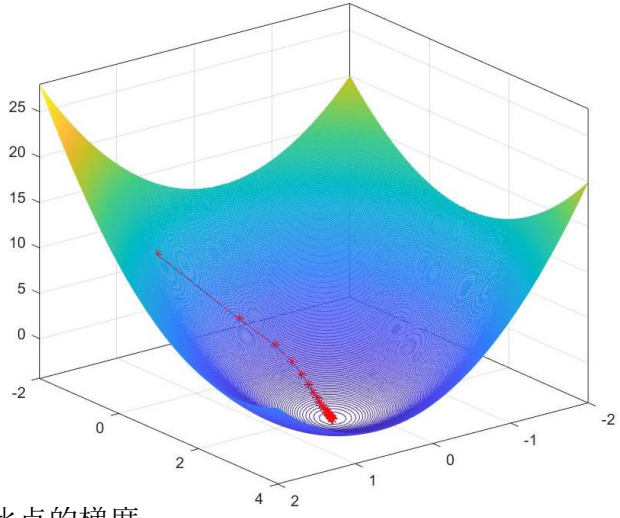
牛顿法示例（二维）

$$f(x,y) = -xy - 4y + 3x^2 + y^2$$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} -y + 6x \\ -x - 4 + 2y \end{bmatrix}$$

$$H = \begin{bmatrix} \dfrac{\partial^2 f(x,y)}{\partial x^2} & \dfrac{\partial^2 f(x,y)}{\partial x \partial y} \\ \dfrac{\partial^2 f(x,y)}{\partial y \partial x} & \dfrac{\partial^2 f(x,y)}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 6 & -1 \\ -1 & 2 \end{bmatrix}$$

$$H^{-1} = \begin{bmatrix} 0.181818 & 0.0909091 \\ 0.0909091 & 0.545455 \end{bmatrix}$$



$Step1$：随机选择一个初始点$x_0 = (1,-1)$，然后计算函数在此点的梯度

$$\nabla f(1,-1) = \begin{bmatrix} 1 + 6 \times 1 \\ -1 - 4 + 2 \times (-1) \end{bmatrix} = \begin{bmatrix} 7 \\ -7 \end{bmatrix}$$

之前梯度下降89步迭代

$$Step2：x_1 = x_0 - H^{-1}\nabla f = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0.181818 & 0.0909091 \\ 0.0909091 & 0.545455 \end{bmatrix} \begin{bmatrix} 7 \\ -7 \end{bmatrix} = \begin{bmatrix} 0.363636 \\ 2.18182 \end{bmatrix}$$

$Step3$：执行迭代操作，直至满足收敛条件(达到指定迭代次数，或$f(x)$近似收敛到最优解)

设置收敛条件$\left|f(x_n) - f(x_{n-1})\right| < 2 \times 10^{-15}$，程序迭代1步停止，$x_1 = \begin{bmatrix} 0.363636 \\ 2.18182 \end{bmatrix}$

牛顿法的优点和缺点

牛顿法的优点：

- 二阶收敛，收敛速度快，对于正定二次函数一步迭代即达最优解
- 精度高
- 距离最优值较近时，收敛速度极快

牛顿法的局限：

- 牛顿法是局部收敛的，当初始点选择不当时，往往导致不收敛
- Hessian矩阵必须可逆，否则算法困难（非凸问题可能不可逆）
- 迭代算法，每一步都需要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂

Levenberg-Marquardt（列文伯格—马夸尔特方法）结合了梯度下降法和牛顿法。基本思想是利用一阶方法（梯度下降）接近最优点x*，然后利用二阶方法（牛顿法）快速收敛

传统优化方法的优点和局限

- 传统优化方法具有计算简单、优化效率高、可靠性强等优点，是十分重要且应用广泛的优化方法。然而，传统优化方法通常在以下3个方面存在局限性。

1. 单点计算方式大大限制了计算效率的提升。传统优化算法是从一个初始解出发进行迭代计算的，每次迭代只对一个点进行计算，这种方式难以发挥现代计算机高速计算这一特点
2. 改进解的移动方向时容易陷入局部最优，不具备全局搜索能力。传统优化算法在每一次迭代计算过程中，要求解能够向降低目标函数值的方向移动，这种算法不具有“跳出局部”的能力，即算法一旦进入某个局部极值区域，就难以再跳出
3. 对目标函数和约束条件函数的性质有要求，限制了算法的应用范围。传统优化算法要求目标函数和约束条件是连续可导的、凸的，可行域是凸集，但是对于实际问题而言，这些条件通常难以满足

在什么情况下要考虑使用进化优化算法?

- 复杂的问题
  - ✓ 很简单的问题可以通过精确方法进行求解。如果问题的规模很大，参数维度很高，可以用进化优化算法进行求解
- 一个问题可以进行搜索的时间非常有限的情况
- 实际优化问题（不可微分的函数及限制条件；不连续的函数和搜索空间；离散的搜索空间；混合变量（离散变量，连续变量）；高维问题（变量，限制条件）；非线性限制条件；多模态函数；多目标函数；变量具有不确定性；计算量大的问题)
- 无法解析建模的黑箱问题。对函数形式一无所知，仅可以将参数输入仿真系统得到相应的输出

进化优化算法

- 大部分进化优化算法都来自于自然界或生物进化机制的启发

这些算法能够较好地解决实际优化问题的复杂性、约束性、非线性和不确定性

1.自适应：适应变化的环境

2.随机性：在某个限度之内的随机性

3.交流：个体互相交流并从彼此的成功和失败中学习

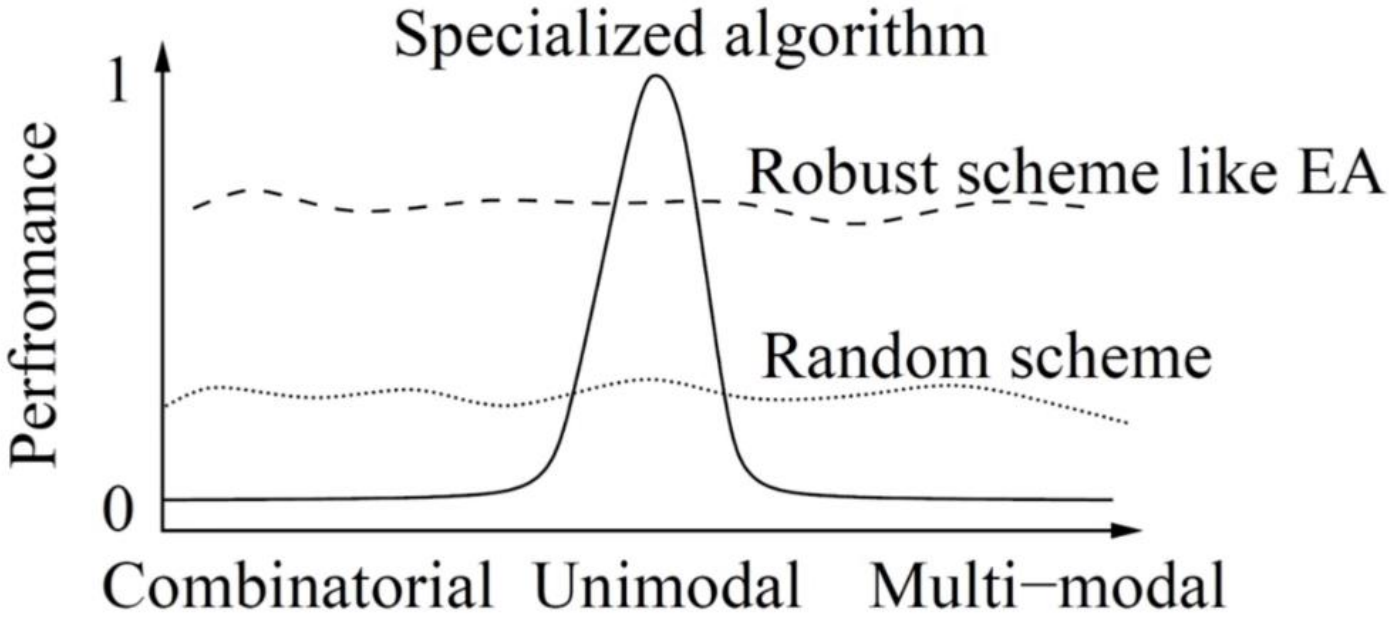
4.反馈：失败提供负反馈，成功提供正反馈

5.探索与开发：探索新的想法或策略，利用已被证明为成功的想法和策略

搜索机制

- 在搜索过程中通常是两个目标之间的平衡
  - ✓ 多样化或探索 （Diversification or exploration）:生成多样化的解在全局尺度下探索搜索空间
  - ✓ 强化或开发（Intensification or exploitation）:已知在某个区域内发现了很好的解，聚焦在这个局部区域进行搜索
- 一个很好的平衡
  - ✓ 快速识别具有高质量解的搜索空间
  - ✓ 不在已完成探索或没有希望获得优质解的空间浪费太多时间
- 每个进化优化算法都利用不同的策略平衡探索与开发。几乎所有的进化优化算法都是爬山法和随机搜索法的精心组合

进化优化算法 vs. 专用化算法

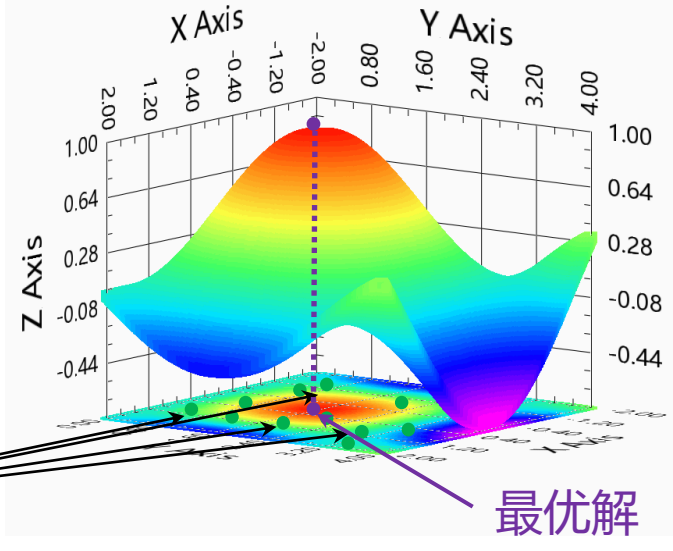


概述

- 引言
- 进化优化算法的分类
- 一个应用优化算法的例子
- 最优化问题的数学模型
- 最优化问题的分类
- 最优化方法的分类
- 进化优化算法通用框架
  - ✓流程图及通用框架
  - ✓优点及局限

进化优化算法

- 来自于自然灵感/启发的算法：模仿自然/生物进化现象或过程
- 基于自然进化+遗传学
  - ✓自然进化：后代（新解）通过交叉或变异等操作生成
  - ✓适者生存：优质解保留，不好的解淘汰。在环境资源（例如食物）有限的条件下，强壮的个体比弱小的个体生存几率更大
  - ✓遗传学：信息经过编码
- 进化优化算法是基于种群的算法
  - ✓许多的解或点



专业术语

优化 Optimization	进化优化算法 Evolutionary Algorithm
决策参量 Decision variables	标记 Marks, 主题 subjects, 位置 position, 基因 gene
解 Solution	种群成员 Population member, 学习者 learner, 染色体chromosome, 子代 child, 父代 parent, 粒子particle, 蜜蜂 bee, 飞蛾 moth, 火焰 flame, 流 stream
一组解 Set of solutions	种群 Population, 类 class, 飞蛾 moths, 火焰 flames, 水体 water body, 群体 swarm
目标函数值 Objective function value	花蜜数量 Nectar amount, 能量 energy, 适应度值 fitness
迭代 Iteration	代 Generation, 循环 cycles

进化优化算法与优化问题

进化优化算法

1）随机生成一个/一组解

2）基于当前一个/一组解的目标函数适应度值，通过“智能”操作生成其他解

$$X = [5 \quad 2 \quad 3]$$

决策变量

目标函数适应度值

$$f = -52$$

优化问题

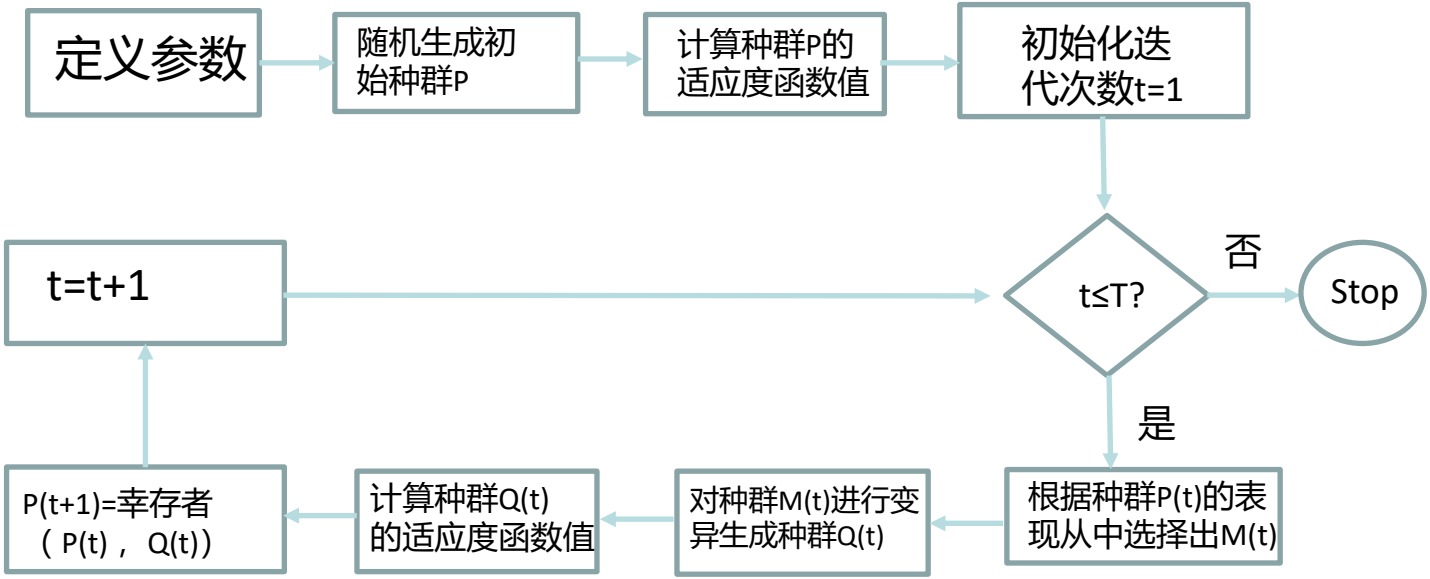
评估目标函数适应度值

最小化

$$f = x_1^2 - 8x_1x_2 + x_3$$

进化优化算法的通用机制

- 问题：适应度函数，决策变量的取值范围
- 算法：种群规模，最大迭代次数（T）



进化优化算法的通用机制

Algorithm 1 Generatlized Framework

1: Solution representation %Genetics

2: Input: t: = 1 (Generation counter), Maximum allowed generation = T

3: Initialize random population (P(t)); %Parent population

4: Evaluate (P(t)); %Evaluate objective, constraints and assign fitness

5: **while** t ≤ T **do**

6: M(t) := Selection(P(t)); %Survival of the fittest

7: Q(t) := Variation(M(t)); %Crossover and mutation

8: Evaluate Q(t); %Offspring population

9: P(t+1) := Survivor(P(t), Q(t)); %Survival of the fittest

10: t := t+1;

11: **end while**

进化优化算法的优点

- 可以应用于没有很好解决办法的复杂问题
  - ✓多模态（Multi-modalities)，不连续，非线性问题
  - ✓离散变量空间
- 特别适合于求解具有多个解的问题
  - ✓多模态优化问题
  - ✓多目标优化问题
- 开发成本低：程序通用性好，只需要调整适应度函数及参数取值范围，就适用于新的优化问题
- 并行实现方式适合于计算量大的复杂问题

进化优化算法的局限

- 在有限的时间内不能保证找到最优解
  - 有限时间和迭代次数情况下不能保证收敛到最优解
  - 但是，随着迭代次数的增加可以渐进收敛到最优解
- 进化算法中参数的取值通常需进行反复试验尝试
  - 参数的自适应选择是一种解决方案
- 基于种群的方式可能会增加运算量

遗传算法

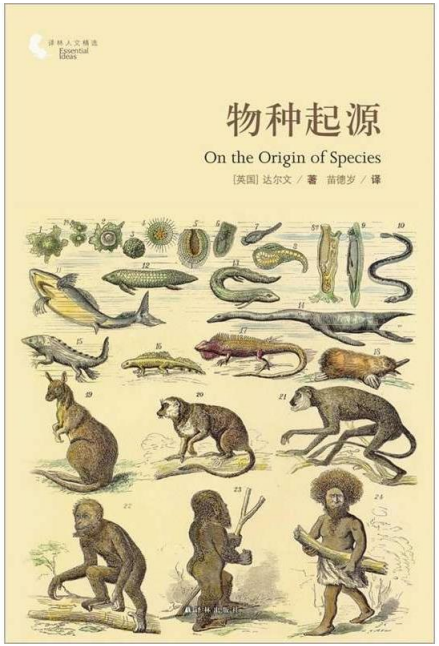
Genetic Algorithm

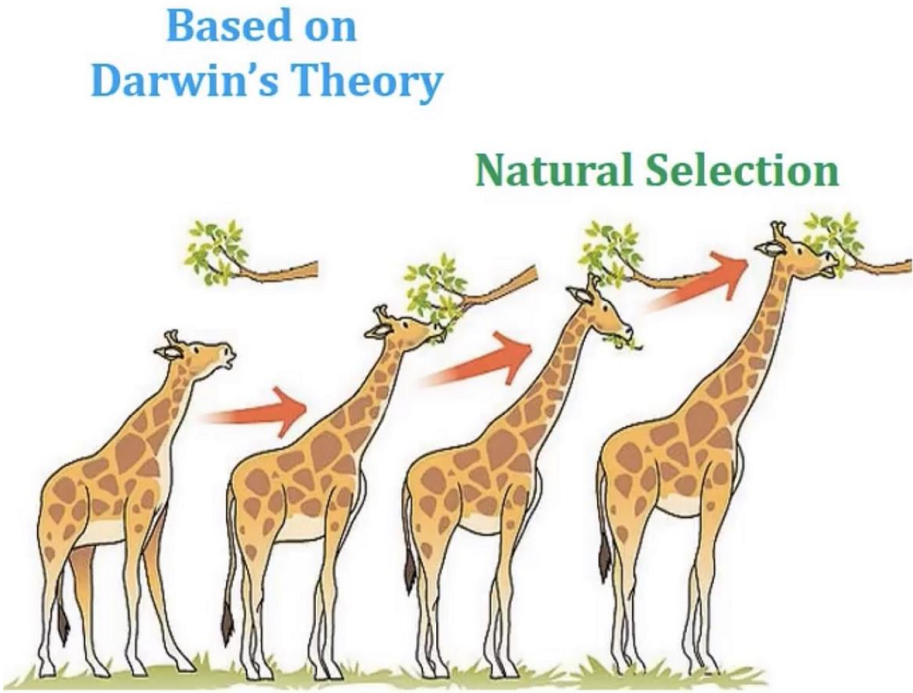
遗传算法简介

- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

物竞天择，适者生存

“有限的资源将所有的生命都逼上生存竞争的战场，只有受到自然偏爱的物种能够存活下来，在自然选择的法则下开始物种起源。”
----- 达尔文 《物种起源》





什么是遗传算法?

- 遗传算法（Genetic Algorithm, GA）是模拟生物在自然环境中的遗传和进化的过程而形成的自适应全局优化搜索算法
- 它借用了生物遗传学的观点，通过自然选择、遗传和变异等作用机制，实现各个个体适应性的提高
- 遗传算法本质是一种并行、高效、全局搜索的方法，它能在搜索过程中自动获取和积累有关搜索空间的知识，并自适应地控制搜索过程以求得最优解
- 用于优化高度复杂的目标函数

1. 在数学上很难建模
2. NP-Hard 问题，计算量极大
3. 包含大量参数（离散 及/或 连续）

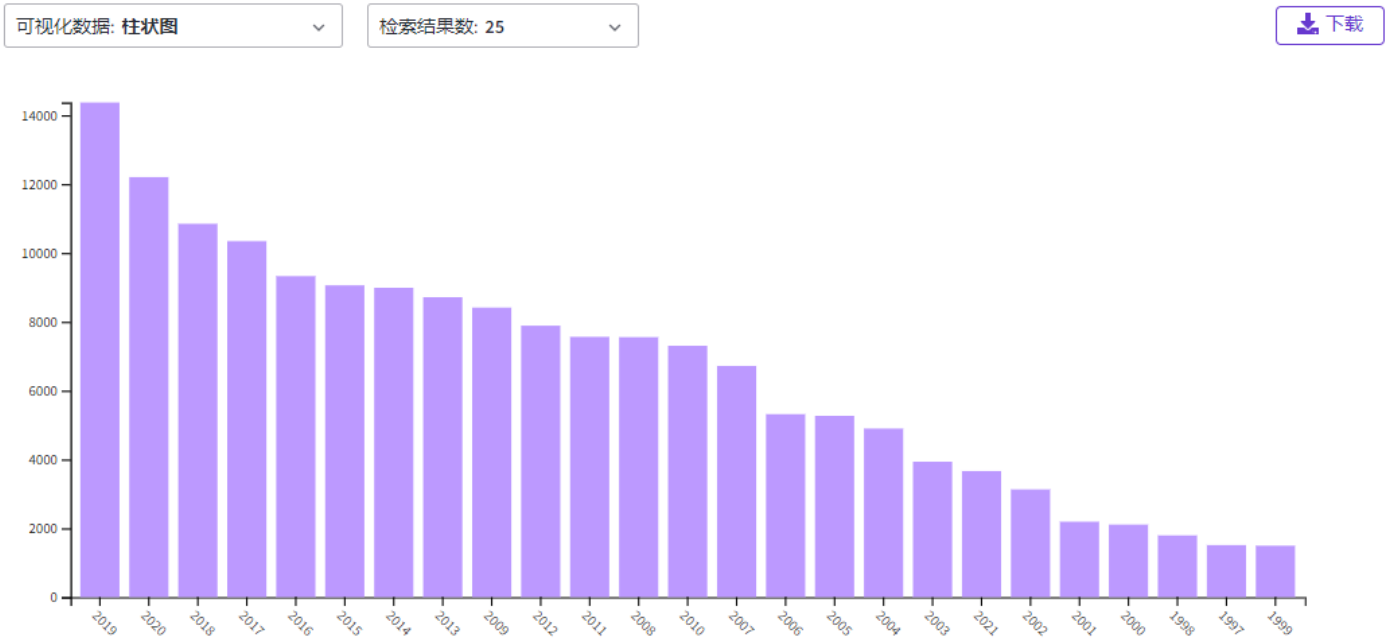
遗传算法简介

- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

遗传算法的发展

- John Holland教授（Michigan University, USA, 1965）首次提出。第一篇GA的文章发表于1975年，起源于20世纪60年代对自然和人工自适应系统的研究
- 20世纪70年代，K.A.De Jong基于遗传算法的思想，在计算机上进行了大量的纯数值函数优化计算试验
- 20世纪80年代，遗传算法由D.E. Goldberg在一系列研究工作的基础上归纳总结而成
- 20世纪90年代，遗传算法作为一种高效、实用、稳健性强的优化技术，发展极为迅速，在机器学习、模式识别、神经网络、控制系统优化及社会科学等不同领域得到广泛应用
- 21世纪，以不确定性、非线性、时间不可逆为内涵的复杂性科学成为一个研究热点，掀起遗传算法研究和应用热潮

遗传算法的发展


可视化数据: 柱状图
检索结果数: 25
下载
14000
12000
10000
8000
6000
4000
2000
0
2019
2020
2018
2017
2016
2015
2014
2013
2009
2012
2011
2008
2010
2007
2006
2005
2004
2003
2021
2002
2001
2000
1998
1997
1999

遗传算法的发展


可视化数据: 树状图
检索结果数: 25
下载
166,671
Mathematics
112,002
Computer Science
86,248
Engineering
45,877
Mathematical Computational Biology
37,782
Genetics Heredity
36,986
Business Economics
33,660
Biochemistry Molecular Biology
29,918
Automation Control Systems
24,506
Science Technology Other Topics
19,077
Communication
16,389
Energy Fuels
15,317
Life Sciences Biomedicine Other Topics
14,539
Robotics
13,859
Telecommu
11,821
Biotechn Applied Microbio
11,541
Physics
11,459
Radiology Nuclear Medicine Medical
11,349
Chemistry
10,612
Instrumen Instrumen
10,357
Biophysic
9,696
Cell Biology
9,287
Medical Informatics
8,520
Operations Research
8,374
Environmen Sciences
7,227
Materials Science

遗传算法的研究内容和研究方向

遗传算法简介

- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

遗传算法的生物学基础

GA算法的基本原理基于两个基本生物学过程

1. 孟德尔遗传学说（Genetics）： Gregor Johan Mendel (1865)

1865年02月08日 奥地利遺傳學家 Gregor Johann Mendel 首次發表豌豆遺傳學研究成果

大大大

BY GENEONLINE ON 2021 年 2 月 8 日　　基因史上的今天

奧地利遺傳學家 Gregor Johann Mendel（1822年7月20日－1884年1月6日）是一位天主教聖職人員，遺傳學的奠基人，約從1856年到1863年，他進行了8年的豌豆雜交實驗，豌豆通常是自花受精的，但是孟德爾人工地將一個高的同一個矮的品種進行雜交，獲得了只產生高植株的種子，孟德爾把他的實驗結果解釋為每一植株都具有兩個決定高度性狀的因數，每一親體賦予一個因數，高的因數是顯性，而矮的因數是隱性，因此雜交後第一代的植株全都是高的，當這一代自花受精後，這些因數在子代中排列可以是兩個高因數在一起，或者兩個矮因數在一起，或者一高一矮，一矮一高，前兩種組合將會繁育出同樣的後代，各自生出全是高的或全是矮的植物，而後面的兩種組合則將以三與一之比生出高的或矮的植物來，1865年，時年42歲的他於Brünn自然科學學會首度發表他在豌豆進行的遺傳學研究成果，但直到他過世後16年，其論述才被再次發掘並受到重視。

參考文獻

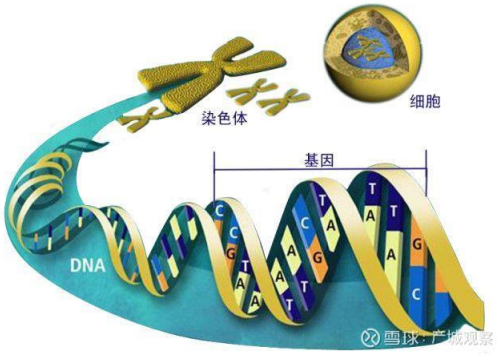
http://www.biography.com/people/gregor-mendel-39282#!

- 遗传是指父代与子代之间，在性状上存在的相似现象
- 变异是指父代与子代之间，以及子代的个体之间，在性状上存在的差异现象
- 遗传能使生物的性状不断地传送给后代，保持了物种的特征
- 变异能使生物的性状发生改变，从而适应新的环境并不断地向前发展

遗传算法的生物学基础

1. 孟德尔遗传学说（Genetics）： Gregor Johan Mendel (1865)

- 遗传物质的主要载体是染色体，基因是有遗传效应的片段，它存储这遗传信息，可以准确复制，也能够发生突变
- 生物体自身通过对基因的复制和交叉，使其性状的遗传得到选择和控制



- 通过基因重组、基因变异和染色体在结构和数目上的变异产生丰富多彩的变异现象
- 生物的遗传特征使生物界的物种能够保持相对的稳定；生物的变异特征使生物产生新的性状，推动生物的进化和发展

遗传算法的生物学基础

2. 达尔文进化论（Evolution）： Charles Darwin (1875)

- 自然选择学说认为适者生存，生物要存活下去，就必须进行生存斗争
- 生存斗争包括种内斗争，种间斗争以及生物与环境之间的斗争
- 在生存斗争中，具有有利变异的个体容易存活下来，并且有更多的机会将有利变异传给后代
- 具有不利变异的个体就容易被淘汰，产生后代的机会也将少的多
- 达尔文把这种在生存斗争中适者生存、不适者淘汰的过程称为自然选择

遗传算法的生物学基础

生物遗传和进化的规律

1. 生物的所有遗传信息都包含在其染色体中，染色体决定了生物的性状。染色体是由基因及其有规律的排列所构成的

2. 生物的繁殖过程是由其基因的复制过程完成的。同源染色体的交叉或变异会产生新的物种，使生物呈现新的性状

3. 对环境适应能力强的基因或染色体，比适应能力差的基因或染色体有更多的机会遗传到下一代

遗传算法的生物学基础

生物遗传和进化的规律的四个主要假定：

1. 信息传播：一个子代具有父代的很多特征（例如，信息从父代传递给子代）遗传[Heredity]
2. 种群多样性：在下一代中特征具有不同的变化。多样性[Diversity]
3. 生存竞争：在子代中只有小部分比例能够幸存。选择[Selection]
4. 最优秀者的生存：子代存活依赖于他们继承的特征。排序[Ranking]

遗传算法简介

- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

遗传算法的基本概念

- 遗传算法使用群体搜索技术，将种群代表一组问题解，通过对当前种群施加选择、交叉和变异等一系列遗传操作来产生新一代的种群，并逐步使种群进化到包含近似最优解的状态

遗传学术语	遗传算法术语
个体	可行解
染色体	可行解的编码
基因	可行解编码的分量
基因形式	遗传编码
适应度	评价函数值
选择	选择操作
交叉	交叉操作
变异	变异操作

遗传算法的基本概念

1. 群体和个体

   群体是生物进化过程中的一个集团，表示可行解集

   个体是组成群体的单个生物体，表示可行解

2. 染色体和基因

   染色体是包含生物体所有遗传信息的化合物，表示可行解的编码

   基因是控制生物体某种性状（即遗传信息）的基本单位，表示可行解编码的分量

3. 遗传编码

   遗传编码将优化变量转化为基因的组合表示形式，优化变量的编码机制有二进制编码、十进制编码（实数编码）等

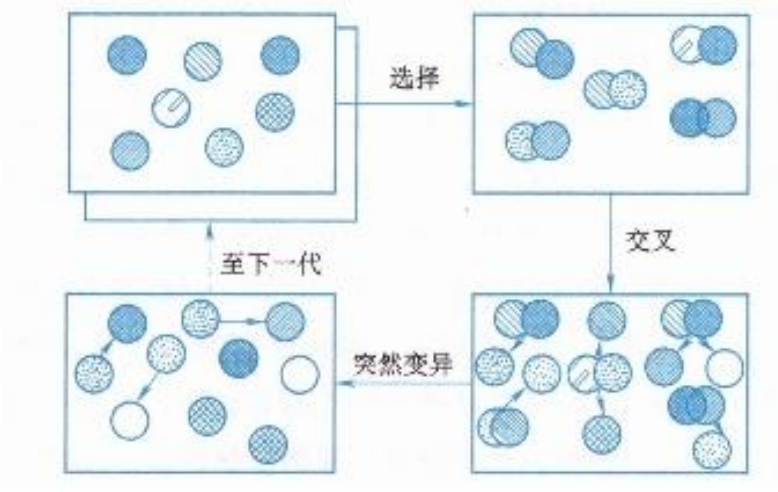
4. 适应度

   适应度即生物群体中个体适应生存环境的能力。在遗传算法中，用来评价个体优劣的数学函数称为个体的适应度函数

遗传算法的基本概念

5. 遗传操作

遗传操作是优选强势个体的“选择”、个体间交换基因产生新个体的“交叉”、个体基因信息突变而产生新个体的“变异”这三种变换的统称



遗传算法的基本操作过程

遗传算法简介

- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

遗传算法的特点

- 遗传算法是模拟生物在自然环境中遗传和进化的过程而形成的一种并行、高效、全局搜索的方法。具有如下特点：

1. 遗传算法以决策变量的编码作为运算对象。这种对决策变量的编码处理方式，使得在优化计算过程中可以借鉴生物学中染色体和基因等概念，模仿自然界中生物的遗传和进化等的机理，方便地应用遗传操作算子

2. 遗传算法直接以目标函数值作为搜索信息，不需要目标函数值的导数值等其他一些辅助信息。对于实际应用中无法或很难求导的目标函数的优化和组合优化问题，遗传算法显示高度的优越性，避开了函数求导这一障碍

遗传算法的特点

- 遗传算法是模拟生物在自然环境中遗传和进化的过程而形成的一种并行、高效、全局搜索的方法。具有如下特点：

3. 遗传算法同时使用多个搜索点的搜索信息。遗传算法对最优解的搜索过程，是从一个由很多个体所组成的初始群体开始的，而不是从单一的个体开始的。对这个群体所进行的选择、交叉、变异等运算，产生出新一代的群体，其中包括了很多群体信息。这些信息可以避免搜索一些不必搜索的点，相当于搜索了更多的点，这是遗传算法所特有的一种隐含并行性

遗传算法的特点

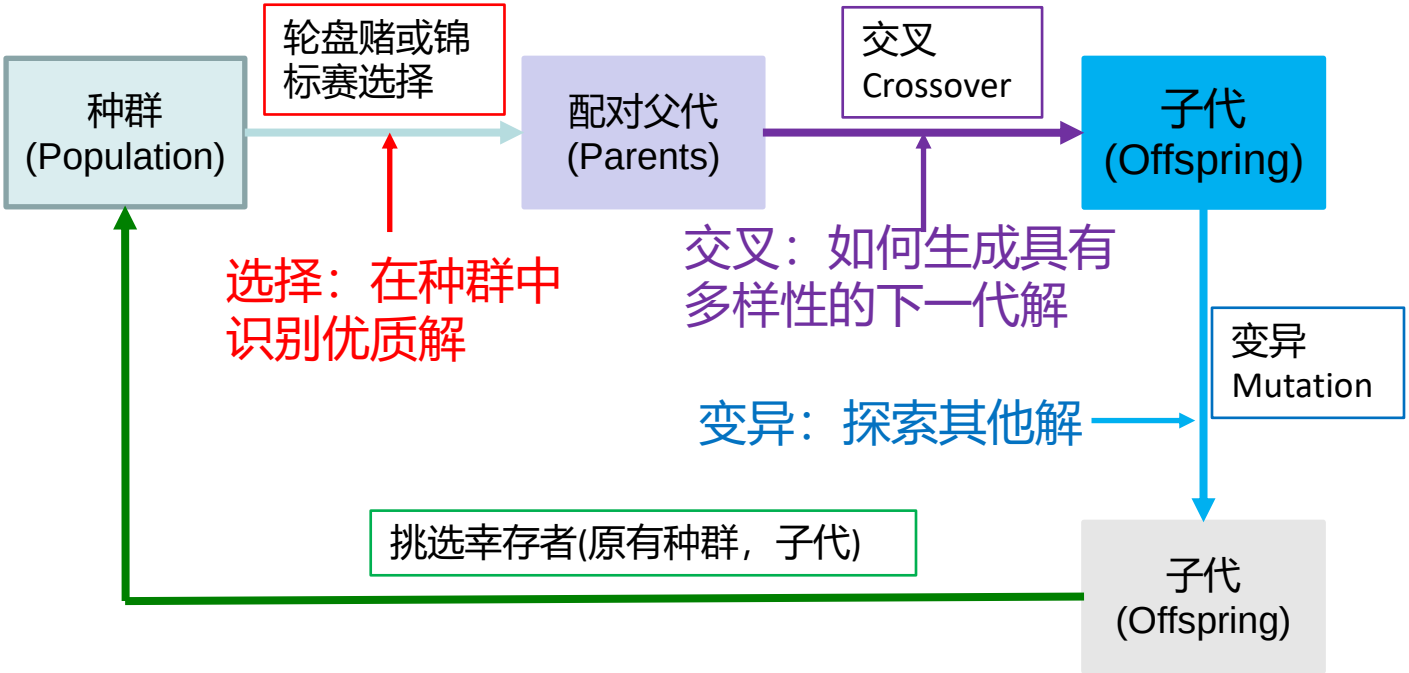
4. 遗传算法是一种基于概率的搜索技术。遗传算法属于自适应选择概率搜索技术，其选择、交叉、变异等运算都是以一种概率的方式进行的，从而增加了其搜索过程的灵活性。虽然这种概率特性也会使种群中产生一些适应度不高的个体，但随着进化过程的进行，新的种群中总会更多地产生出优良的个体。与其他一些算法相比，遗传算法的稳健性使得参数对其搜索效果的影响尽可能小

5. 遗传算法具有自组织、自适应和自学习等特征。当遗传算法利用进化过程获得信息自行组织搜索时，适应度大的个体具有较高的生存概率，并获得更适应环境的基因结构。同时，遗传算法具有可扩展性，易于同别的算法相结合，生成综合双方优势的混合算法

遗传算法简介

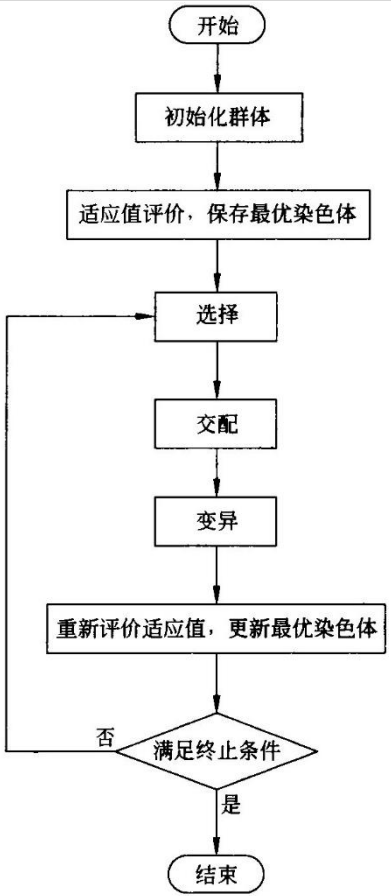
- 什么是遗传算法?
- 遗传算法的发展和研究方向
- 遗传算法的生物学基础
- 遗传算法的基本概念
- 遗传算法的特点
- 遗传算法的基本流程

遗传算法的基本流程



收敛条件(Convergence)：如何确定算法停止运行判据

遗传算法的伪代码




```
/*
  P(t)表示某一代的群体,t为当前进化代数
  Best表示目前已找到的最优解
*/
Procedure GA
begin
   t←0;
   initialize(P(t));          //初始化群体
   evaluate(P(t));            //适应值评价
   keep_best(P(t));           //保存最优染色体
   while(不满足终止条件) do
   begin
     P(t)←selection(P(t));    //选择算子
     P(t)←crossover(P(t));    //交配算子
     P(t)←mutation(P(t));     //变异算子
     t←t+1;
     P(t)←P(t-1);
     evaluate(P(t));
     if (P(t)的最优适应值大于 Best 的适应值)
       //以 P(t)的最优染色体替代 Best
       replace(Best);
     end if
   end
end
```

遗传算法的基本流程

- 用遗传算法求解问题的基本步骤：

1. 初始化规模为N的群体，其中染色体每个基因的值采用随机数产生器生成并满足问题定义的范围。当前进化代数Generation=0
2. 采用评估函数对群体中所有染色体进行评价，分别计算每个染色体的适应值，并保存适应值最大的染色体Best
3. 采用轮盘赌选择算法对群体染色体进行选择操作，产生规模同样为N的种群
4. 按照概率P_c从种群中选择染色体进行交配。每两个进行交配的父代染色体，交换部分基因，产生两个新的子代染色体，子代染色体取代父代染色体进入新种群。没有进行交配的染色体直接复制进入新种群

遗传算法的基本流程

- 用遗传算法求解问题的基本步骤：

5. 按照概率P_m对新种群中染色体的基因进行变异操作。发生变异的基因数值发生改变。变异后的染色体取代原有染色体进入新群体，未发生变异的染色体直接进入新群体
6. 变异后的新群体取代原有群体，重新计算群体中各个染色体的适应值。倘若群体的最大适应值大于Best的适应值，则以该最大适应值对应的染色体替代Best
7. 当前进化代数Generation加1。如果Generation超过规定的最大进化代数或Best达到规定的误差要求，算法结束；否则返回第3步