数据分析与处理技术——优化建模技术

商学院 徐宁

1.函数极值

单变量函数极值

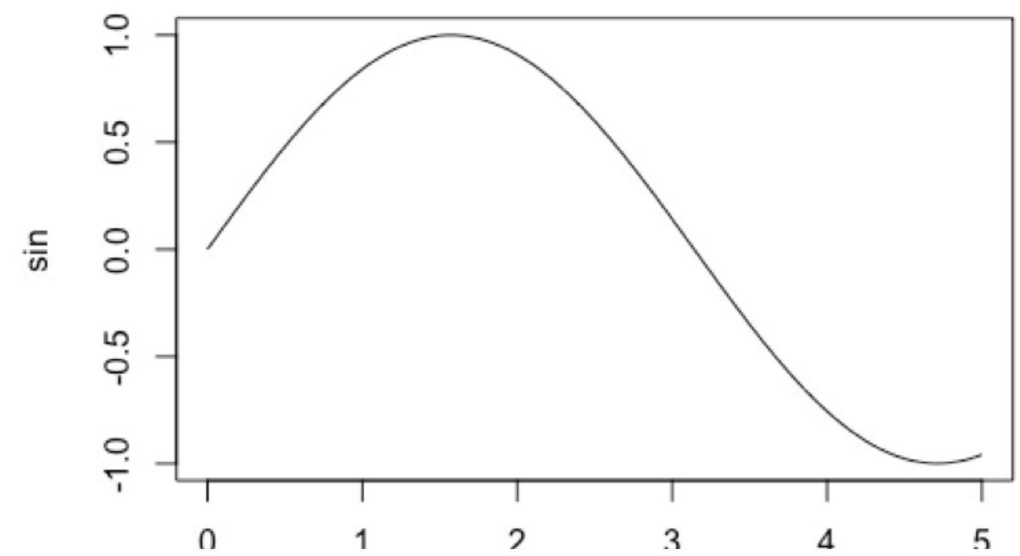
多变量函数极值

*遗传算法

函数区间图像

- plot也可以绘制函数图像，以函数作为参数输入，同时给出范围：

```
> plot(sin,0,5)
```

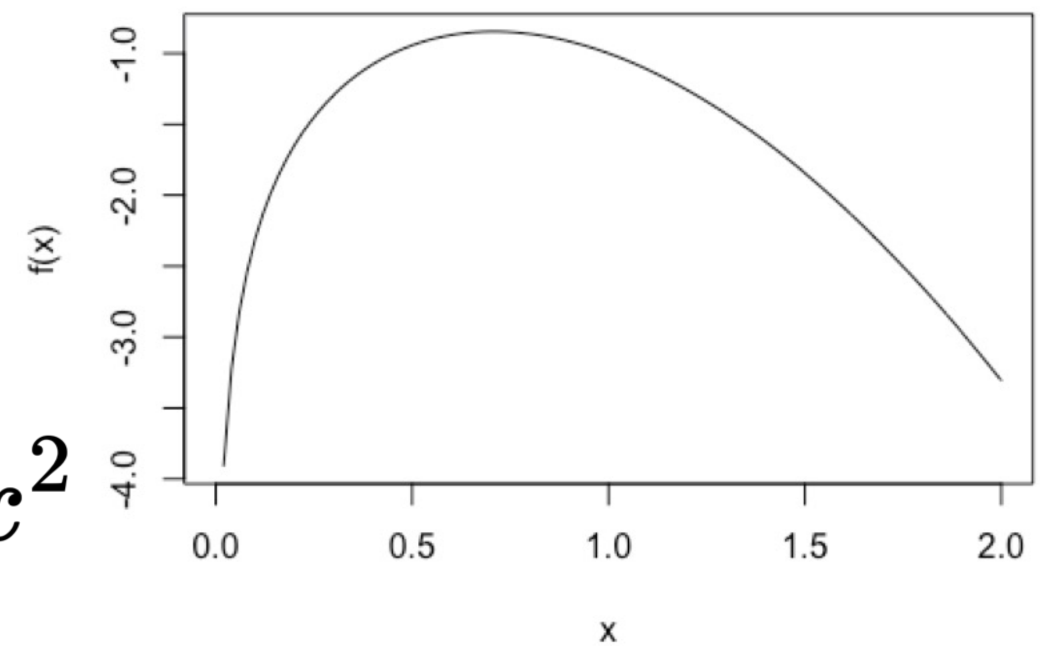


- 曲线绘制函数与plot类似：

```
> f=function(x) log(x)-x^2
```

```
> curve(f,0,2)
```

$$f(x) = \ln(x) - x^2$$



单变量函数极值

- 计算机利用高速运算可以对简单函数做遍历搜索来找出极致位置，例如函数：

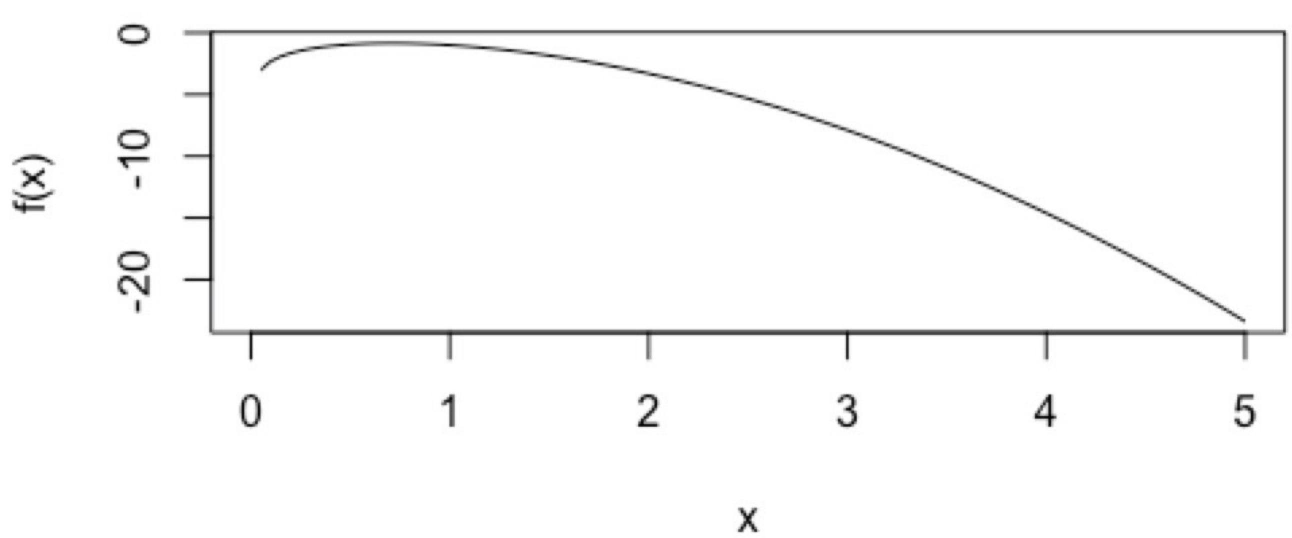
$$f(x)=\ln(x)-x^2$$

```
> optimize(f,c(0,5))
$minimum
[1] 4.999922

$objective
[1] -23.3898

> optimize(f,c(0,5),maximum = T)
$maximum
[1] 0.7070898

$objective
[1] -0.8465736
```

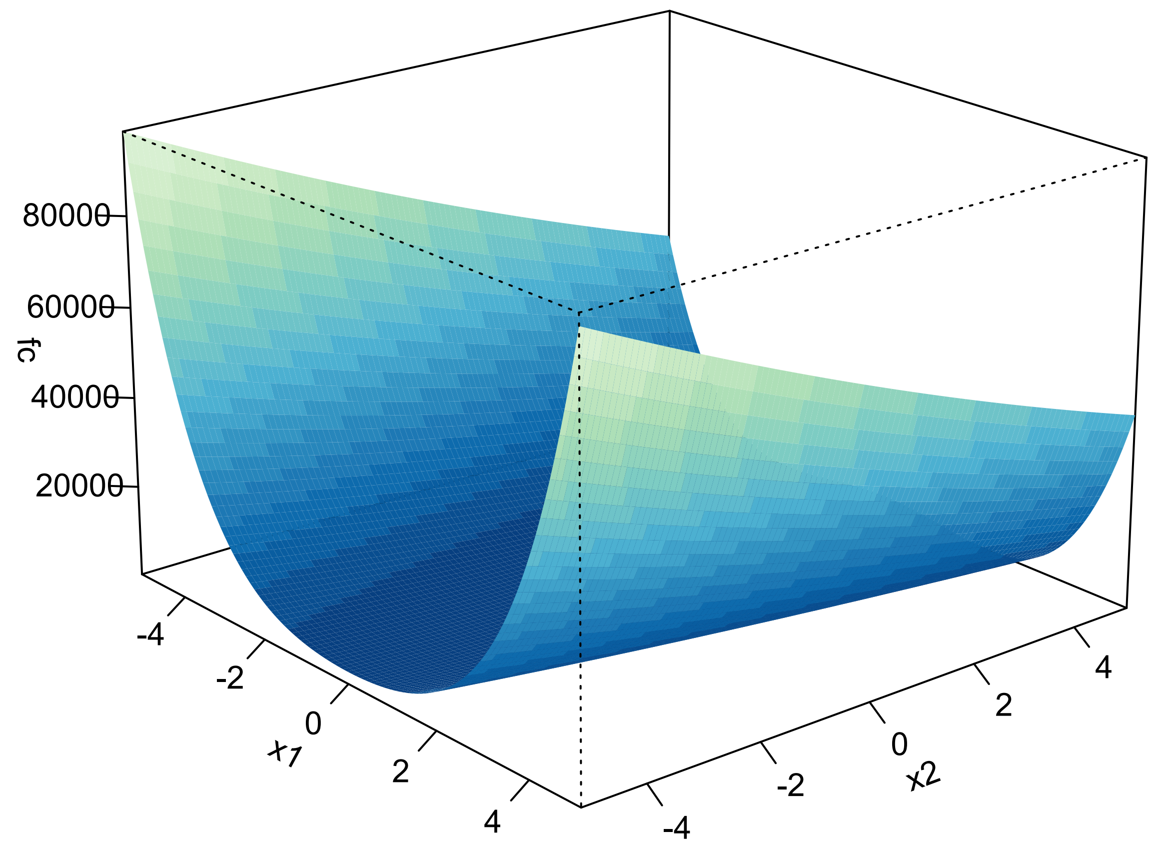


多变量凸函数极值

凸函数的极值搜索，例：

$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

```
f<-function(x) {
      y<-100*(x[2]-x[1]^2)^2+(1-x[1])^2
      return(y)
}
```



梯度搜索算法

```
solution=optim(c(2,2),fn = f,
               method="L-BFGS-B",
               upper=c(10,10),
               lower = c(-10,-10),
               control=list(fnscale=T))
```

搜索策略参数

optim中的method参数

- Nelder-Mead:默认方法，稳健性较好但比较慢的方法。
- BFGS：拟牛顿法(变尺度法)
- CG：共轭梯度法
- SANN：模拟退火法的变体方法
- Brent：仅用于一维优化

optim中的control参数

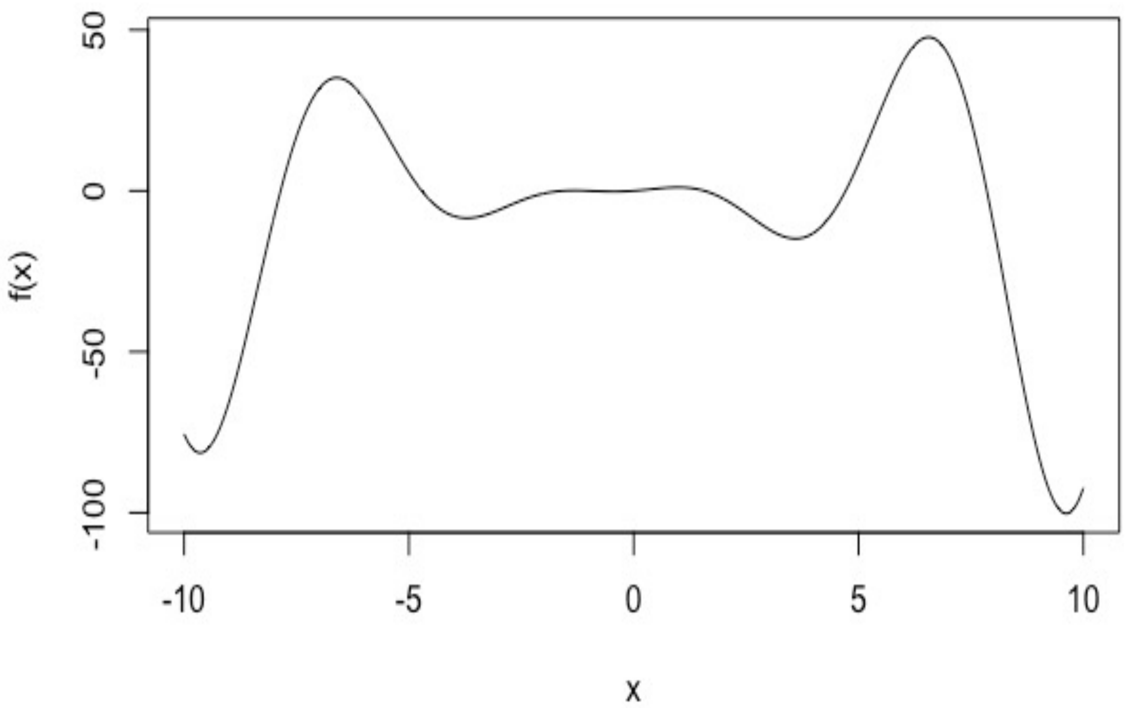
- fnscale:逻辑参数，True为最小化问题，False为最大化问题

*非凸函数极值算法

单变量非凸函数

```
f <- function(x)  (x^2+x)*cos(x)

curve(f,from=-10,to=10,n=1000)
```



函数f作为遗传算法的适应度函数放入算法函数中。

工具包GA中的函数ga()执行遗传算法，搜索适应度函数的极值。

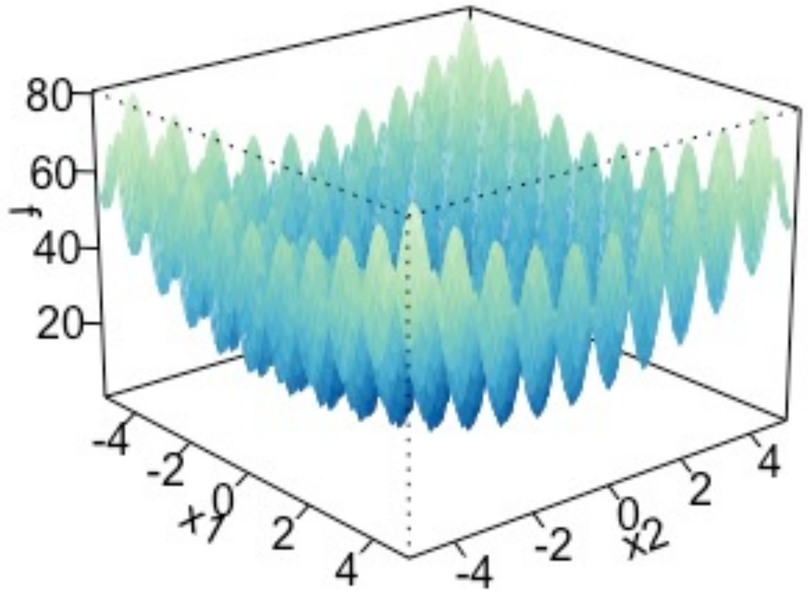
```
library(GA)
ga(type = "real-valued",
   fitness = f,
   lower = -10,
   upper = 10)
```

*遗传算法的参数

多维变量非凸函数

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10[\cos(2\pi x_1) + \cos(2\pi x_2)]$$

```
Rastrigin <- function(x1, x2)
{
  20 + x1^2 + x2^2 - 10*(cos(2*pi*x1) + cos(2*pi*x2))
}
```



遗传算法采用随机搜索策略，与函数性质无关,只要能够构成适应度函数即可。

```
ga(type = "real-valued",
         fitness =  function(x) -Rastrigin(x[1], x[2]),
         lower = c(-5.12, -5.12),  #上限值
         upper = c(5.12, 5.12),    #下限值
         popSize = 50,             #种群数量
         maxiter = 1000,           #迭代次数
         monitor = NULL)           #关闭过程追踪
```

2.线性规划求解

求解器简介

模型矩阵形式

Lpsolve

大规模运筹求解器

解决较大规模的运筹优化问题常常需要用到规划求解器。

- 求解器主要分为商业和开源两大类

求解器	商业	GUROBI	Cplex	Cardinal	Xpress
	开源	Ortools	lpsolver	GLPK	SCIP

- 求解模型的常用编程语言

常用编程工具	通用	Python	MATLAB	R	Julia
		C/C++	JAVA		
	专用	AMPL	LINGO	GAMS	NEOS

线性规划问题引入

- 某工厂生产3种型号产品，分别销售价格3万、2万、1万元。三种型号产品需要原材料A B，第一种产品需要耗费材料A 1吨、B 2吨，第二种产品消耗A 2吨、B 4吨；第三种消耗A 2.3吨。A和B两种材料每月的供应量最多为19吨和14吨。

- 问题：每月生产3种产品 x_1 x_2 x_3分别多少才能获得最大收益。

$$\max \quad 3x_1+2x_2+x_3$$

$$s.t.\begin{cases} x_1+2x_2+2.3x_3 \le 19 \\ 2x_1+4x_2 \le 14 \\ x_1, x_2, x_3\text{为非负实数} \end{cases}$$

标准模型的矩阵化表示

$$\max \quad 3x_1+x_2+3x_3 \xrightarrow{\text{目标函数系数}} [3 \quad 1 \quad 3]$$

$$s.t.\begin{cases} -x_1+2x_2+\ x_3 \le 4 \\ 4x_2-3x_3 \le 2 \\ x_1-\ 3x_2+2x_3 \le 3 \\ x_1,\ \ x_3\text{为非负实数} \\ x_2\ \text{为非负整数} \end{cases}$$

$$\xrightarrow{\text{主约束}} \begin{bmatrix} -1 & 2 & 1 \\ 0 & 4 & -3 \\ 1 & -3 & 2 \end{bmatrix} \begin{matrix} \le \\ \le \\ \le \end{matrix} \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$$

$$\xrightarrow{\text{类型约束}} \text{‘I’ ‘C’ ‘I’}$$

```
obj <- c(3, 1, 3)
mat <- matrix(c(-1, 0, 1, 2, 4, -3, 1, -3, 2), nrow = 3)
dir <- c("<=", "<=", "<=")
rhs <- c(4, 2, 3)
types <- c("I", "C", "I")
max <- TRUE
```

矩阵建模方式(lpSolve)

标准规划模型的矩阵形式各部分命名如下：

sense	Max				direction	RHS
	Obj	3	1	3		
constraints	C1	-1	2	1	<=	4
	C2	0	4	-3	<=	2
	C3	1	-3	2	<=	3
	Type					

分别将各部分写入程序，变成计算机“认识”的形式。

Solver工作原理

R环境

R的语法规则

make.lp

……

solve

……

get.objective

```
#加载工具包
library(lpSolve)
library(lpSolveAPI)
```

API转换信息

Solver

二进制代码

组建模型

开始解算

存储结果

R提供编程语法并将代码按照solver的功能进行“翻译”，求解过程在solver中，计算结果回传到R环境内的变量

建模方法

通过make.lp构建一个模型框架，下边是行方式添加约束，逐步将md模型变量填充配置。

```
md=make.lp(0,3)                           创建模型框架
set.objfn(md,c(3,1,3))                    对md设置目标函数
add.constraint(md,c(-1,2,1),"<=",4)       对md添加约束
add.constraint(md,c(0,4,-3),"<=",2)       对md添加约束
add.constraint(md,c(1,-3,2),"<=",3)       对md添加约束
set.type(md,1:3,"integer")                设置md的决策变量类型， 1，2，3 是integer类型
lp.control(md,sense="max")                修改md模型的控制参数，其中sense设为求最大化问题
print(md)                                 打印md当前状态
```

模型配置完毕，用solve函数启动求解器进行解算，之后便可以对md变量提取计算结果。

```
solve(md)                                 启动求解器解算solve(md)
get.objective(md)                         获取md模型的目标函数最优值
get.variables(md)                         获取决策变量结果
get.dual.solution(md)                     获取对偶模型的结果
```

建模常用语句

- set.type对应的决策变量类型integer,binary,real
- make.lp预置模型基本结构：约束数量和决策变量数量。
- set类函数只能填充模型参数，不能改变模型结构。
- add类函数可以扩充模型结构，如添加约束扩充了约束数。
- get类函数只能在solve解算模型之后提取计算结果。