

计算机科学与技术学院
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

Matlab语言

—最优化方法

Matlab基础: **MATLAB**=**MAT**rix+**LAB**oratory=矩阵实验室

优点: 简单、易用, 各个学科都可以使用!

在欧美大学里, 诸如应用代数、线性代数、数理统计、自动控制、数字信号处理、模拟与数字通信、时间序列分析和动态系统仿真等课程的教科书都把MATLAB作为教学内容。MATLAB已成为攻读学位的大学生、硕士生、博士生必须掌握的基本工具。

缺点: 它和其他高级程序相比, 程序的执行速度较慢。

Matlab Main Toolbox——matlab主工具箱

Control System Toolbox——控制系统工具箱

Communication Toolbox——通讯工具箱

Financial Toolbox——财政金融工具箱

System Identification Toolbox——系统辨识工具箱

Fuzzy Logic Toolbox——模糊逻辑工具箱

Higher-Order Spectral Analysis Toolbox——高阶谱分析工具箱

Image Processing Toolbox——图象处理工具箱

LMI Control Toolbox——线性矩阵不等式工具箱

Model predictive Control Toolbox——模型预测控制工具箱

μ -Analysis and Synthesis Toolbox—— μ 分析工具箱

Neural Network Toolbox——神经网络工具箱

Optimization Toolbox——优化工具箱

Partial Differential Toolbox——偏微分方程工具箱

Robust Control Toolbox——鲁棒控制工具箱

Signal Processing Toolbox——信号处理工具箱

Spline Toolbox——样条工具箱

Statistics Toolbox——统计工具箱

Symbolic Math Toolbox——符号数学工具箱

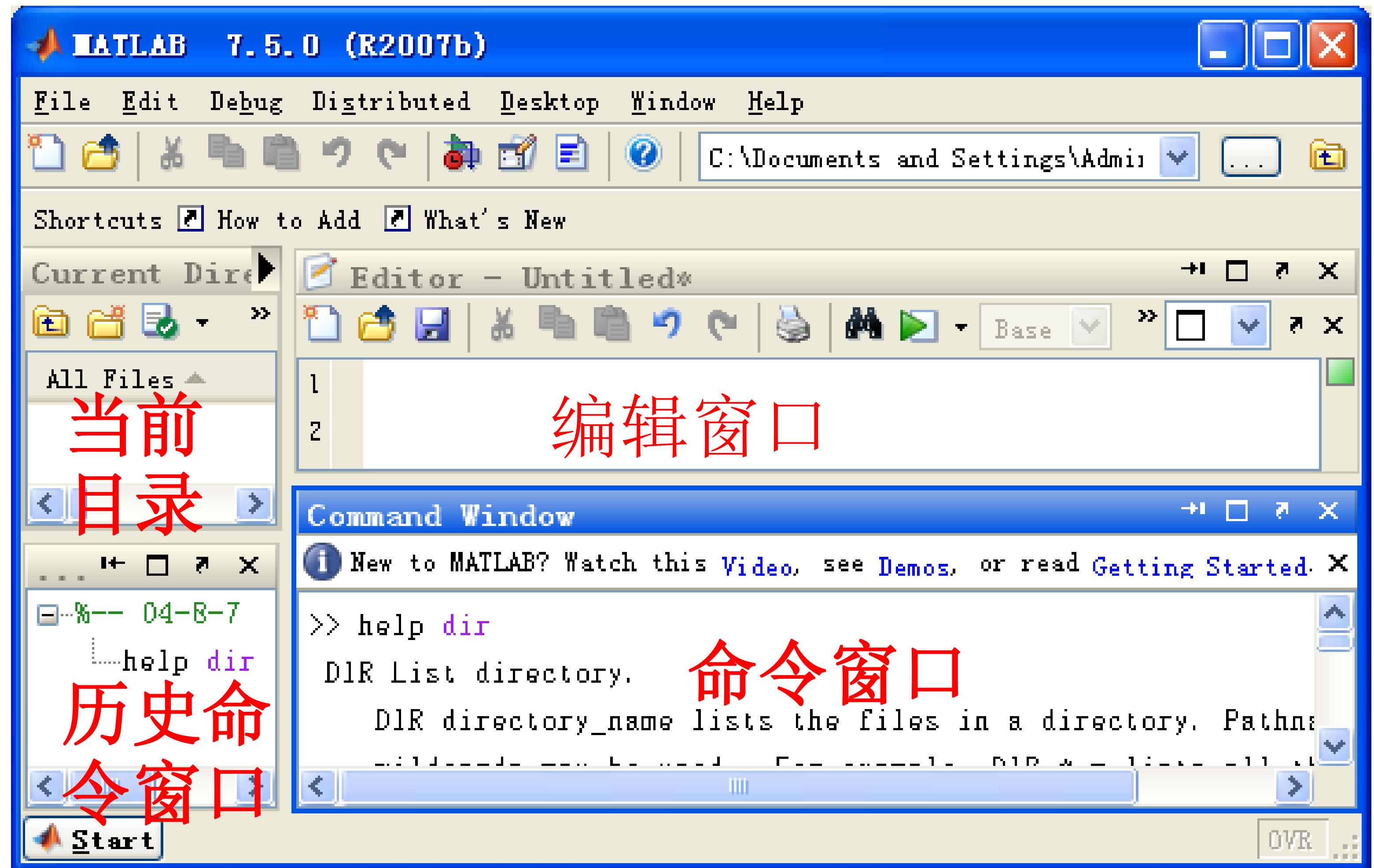
Simulink Toolbox——动态仿真工具箱

Wavele Toolbox——小波工具箱



Matlab基础：工作界面

Windows
和linux下
都有相应
安装版本



Matlab基础：常用命令

命令1：帮助命令help，用于查询函数的用法

例：help max

```
>> help max
max      Largest component.
        For vectors, max(X) is the largest element in X. For matrice
max(X) is a row vector containing the maximum element from e
```

命令2：清屏clc，使命令窗口的显示消失

例：clc

```
fx >> clc|
```

命令4：保存内存变量 Save 文件名

命令5：调用内存变量 Load 文件名

命令3：清除内存变量clear



Matlab基础：标点的含义

(1) 分号;

区分行以及取消运行显示等。

例：A和B的区别。

A=[1, 2;3, 4]与B=[1, 2;3, 4];

```
>> A=[1, 2;3, 4]

A =

     1     2
     3     4

>> B=[1, 2;3, 4];

fx >>
```

(2) 逗号,

区分列及函数参数分隔符

例：A=[1, 2;3, 4],

B=[1, 4, 3;3, 2, 1;4, 5, 6]

(3) 小括号()：指定运算过程的先后次序等。例：

x=0.5;

y=sin(x)/(2+cos(x))

z= sin(x)/2+cos(x)

(4) 方括号[]：矩阵定义标志等。见上。



Matlab基础：标点的含义

(5) 输入三个点表示续行号...

例：

`y=2+3`

也可写为

```
>> y=2+...  
3  
  
y =  
  
5
```

(6) 注释标记百分号%：

例：

`b=sin(x); %正弦函数`

(7) 字符串用单引号' '

例：

`a='Hello Matlab'`

```
>> a='Hello Matlab'  
  
a =  
  
Hello Matlab
```

(8) 冒号：

有多种应用功能，常用于矩阵操作
如：选取矩阵的所有行、列；矩阵定义



Matlab基础：变量、常量

变量：不需要定义，**直接用！**

常量：常见的常量Matlab也已经帮你**设置好了**，**直接用！**


例如：

i,j: 虚数单位；

pi: π ；

NaN: 表示不定值,比如0/0；

inf: 无穷大（infinite），比如1/0。



```
>> pi  
  
ans =  
  
3.1416
```



Matlab基础：运算符

算术操作符：乘方[^] 左除\ 右除/

矩阵如果在运算符前加个[.]，表示点到点运算，对应元素相运算。

例如：

A=[1,2;3,4]

<pre>>> A.^2</pre>	<pre>>> A^2</pre>
<pre>ans =</pre>	<pre>ans =</pre>
<pre>1 4</pre>	<pre>7 10</pre>
<pre>9 16</pre>	<pre>15 22</pre>

猜猜下面的结果分别为什么

A=[1,2;3,4]

B=[1,2;3,4]

A.*B

A*B

逻辑运算符：

- (1) ==：等于。
- (2) ~=：不等于。
- (3) <：小于。
- (4) >：大于。



Matlab基础：运算符

例：

已知水的黏度随温度的变化公式为 $\mu = \mu_0 / (1 + at + bt^2)$

其中 $\mu_0 = 1.785 \times 10^{-3}$, $a = 0.03368$, $b = 0.000221$,

求水在 0, 20, 40, 80℃ 时的黏度。

程序如下：

```
miu0=1.785e-3;  
a=0.03368;  
b=0.000221;  
t=0:20:80  
miu=miu0./(1+a*t+b*t.^2)
```

运行后的结果为：

miu =

0.0018 0.0010 0.0007 0.0005 0.0003



Matlab基础： 向量(一维数据)的生成和基本运算

●向量的生成

- (1) 直接输入：如 $a=[1,2,5,3]$
- (2) 利用冒号表达式生成：
如： $b=[2:2:10]$,此时 $[\]$ 可省略，结果为： $b = 2 \quad 4 \quad 6 \quad 8 \quad 10$
- (3) 线性等分向量生成： $y=\text{ linspace}(x1,x2,n)$,生成 n 维向量，
使得 $y(1)=x1,y(n)=x2$ 。如： $y=\text{ linspace}(1,7,3)$ ，结果为： $y = 1 \quad 4 \quad 7$

●向量的基本运算

- (1) 向量的加减：用 $+$ 、 $-$ 。同维向量才可以加、减。相应元素加减
- (2) 向量与数可以加、减和乘。用 $+$ 、 $-$ 和 $*$ 。数与向量的每个元素进行作用
- (3) 向量与数可以相除，向量除以数为：向量/数，数除以向量用点除：数./向量



Matlab基础： 矩阵的生成和基本运算

●矩阵的生成

- (1) 直接输入： 如如： $a=[1,3,4;4,3,2]$
- (2) 创建M文件输入大矩阵:当矩阵很大时,直接输入显得很笨,出错不易修改.我们可以编写一个M文件,M文件的扩展名必须是m
例如： 编写一个名为matrix.m(名字任意)的M文件如下：
 $mat=[1, 2, 3, 3;3, 4, 5, 1;3, 2, 1, 4;8, 9, 7, 5]$
在命令窗口中输入matrix, 就会运行该文件. 查看矩阵的结构可用size(mat)

●矩阵的基本运算

- (1) +、-、*： 加、减、乘运算
- (2) 矩阵求逆： inv(A) 为A的逆(inverse)
- (3) 求转置矩阵： A'
- (4) 求矩阵的行列式： $\det(A)$ (determinant是行列式)
- (5) 矩阵幂运算： 用 \wedge . 如 A^3 , 表示 $A*A*A$
- (6) 求矩阵呢的秩： rank(a)



Matlab基础：矩阵的生成和基本运算

●矩阵的基本运算

记忆

斜杠压倒的作为分母

除法有两种形式：左除“\”和右除“/”

左除不计算逆直接进行除法运算, 这样可避免奇异矩阵无法求逆带来的麻烦
右除是先计算逆再做乘法

左除:

$Ax=b$, 即 $x = A^{-1}b$, A^{-1} 可以视作1除以A, 矩阵A应当做分母,
因此, 在matlab中, 方程的解为 $A \backslash b$ 或 $x = \text{inv}(A) * b$

$$= \begin{bmatrix} 1 \backslash 2 & 1/2 \\ 2.0000 & 0.5000 \end{bmatrix}$$

右除:

$xA=b$, 即 $x = bA^{-1}$, 同理方程的解: b/A

例: $A = [1, 2, 3, 2; 3, 2, 4, 1; 3, 1, 5, 6; 2, 5, 3, 2]$, $b = [1; 3; 2; 1]$, 求方程组 $Ax=b$ 的解.
由于 $\text{rank}(A) = \text{rank}(B) = 4$ (B为增广矩阵), 所以有唯一解, $x = A \backslash b$, 或 $x = \text{inv}(A) * b$.



优化用到的: Matlab求梯度: 线性函数逼近

◆使用特定点处的梯度线性逼近附近点处的函数值, 并将其与实际值进行比较。函数值的线性逼近方程是

$$f(x) \approx f(x_0) + (\nabla f) x_0 \cdot (x - x_0)$$

◆也就是说, 如果知道函数 $f(x_0)$ 的值和导数 $(\nabla f) x_0$ 在特定点 x_0 的斜率, 则可以根据此信息近似求出该函数在附近点 $f(x) = f(x_0 + \epsilon)$ 的值。

◆计算正弦函数在 -1 和 0.5 之间的一些值。然后计算梯度。



Matlab求梯度： 线性函数逼近

```
y = sin(-1:0.25:0.5); %计算正弦函数在 -1 和 0.5 之间的值。  
yp = gradient(y,0.25); % 计算梯度。  
y_guess = y(end) + yp(end)*(0.5005 - 0.5)  
    %根据  $x = 0.5$  处的函数值和导数来预测  $\sin(0.5005)$  的值  
    % $f(x) \approx f(x_0) + (\nabla f) x_0 \cdot (x - x_0)$   
y_actual = sin(0.5005)
```



Matlab求Hessian矩阵:

```
clc
```

```
syms x y z;
```

```
f=x^2+y^2+z^2+x*y*z;
```

```
% hessian(f,[x,y,z]) %找不到hessian.m
```

```
J1=jacobian(f,[x,y,z])
```

```
J2=jacobian(J1,[x,y,z])
```

$$H(\mathbf{x}) = \nabla^2 f(\mathbf{x})$$



常用最优化方法

比如想从广州去杭州，怎样最快又最经济（目标函数）？

各种导航APP： 比如高德/百度地图

你有很多种方法，可以坐火车，飞机，汽车(很多种解，而且可以对这些解进行组合)，但总是有个组合最让你满意（最优解），最符合你的期望。怎么去求解这个最优解，由此产生的一系列方法。



◆线性回归建模

◆无约束优化梯度分析法

◆无约束优化迭代法

◆线性回归求解

适用人群

- 有志于从事人工智能研究或者工作的在校学生
- 希望在原本的基础上更进一步掌握人工智能数学原理的从业者
- 有转行需求的其他互联网相关职位的从业者或兴趣者

学前基础

1. 学过高等数学、线性代数、概率论三门课程中的至少两门；
2. 具有**简单的 Matlab 基础知识**，简单的 Python 基础知识。



最优化方法--数学基础--线性回归

在统计学中，线性回归是利用称为线性回归方程的最小二乘函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合。一个带有一个自变量的线性回归方程代表一条直线。我们需要对线性回归结果进行统计分析。

实例：学习时间和分数有什么样的关系？

1.提出问题

已知一部分学生的学习时间和分数，预测另一部分学生的分数。其中，特征为学习时间，标签为分数。

2.理解数据

将数据集的前五项列出



线性回归—示例

提出问题

已知一部分学生的学习时间和分数，预测另一部分学生的分数。其中，特征为学习时间，标签为分数

学习时间:

[0.50,0.75,1.00,1.25,1.50,1.75,1.75,2.00,2.25,
2.50,2.75,3.00,3.25,3.50,4.00,4.25,4.50,4.75,5.00,5.50]

分数:

[10, 22, 13, 43, 20, 22, 33, 50, 62, 48, 55, 75, 62, 73, 81, 76,
64, 82, 90, 93]



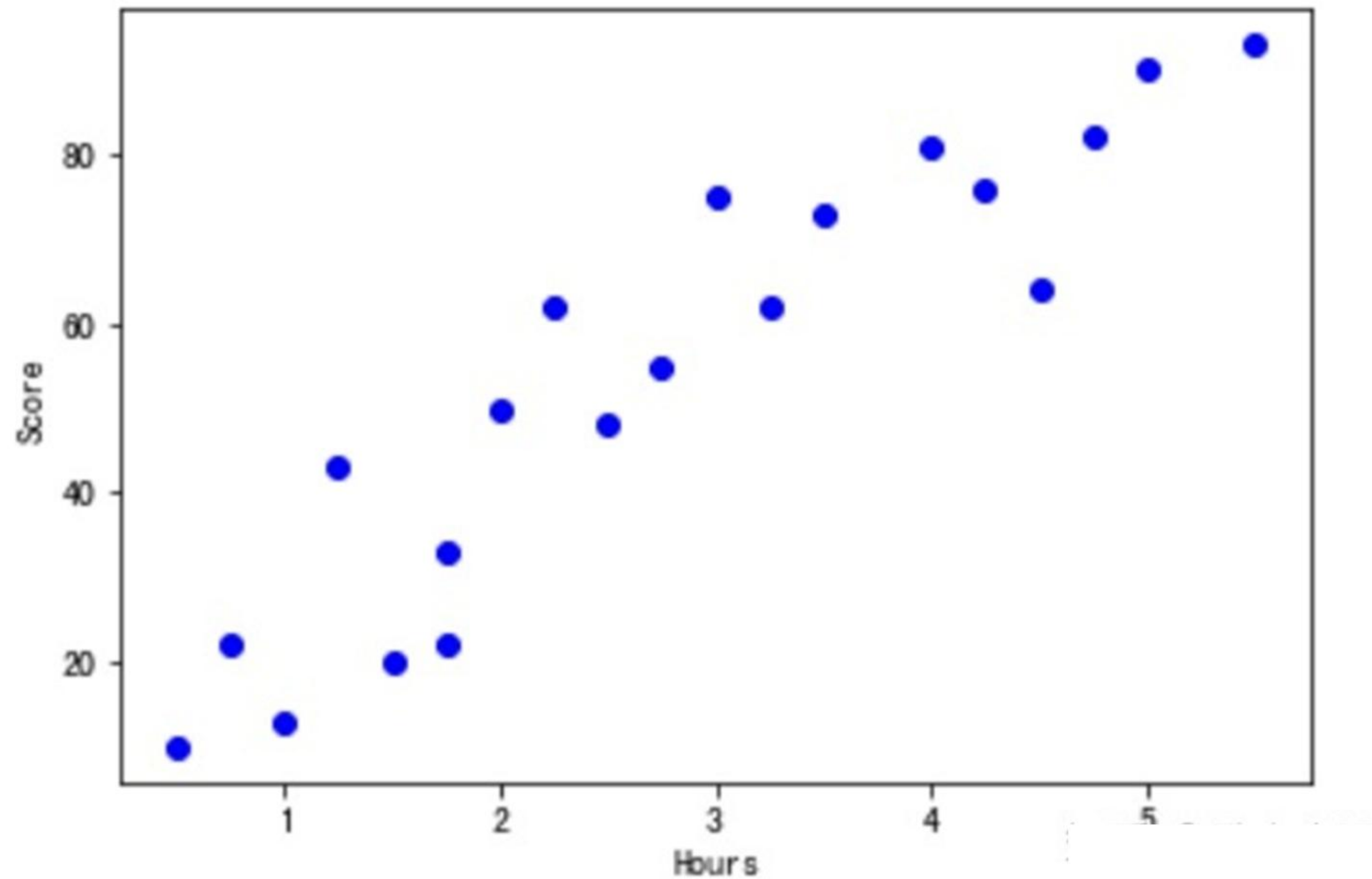
前五项

Out[9]:

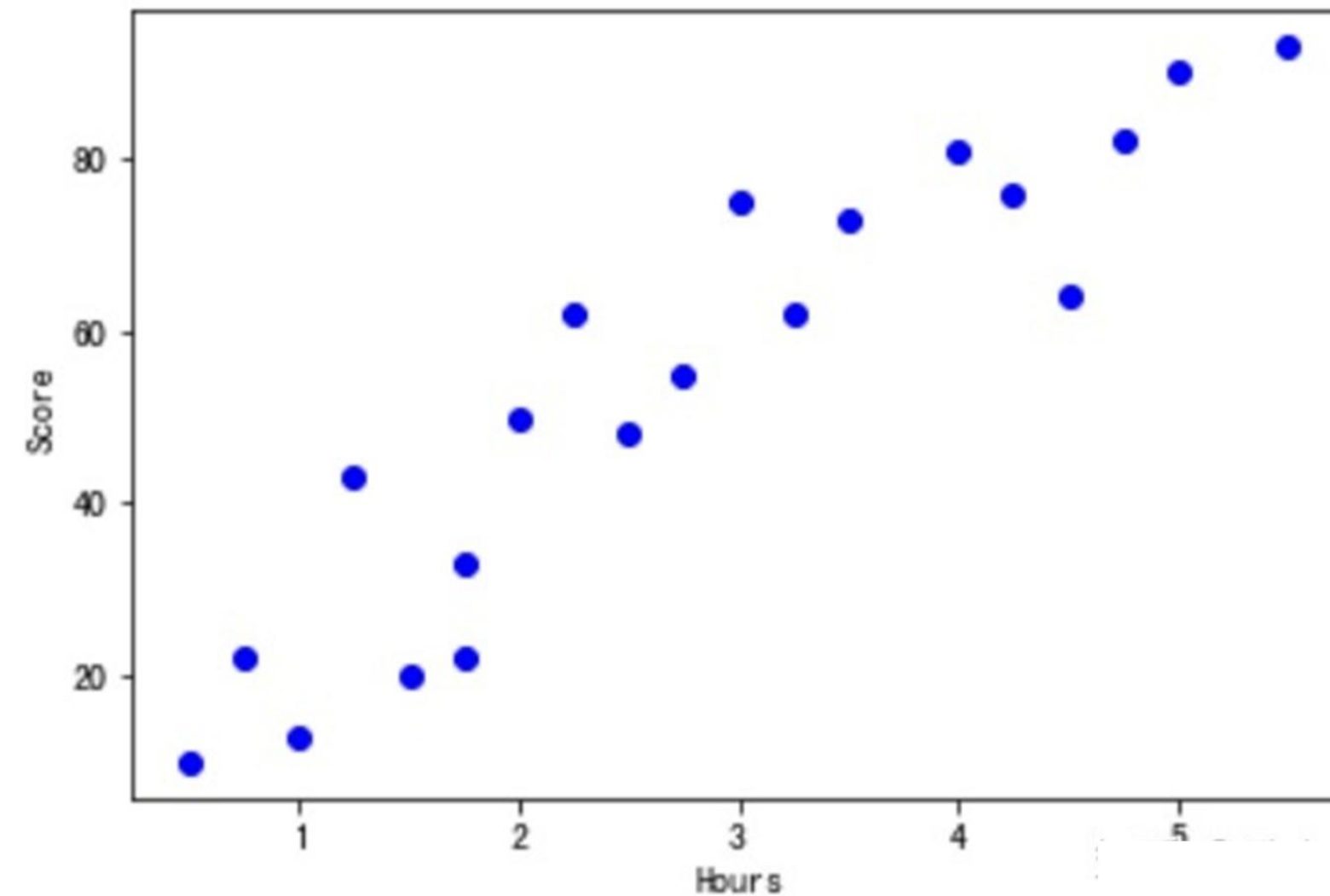
	学习时间	分数
0	0.50	10
1	0.75	22
2	1.00	13
3	1.25	43
4	1.50	20



散点图

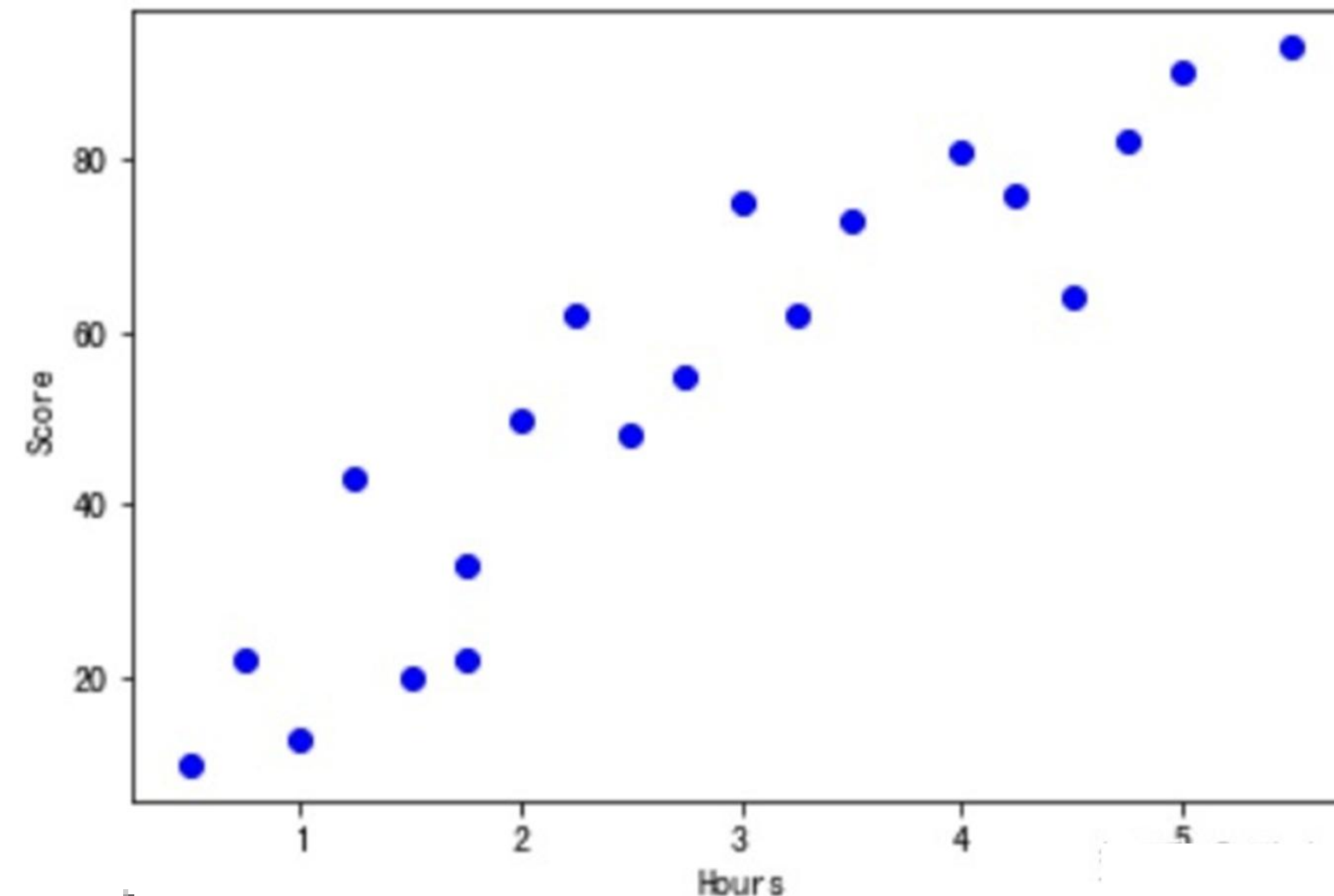


分析:



通过散点图，可知学习与分数成正相关。相关性分为正线性相关、负线性相关和非线性相关。衡量两个变量的相关性有两种方法：比例系数与相关系数 r 。比例系数为正，即正相关。

分析:



相关系数 $r = \frac{cov(x, y)}{\sigma_x \sigma_y}$, 其中 $cov(x, y) = E[(x - E(x))(y - E(y))]$

为协方差, σ_x 、 σ_y 为标准差。相关系数在[0,0.3]为弱相关, 在[0.3,0.6]为

中等程度相关, 在[0.6,1]为强相关。



分析:

	学习时间	分数
学习时间	1.000000	0.923985
分数	0.923985	1.000000

相关系数约为0.924，说明学习时间和分数是强相关。



构建模型

(1) 提取特征和标签

特征：学习时间

标签：分数

(2) 建立训练数据和测试数据

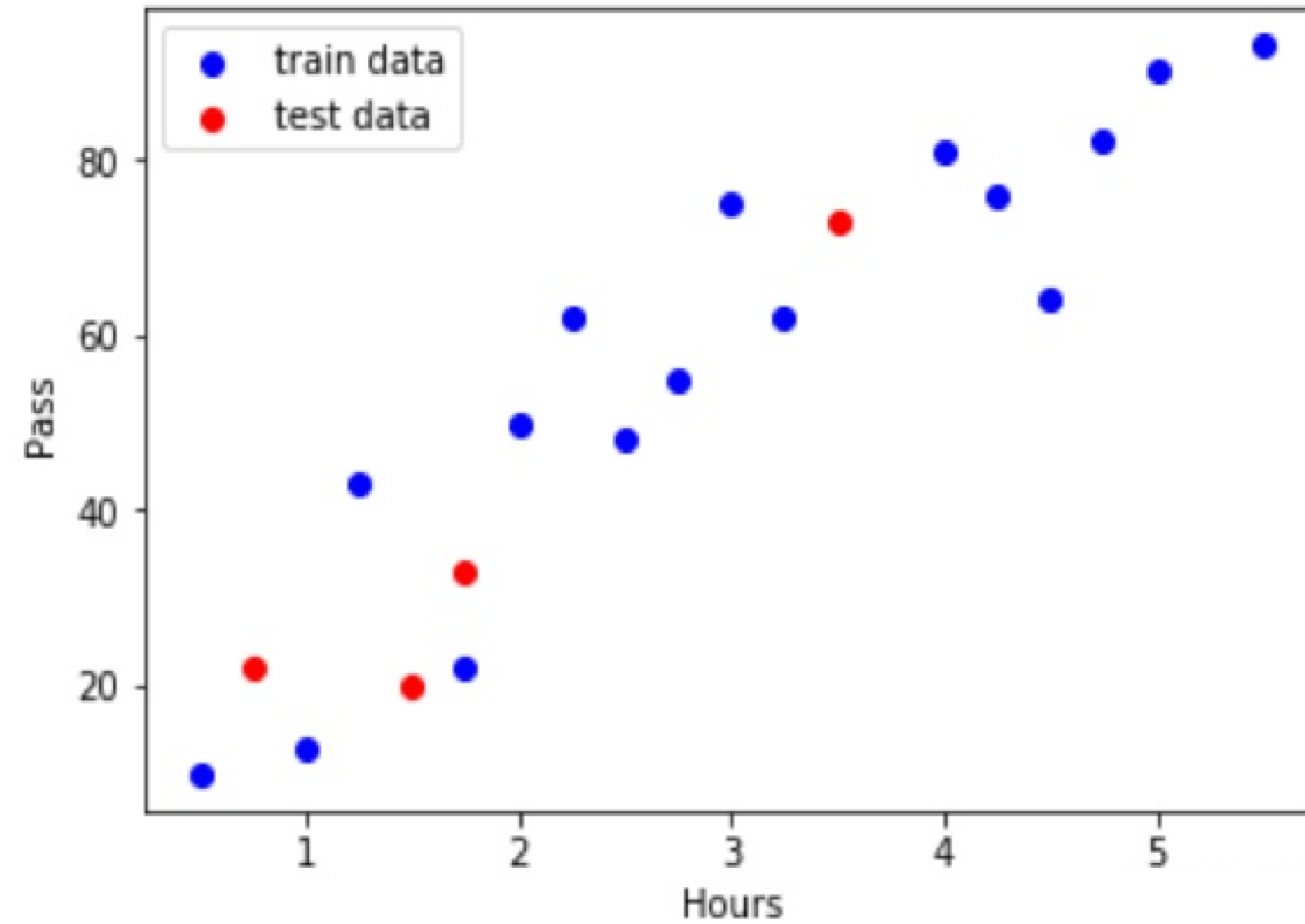
功能是从样本中随机的按比例选取训练数据（train）和测试数据（test）

原始数据特征：（20，），训练数据特征：（16，），测试数据特征：
（4，）

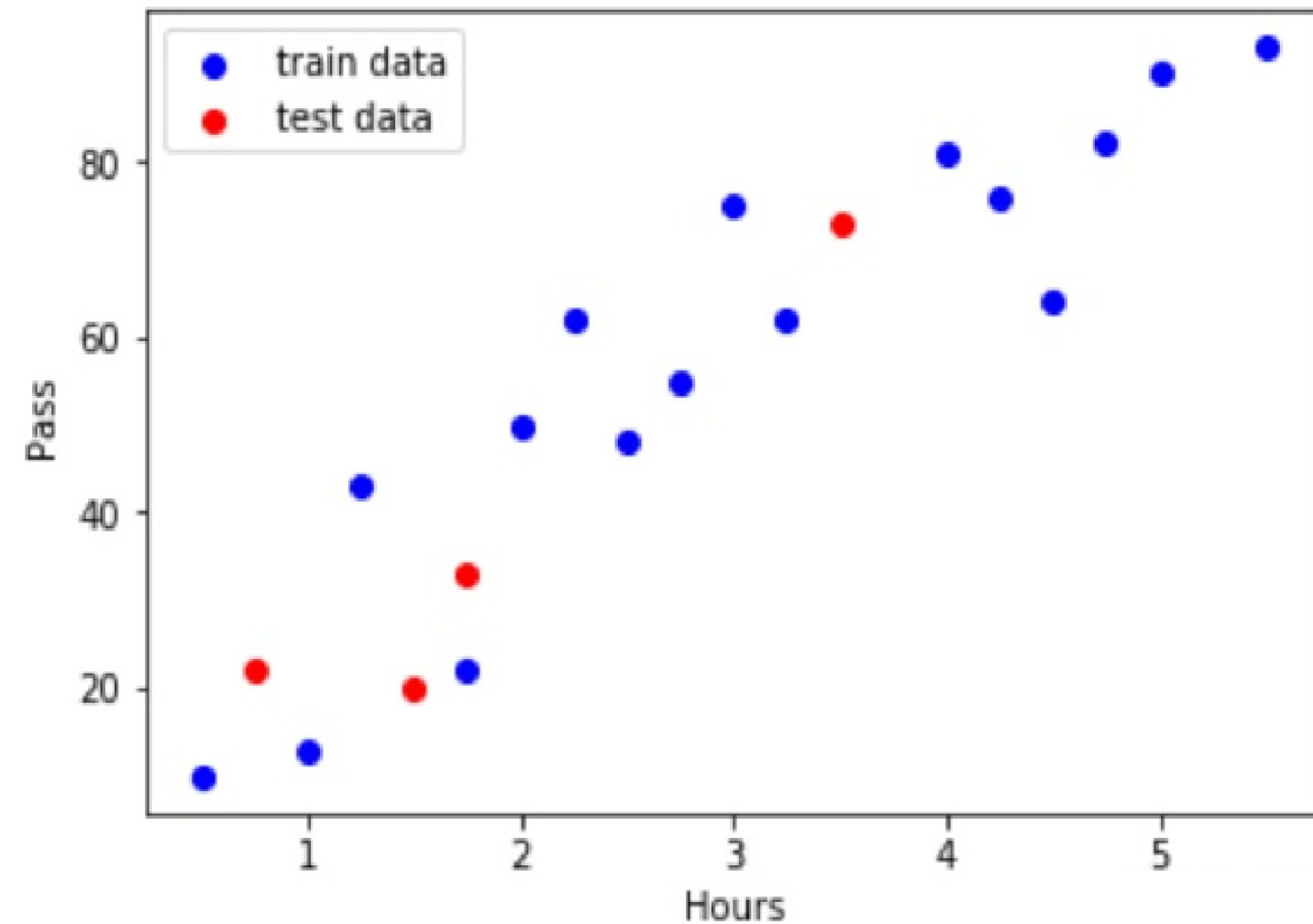
原始数据标签：（20，）训练数据标签：（16，）测试数据标签：（4，）



散点图



训练模型（使用训练数据）——绘制最佳拟合线



$$y\text{误差平方和} = \Sigma(y\text{实际值} - y\text{预测值})^2$$

使误差平方和最小的曲线。得到最佳拟合曲线：

$$y = 11.1707480711 + 15.48742033x$$



模型评估（使用测试数据）

决定系数R平方！R平方越大，表明回归模型越精确。

$$y\text{误差平方和} = \Sigma(y\text{实际值} - y\text{预测值})^2$$

$$y\text{的总波动} = \Sigma(y\text{实际值} - y\text{平均值})^2$$

$$\text{决定系数} R^2 = 1 - \frac{\text{误差平方和}}{\text{总波动}}$$

得到 $R^2 = 0.83901474742039228$ ，拟合效果较好



线性回归

	1 name	2 sex	3 age	4 wgt	5 smoke	6 sys	7 dia	8 trial1	9 trial
1 YPL-320	'SMITH'	'm'	38	176	1	124	93	18	
2 GLI-532	'JOHNSON'	'm'	43	163	0	109	77	11	
3 PNI-258	'WILLIAMS'	'f'	38	131	0	125	83	-99	
4 MIJ-579	'JONES'	'f'	40	133	0	117	75	6	
5 XLK-030	'BROWN'	'f'	49	119	0	122	80	14	
6 TFP-518	'DAVIS'	'f'	46	142	0	121	70	19	
7 LPD-746	'MILLER'	'f'	33	142	1	130	88	0	
8 ATA-945	'WILSON'	'm'	40	180	0	115	82	-99	
9 VNL-702	'MOORE'	'm'	28	183	0	115	78	2	
10 LQW-768	'TAYLOR'	'f'	31	132	0	118	86	11	
11 QFY-472	'ANDERS...	'f'	45	128	0	114	77	8	
12 UJG-627	'THOMAS'	'f'	42	137	0	115	68	4	
13 XUE-826	'JACKSON'	'm'	25	174	0	127	74	-99	
14 TRW-072	'WHITE'	'm'	39	202	1	130	95	8	

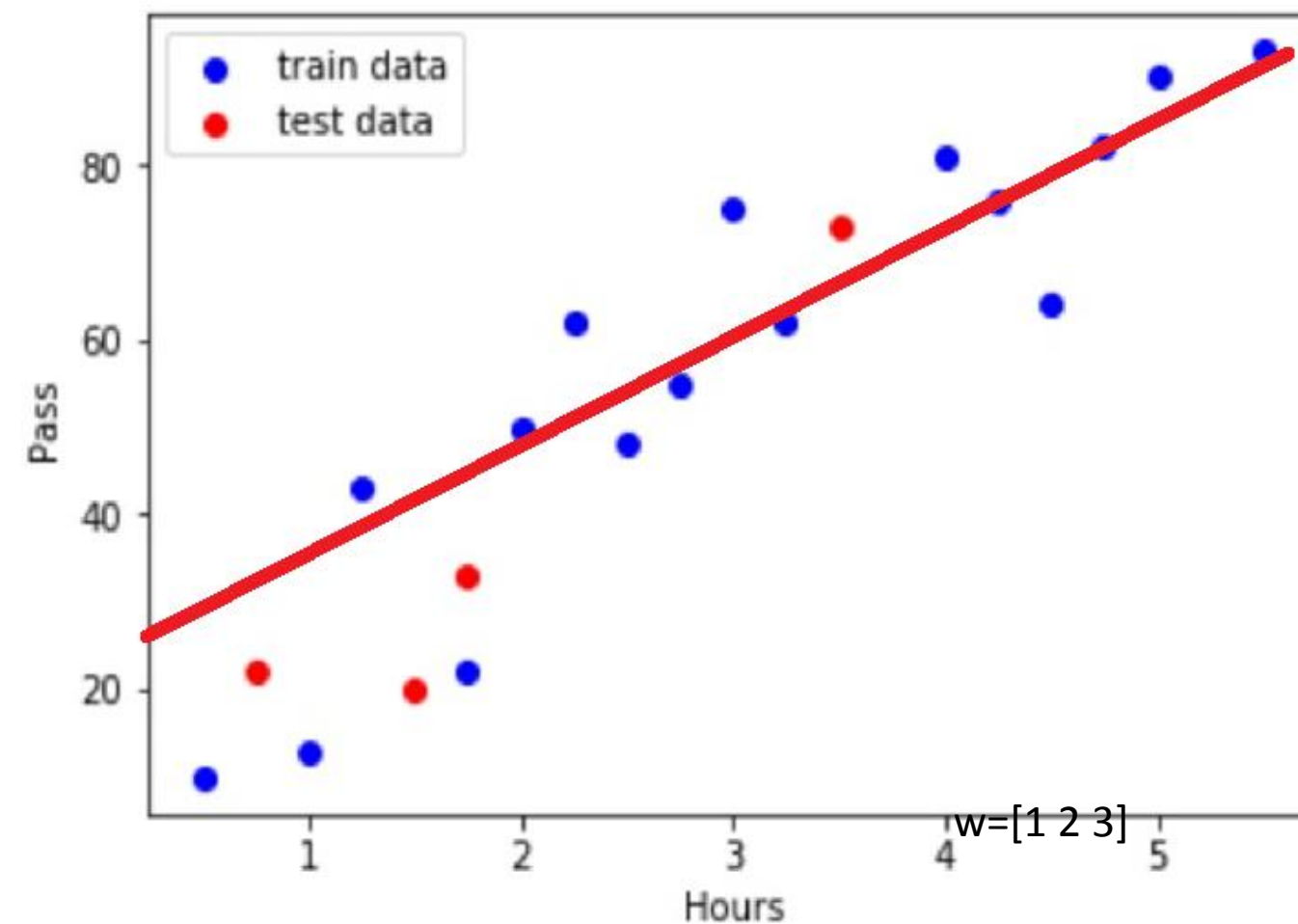
一个维度训练样本

$$\{(x^{(i)}, y^{(i)})\}$$

训练样本集 $\{(x^{(i)}, y^{(i)}) ; i = 1, \dots, N\}$

多个维度: $\{(x_1^{(i)}, x_2^{(i)}, y^{(i)})\} \longrightarrow \{(\mathbf{x}^{(i)}, y^{(i)})\}, \mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$

线性回归



试图学习: $f(x) = wx + b$ 使得 $f(x^{(i)}) \approx y^{(i)}$

试图学习: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$ $f(X^{(i)}) \approx y^{(i)}$

如何求得 w 和 b ?



无约束优化

自变量为标量的函数 $f: \mathbb{R} \rightarrow \mathbb{R}$

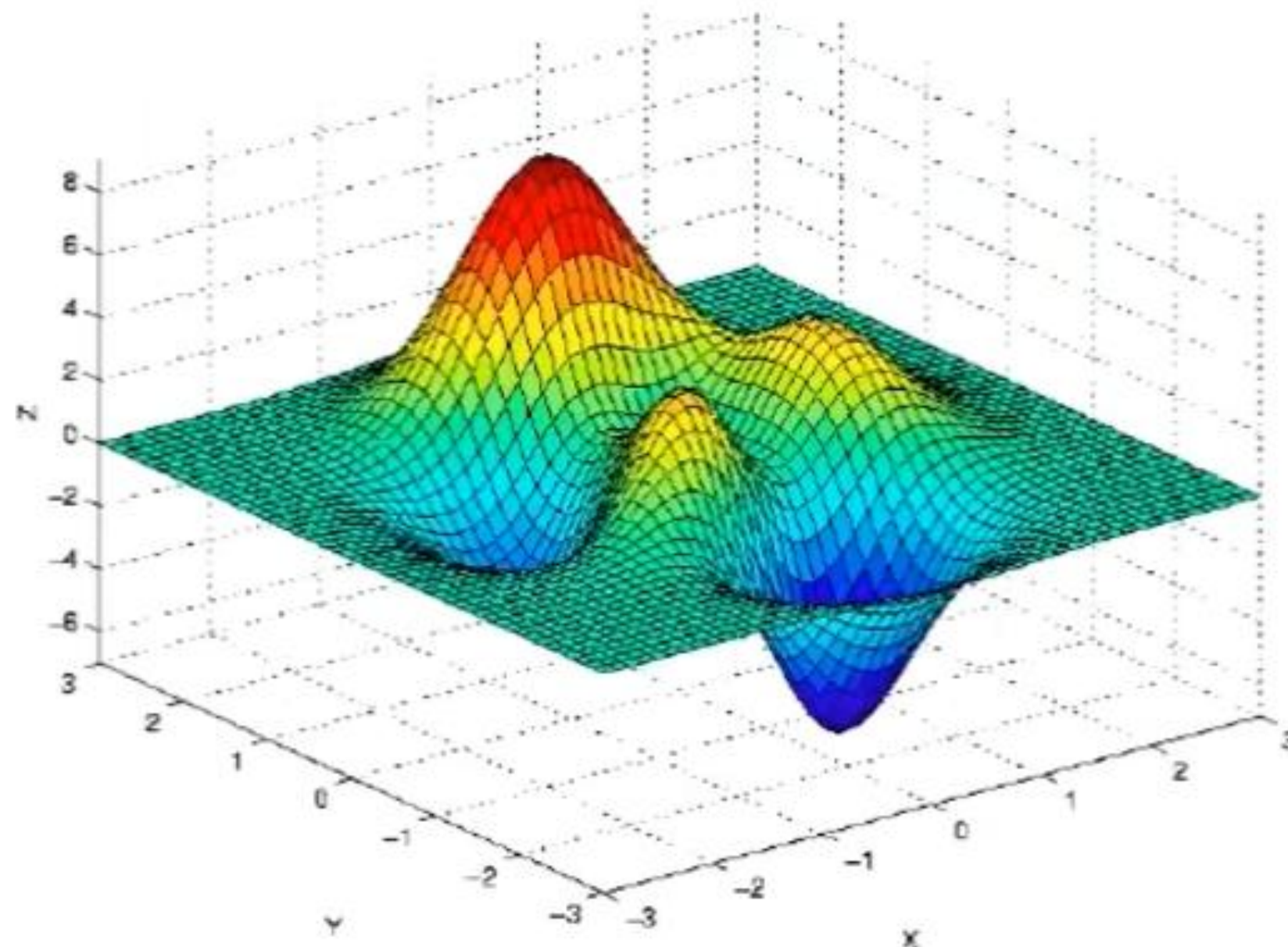
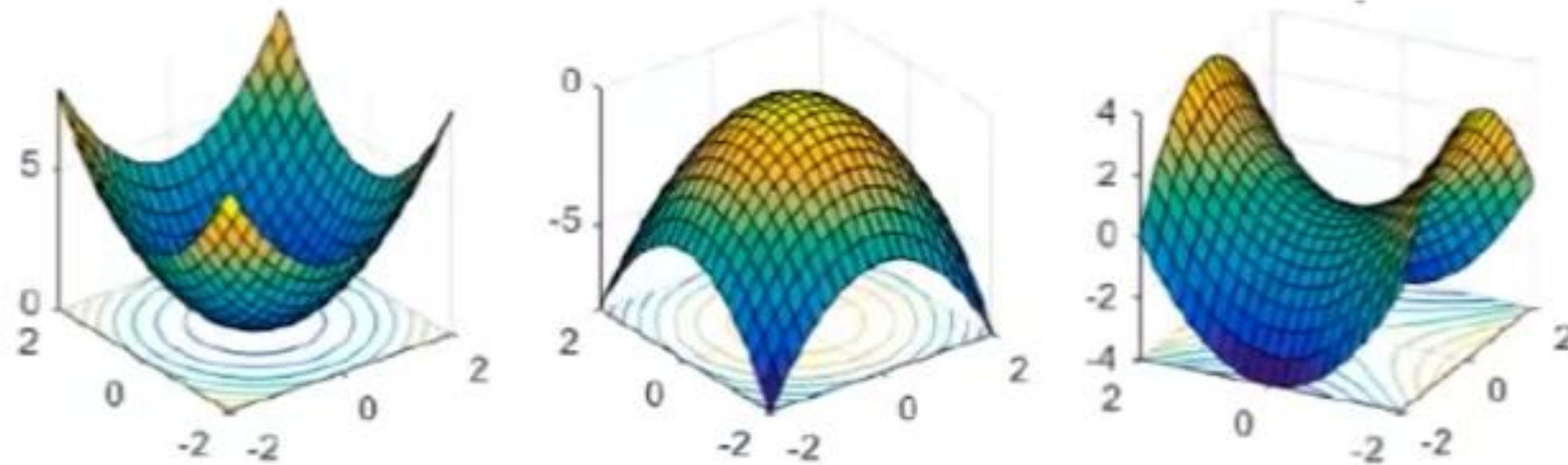
$$\min f(x) \quad x \in \mathbb{R}$$

自变量为向量的函数 $f:$

$$\min f(X) \quad X \in \mathbb{R}^n$$



优化问题的极值点：全局极值、局部极值，鞍点



求函数极值需要的数学: 梯度和 Hessian 矩阵

一阶导数和梯度 (gradient vector)

$$f'(x) \quad \text{多元情况下的导数} \quad g(x) = \nabla f(x) = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

二阶导数和 Hessian 矩阵

$$H(x) = \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \cdots \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & & & \\ & & \ddots & & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & & \frac{\partial^2 f(x)}{\partial x_n^2} & \end{bmatrix} = \nabla(\nabla f(x))^T$$

二次型

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

给定矩阵 $\mathbf{A} \in \mathbb{R}^{n \times n}$ ，函数

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{i=1}^n x_i (\mathbf{A} \mathbf{x})_i = \sum_{i=1}^n x_i \left(\sum_{j=1}^n a_{ij} x_j \right) = \sum_{i=1}^n \sum_{j=1}^n x_i x_j a_{ij}$$

称为二次型

- 给定对称矩阵 $\mathbf{A} \in \mathbb{R}^{n \times n}$ 如果对于所有 $\mathbf{x} \in \mathbb{R}^n$ 有 $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$ ，则为半正定矩阵 (positive semidefnite)，此时特征值 $\lambda(\mathbf{A}) \geq 0$ （特征值全大于0则正定）。
- 如果对于所有 $\mathbf{x} \in \mathbb{R}^n$ ，当 $\mathbf{x} \neq \mathbf{0}$ ，有 $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$ ，则为正定矩阵 (positive defnite)。
- 负定矩阵, 不定矩阵 (indefnite)



◆对于标量 $(\mathbf{a}\mathbf{x})' = \mathbf{a}$ ，由类比可得矢量情况：
向量 \mathbf{a} 和 \mathbf{x} 无关，则 $\nabla(\mathbf{a}^T\mathbf{x}) = \mathbf{a}$, $\nabla^2(\mathbf{a}^T\mathbf{x}) = \mathbf{0}$

◆对于标量 $(\mathbf{x}\mathbf{A}\mathbf{x})' = 2\mathbf{A}\mathbf{x}$ ，由类比可得矢量情况：
对称矩阵矩阵 \mathbf{A} 与 \mathbf{x} 无关，则

$$\nabla(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}, \quad \nabla^2(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}$$

$$\|\mathbf{x}\|_2^2 = x_1^2 + x_2^2 + x_3^2 = \mathbf{x}^T\mathbf{x}$$

$$\begin{aligned} f(x) &= (Ax - b)^T (Ax - b) \\ &= (x^T A^T - b^T) (Ax - b) \\ &= x^T A^T A x - \underbrace{b^T A x} - \underbrace{x^T A^T b} + \underbrace{b^T b} \end{aligned}$$

◆ 二范数 $\|x\|_2^2 = x_1^2 + x_2^2 + x_3^2 = x^T x$

◆ 最小二乘

$$\begin{aligned} f(\mathbf{x}) &= \|\mathbf{Ax} - \mathbf{b}\|_2^2 \\ &= \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} - 2\mathbf{b}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{b} \\ \nabla f(\mathbf{x}) &= 2\mathbf{A}^T \mathbf{A} \mathbf{x} - 2\mathbf{A}^T \mathbf{b} \end{aligned}$$

输入为标量的泰勒级数展开:

$$f(x_k + \delta) \approx f(x_k) + f'(x_k) \delta + \frac{1}{2} f''(x_k) \delta^2 + \cdots + \frac{1}{k!} f^k(x_k) \delta^k + \cdots$$

输入为向量的泰勒级数展开:

$$f(\mathbf{x}_k + \delta) \approx f(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k) \delta + \frac{1}{2} \delta^T \mathbf{H}(\mathbf{x}_k) \delta$$

$$\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

二次型



标量情况

输入为标量的泰勒级数展开:

$$f(x_k + \delta) \approx f(x_k) + f'(x_k) \delta + \frac{1}{2} f''(x_k) \delta^2$$

- 严格局部极小点指: $f(x_k + \delta) > f(x_k)$
- 称满足 $f'(x_k) = 0$ 的点为平稳点 (候选点).
- 函数在 x_k 有严格局部极小值条件为 $f'(x_k) = 0$ 且 $f''(x_k) > 0$



泰勒级数和极值

向量情况

输入为向量的泰勒级数展开:

$$f(\mathbf{x}_k + \boldsymbol{\delta}) \approx f(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k) \boldsymbol{\delta} + \boldsymbol{\delta}^T \mathbf{H}(\mathbf{x}_k) \boldsymbol{\delta}$$

-称满足 $\mathbf{g}(\mathbf{x}_k) = 0$ 的点为平稳点 (候选点),

如果有 $\mathbf{H}(\mathbf{x}_k) > 0$, \mathbf{x}_k 为一严格局部极小点 (反之, 严格局部最大点);

如果 $\mathbf{H}(\mathbf{x}_k)$ 不定矩阵, 是一个鞍点 (saddle point)



梯度等于0，求解的局限性

求 $f(x) = x^4 + \sin(x^2) - \ln(x)e^x + 7$ 的导数

$$\begin{aligned} f'(x) &= 4x^{(4-1)} + \frac{d(x^2)}{dx} \cos(x^2) - \frac{d(\ln x)}{dx} e^x - \ln(x) \frac{d(e^x)}{dx} + 0 \\ &= 4x^3 + 2x \cos(x^2) - \frac{1}{x} e^x - \ln(x) e^x \end{aligned}$$

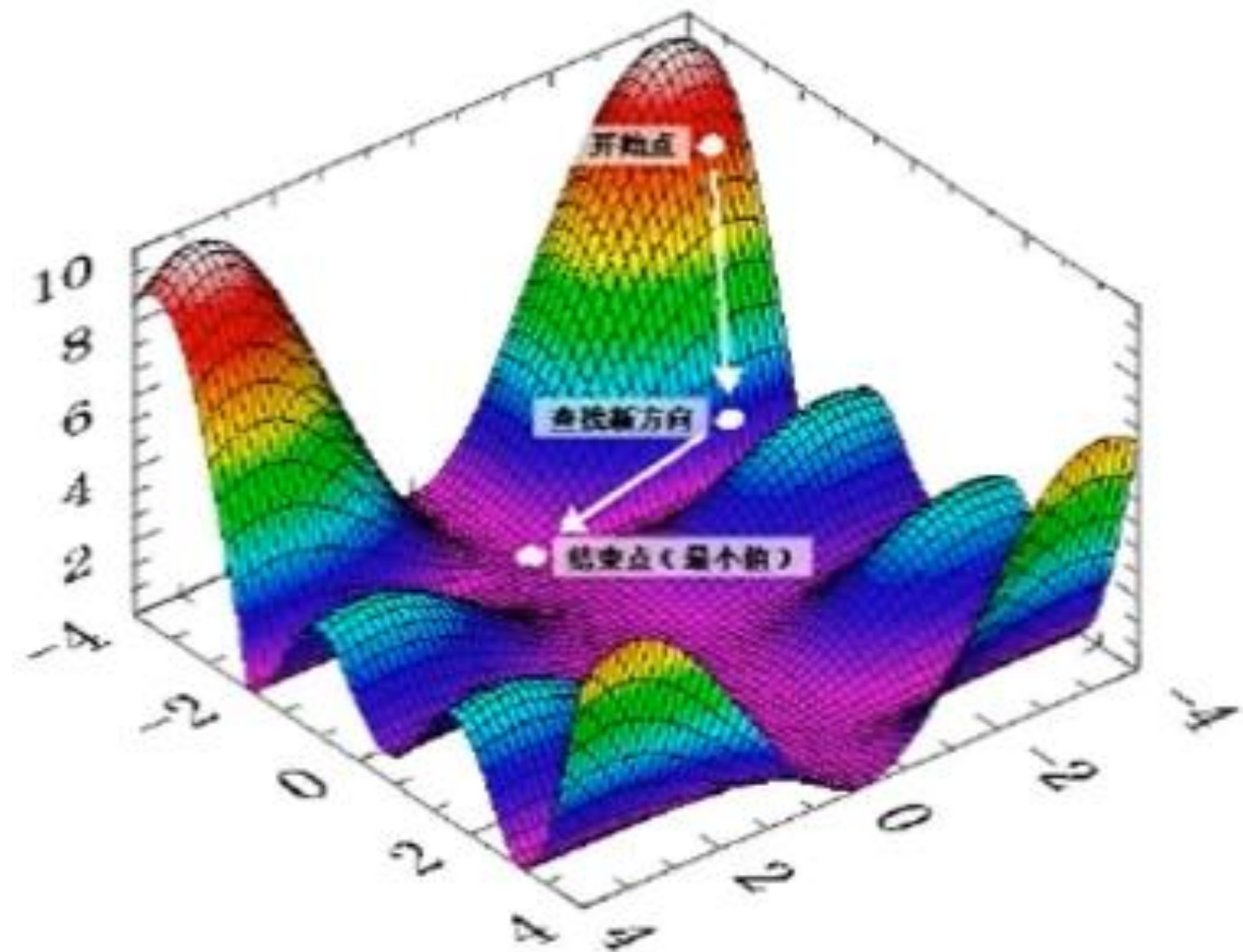
如果求 $f'(x) = 0$ 很难求得结果，多元情况则更加难



无约束优化迭代法：工程以及AI中用的较广

迭代法的基本结构 (最小化 $f(\mathbf{x})$)

- 1 选择一个初始点，设置一个 convergence tolerance ϵ ，计数 $k = 0$
 - 2 决定**搜索方向** \mathbf{d}_k ，使得函数下降. (核心)
 - 3 决定步长 α_k 使得 $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$ 对于 $\alpha_k \geq 0$ 最小化，构建 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
 - 4 如果 $\|\mathbf{d}\|_2 < \epsilon$ ，则停止输出解 \mathbf{x}_{k+1} ；否则继续重复迭代
- 一般求出来的都是局部最优解



梯度下降法：在机器学习和AI中大量使用！

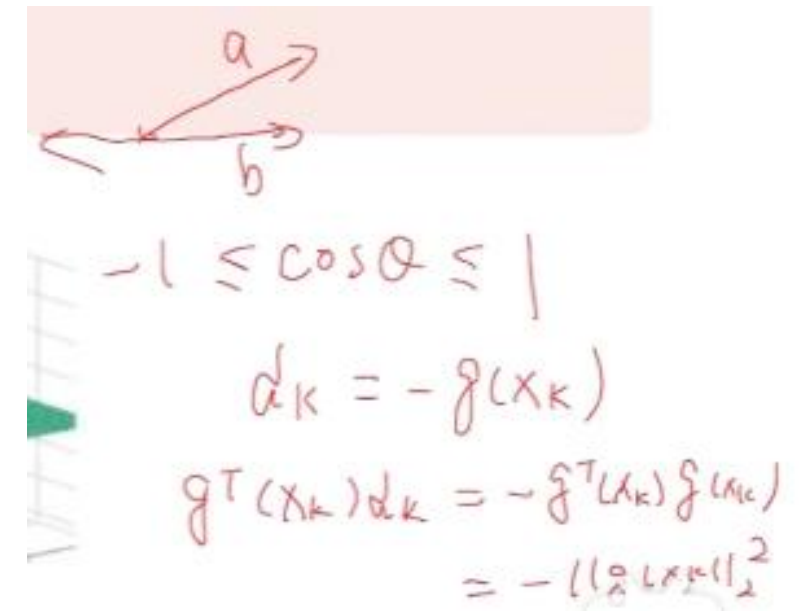
梯度下降法---只保留一阶的情况下是最优的

$\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k)$, 思考为什么这么取？

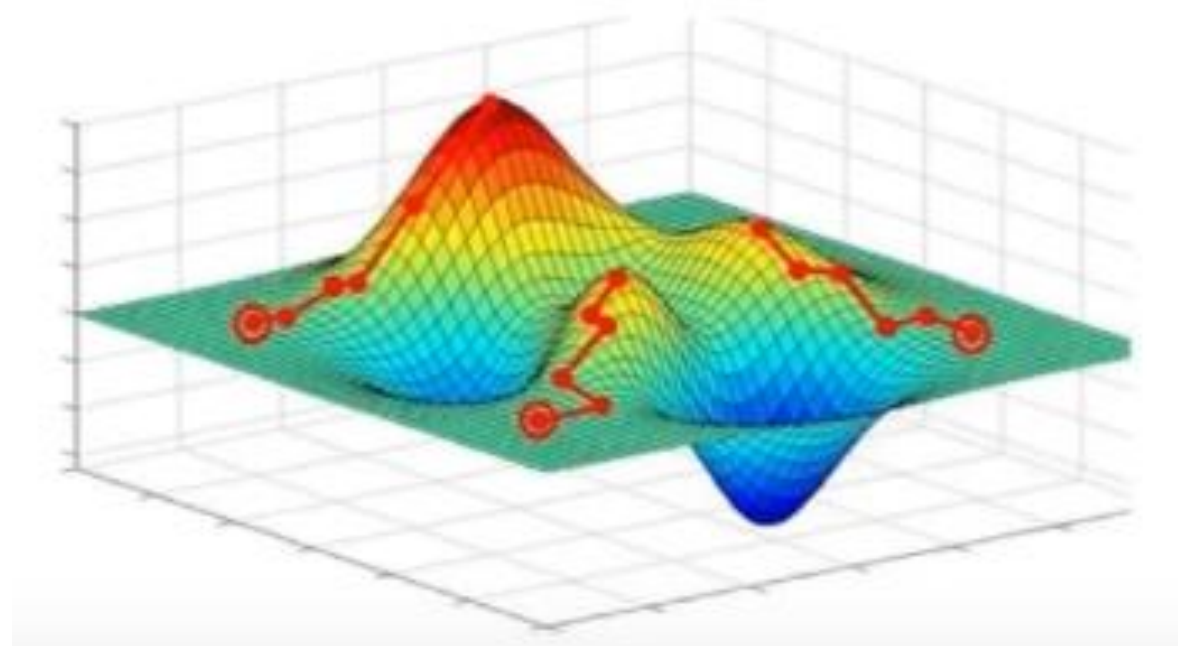
$$f(\mathbf{x}_k + \mathbf{d}_k) \approx f(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k) \mathbf{d}_k$$

- 需要 $f(\mathbf{x}_k + \mathbf{d}_k) \downarrow$, 则 $f(\mathbf{x}_k)$ 加个负数
- 回忆两个向量的内积:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b} = ||\mathbf{a}||_2 \cdot ||\mathbf{b}||_2 \cos \vartheta$$



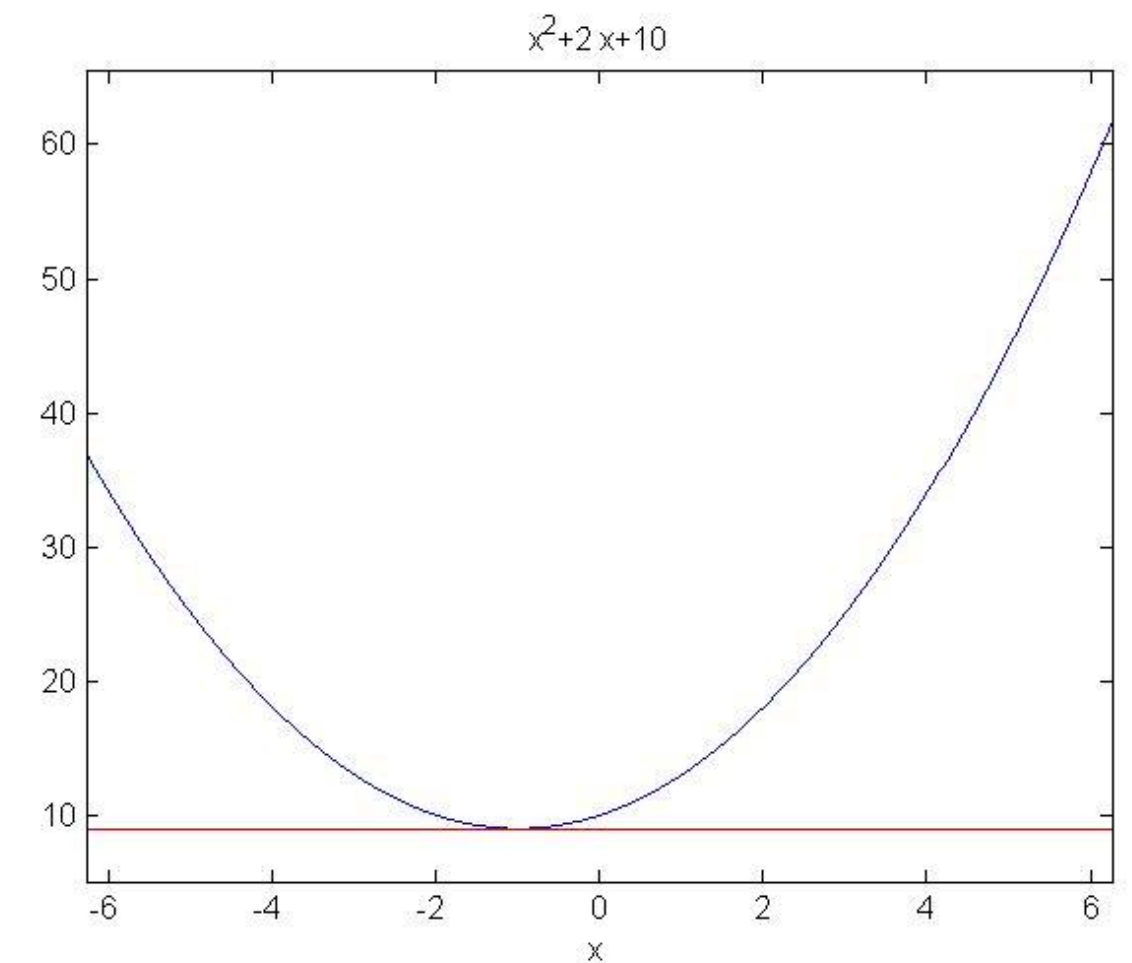
$-1 \leq \cos \vartheta \leq 1$
 $\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k)$
 $\mathbf{g}^T(\mathbf{x}_k) \mathbf{d}_k = -\mathbf{g}^T(\mathbf{x}_k) \mathbf{g}(\mathbf{x}_k) = -||\mathbf{g}(\mathbf{x}_k)||_2^2$



Blog: 【Machine Learning 六】梯度下降法（基于Matlab 求函数最小值）

```
%代码有点问题
%但表达的意思是正确的
clc
syms x y real
y(x) = x^2+2*x+10;
delta(x) = -diff(y(x));
step = 0.1;
first_x = 10;
x_current = first_x;
x_next = first_x;
show_tmp = 0;
show = [];
counter = 0;
delta_last = 0;
```

```
while (true)
    delta_x = double(delta(x_current))*step;
    if(abs(delta_x) < 0.0001)
        break
    end
    x_next = x_current + delta_x;
    x_current = x_next;
    counter = counter + 1;
    show_tmp = x_next;
    show = [show, show_tmp];
    if(counter > 200)
        break;
    end
end
x_next %带入y(x) = x^2+2*x+10;
```



牛顿法：函数近似时，由于二阶项占的比重

- 方向选取 $\mathbf{d}_k = -\mathbf{H}^{-1}(\mathbf{x}_k) \mathbf{g}(\mathbf{x}_k)$

- 方向选取依据

$$f(\mathbf{x}_k + \mathbf{d}_k) = f(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k) \mathbf{d}_k + \frac{1}{2} \mathbf{d}_k^T \mathbf{H}(\mathbf{x}_k) \mathbf{d}_k$$

$$\text{令 } \frac{\partial f(\mathbf{x}_k + \mathbf{d}_k)}{\partial \mathbf{d}_k} = 0 \Rightarrow \mathbf{g}(\mathbf{x}_k) + \mathbf{H}(\mathbf{x}_k) \mathbf{d}_k = 0$$

- 若 Hessian 矩阵正定，则有 $\mathbf{d}_k = -\mathbf{H}^{-1}(\mathbf{x}_k) \mathbf{g}(\mathbf{x}_k)$

- 强制要求 Hessian 矩阵正定

◆ 对于标量 $(\mathbf{a}\mathbf{x})' = \mathbf{a}$ ，由类比可得矢量情况：
向量 \mathbf{a} 和 \mathbf{x} 无关，则 $\nabla(\mathbf{a}^T \mathbf{x}) = \mathbf{a}$, $\nabla^2(\mathbf{a}^T \mathbf{x}) = \mathbf{0}$

◆ 对于标量 $(\mathbf{x}\mathbf{A}\mathbf{x})' = 2\mathbf{A}\mathbf{x}$ ，由类比可得矢量情况：
对称矩阵 \mathbf{A} 与 \mathbf{x} 无关，则
 $\nabla(\mathbf{x}^T \mathbf{A} \mathbf{x}) = 2\mathbf{A}\mathbf{x}$, $\nabla^2(\mathbf{x}^T \mathbf{A} \mathbf{x}) = 2\mathbf{A}$



拟牛顿法（ DFP 、 BFGS ）——同学们自学！

- ◆ 牛顿法的基本思想是在迭代点 \mathbf{x}_k 对目标函数 $f(\mathbf{x})$ 进行二次函数近似，然后把二次模型的极小点作为新的迭代点，并不断重复这一过程，直至求得满足精度的近似极小点。牛顿法有个不足之处：在输入维数很大时Hession矩阵的逆不好求。
- ◆ 拟牛顿法可以在保证收敛速度的同时，不用满足Hesse矩阵处处正定的条件并且可以避免每次都进行Hesse计算，通过构造可以近似Hesse矩阵（或Hesse矩阵的逆）的正定对称阵。

尝试用一正定矩阵逼近 $\mathbf{H}^{-1}(\mathbf{x}_k)$



