

# 智能信息处理

主讲：张磊

办公室：主教1030

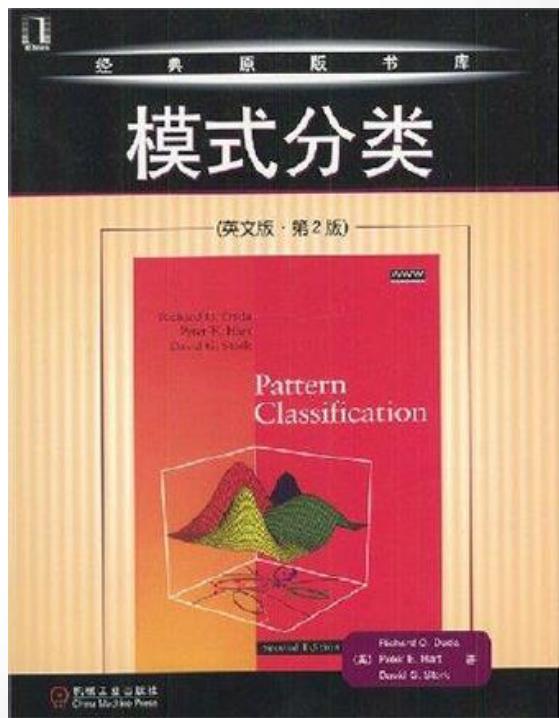
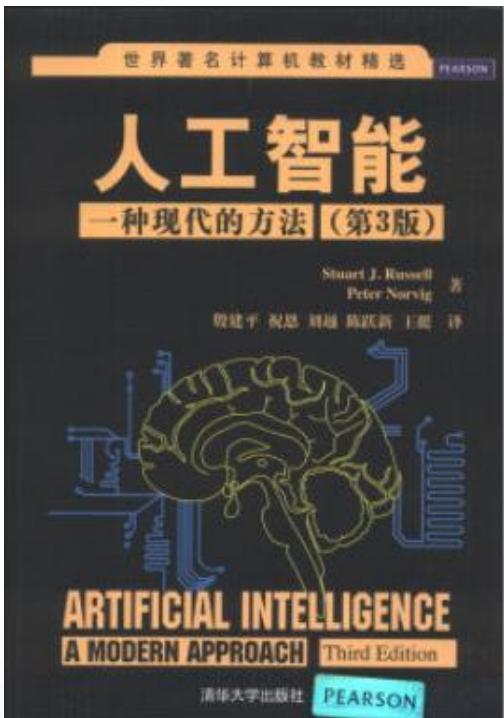
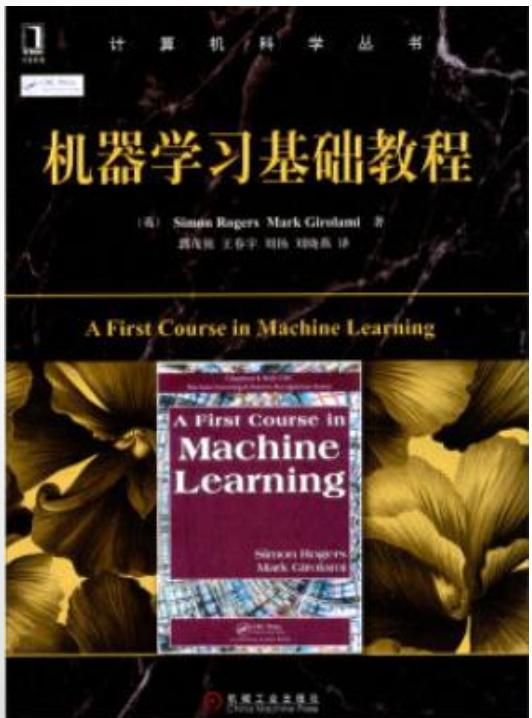
E-mail: [leizhang@cqu.edu.cn](mailto:leizhang@cqu.edu.cn)

Website: <http://www.leizhang.tk>





## 教材与参考书目





## 参考书目

Simon Rogers, Mark Girolami编，《机器学习基础教程》，机械工业出版社

Stuart J. Russell, Peter Norvig编《人工智能—一种现代的方法》第3版，清华大学出版社，

Richard O. Duda, Peter E. Hart, David G. Stork编《模式分类》第2版，机械工业出版社，中信出版社

Tom M. Mitchell编，《机器学习》，机械工业出版社



## 课时安排

学时： 24学时（12次）

考试： 理论笔试

学分： 1.5

成绩（百分制）： 70%考试+30%出勤



# 第一章：绪论

人工智能(Artificial Intelligence, AI) 1956:

人工智能是最新兴的科学与工程领域。智能信息处理为实现人工智能的必要手段，其目标和任务是试图理解和创造智能实体。

人工智能的大量粉丝：潜在的爱因斯坦们和爱迪生们，被大多数领域的科学家们评为“最想参与的研究领域”



# 第一章：绪论

## 什么是人工智能(Artificial Intelligence, AI)：

- ◆ 电影和小说中描绘的人工智能（主宰的可怕未来）塑造了大众对人工智能的想象，但这些都是虚构的。在现实生活中，人工智能基本上是在改善人类健康、安全和提升生产力等好的方面。
- ◆ 多数研究型大学和科技公司（苹果、谷歌、facebook、IBM、微软、百度）也投入巨资，并将人工智能视为未来发展的关键。
- ◆ 好莱坞也将人工智能技术搬上荧幕（反乌托邦人工智能幻想故事）

## 人工智能定义：

一直以来，人工智能缺乏一个精确的、被普遍接受的定义。目前，一个有用的定义：人工智能是致力于让机器变得智能的活动，而智能就是使实体在其环境中远见地、适当地实现功能性的能力。



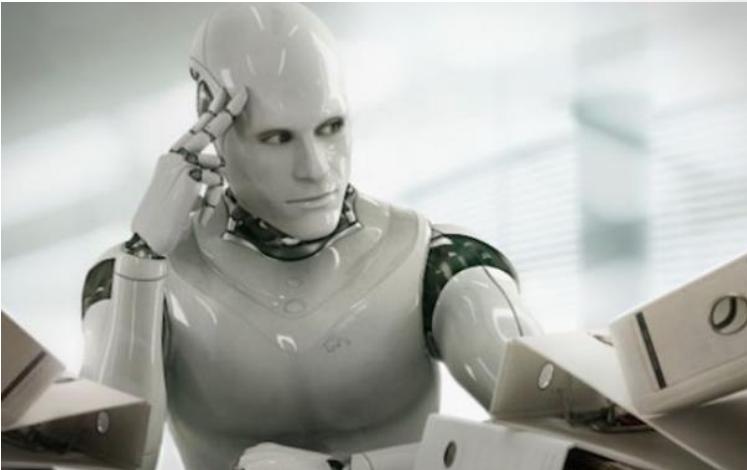
# 第一章：绪论

## 什么是人工智能(Artificial Intelligence, AI)?

### ◆ 像人一样思考

机器要有“脑”，而且会自己思考、决策、求解、学习等。初级的人工智能，需要人类去“教”会机器如何思考，也就是利用计算机模型来研究机器的智能（智力），使机器的自动感知、推理、行为，从计算上成为可能（智能计算）。

利用计算机完成人更加擅长的事情，称为计算智能。



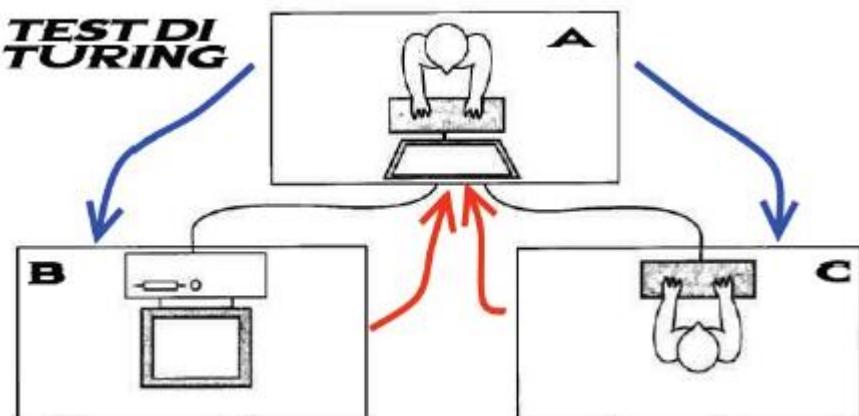


## 什么是人工智能(Artificial Intelligence, AI)?

- ◆ 像人一样行动（行为）

图灵测试：由Alan Turing 1950提出，为人工智能提供了令人满意的可操作的定义。如果一位人类询问者提出一些问题后，无法判断对方的回答是来自人还是计算机，那么这台计算机通过测试（成功“骗”过人）。

60年以后，该测试仍然适合，因此值得称赞





# 第一章：绪论

## 什么是人工智能(Artificial Intelligence, AI)?

◆ 像人一样行动（行为）

为了能“骗”过人类，还需要通过大量的计算机模型和编程工作。计算机需要具备以下能力：

**知识表示与存储**：用于计算机去“记忆”

**机器学习**：对新知识的认知和学习

**自动推理**：思考并回答问题和推出新结论

**自然语言处理**：“听、说”，用于外语交流

**计算机视觉**：“看”世界（感知）

**机器嗅觉、触觉**：“嗅”和“摸”（感知）

**机器人学**：操纵和移动对象



→这些领域涉及人工智能(AI)的大部分内容



# 第一章：绪论

## 什么是人工智能(Artificial Intelligence, AI)?

- ◆ 像人一样行动（行为）

综上所述，要真正通过图灵测试，计算机必须具备理解语言、学习、记忆、推理、决策等能力。从而产生许多学科，如机器感知（计算机视觉、自然语言处理），机器学习（模式识别、增强学习），记忆（知识表示），决策（规划、数据挖掘），这些都属于人工智能的研究范畴。



## 什么是人工智能(Artificial Intelligence, AI)?

大家认为飞机/飞行器属于人工智能吗？？

航空领域的研究人员不会把研究目标定义为制造“能完全像鸽子一样飞行的机器，使得它们可以骗过其他真鸽子”。



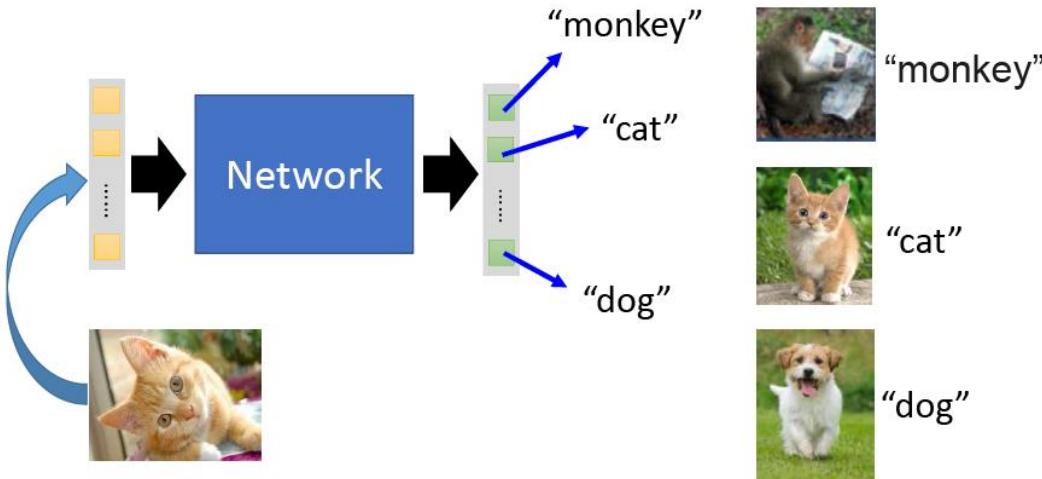
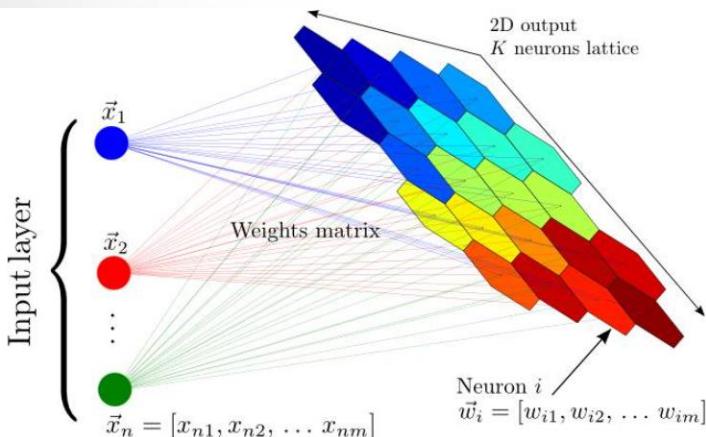


## 如何实现人工智能(AI)？

### ◆ 如何让机器像人一样“思考”？

首先，弄清楚人是如何思考，才能教会机器。这是关于逻辑学、心理学、神经科学、脑科学的复杂问题，目前尚无精确的理论。虽然人工智能通过计算机模型实现初步的思考、推理和行动，但与人脑的工作原理之间的联系尚无解释。

# 第一章：绪论

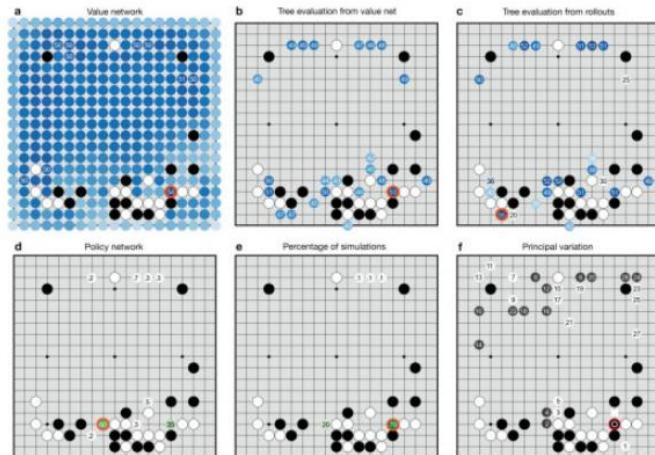


## 如何实现人工智能(AI)?

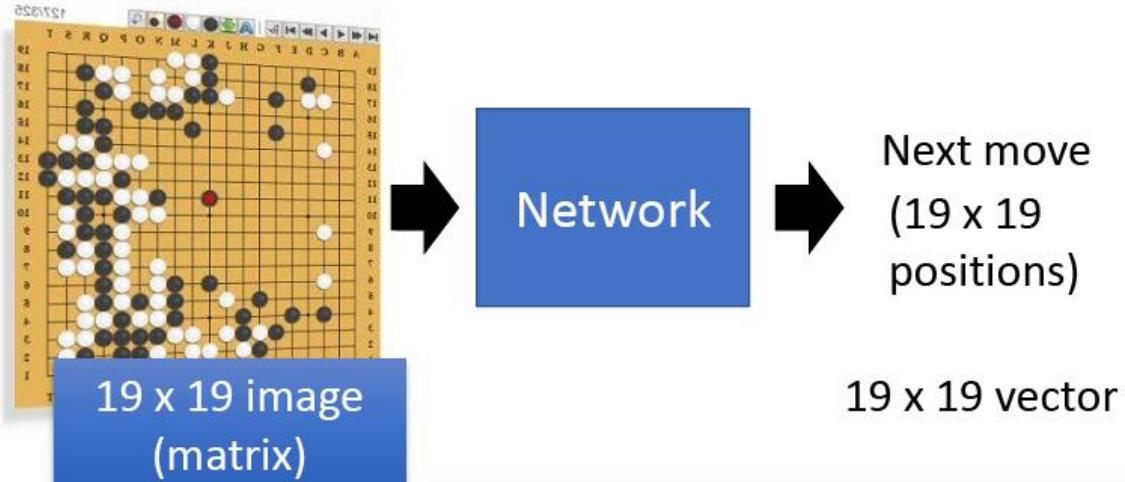
◆ 如何让机器像人一样“思考”？

目前，深度神经网络是较为热的研究方向，其能够实现高逼真的类人行为，甚至表现出超过人类的行为，也被誉为最类似人脑的人工智能技术，在自然语言处理、计算机视觉和围棋上的效果最为突出。

# 第一章：绪论



AlphaGo



如何实现人工智能(AI)?

◆ 如何让机器像人一样“思考”?

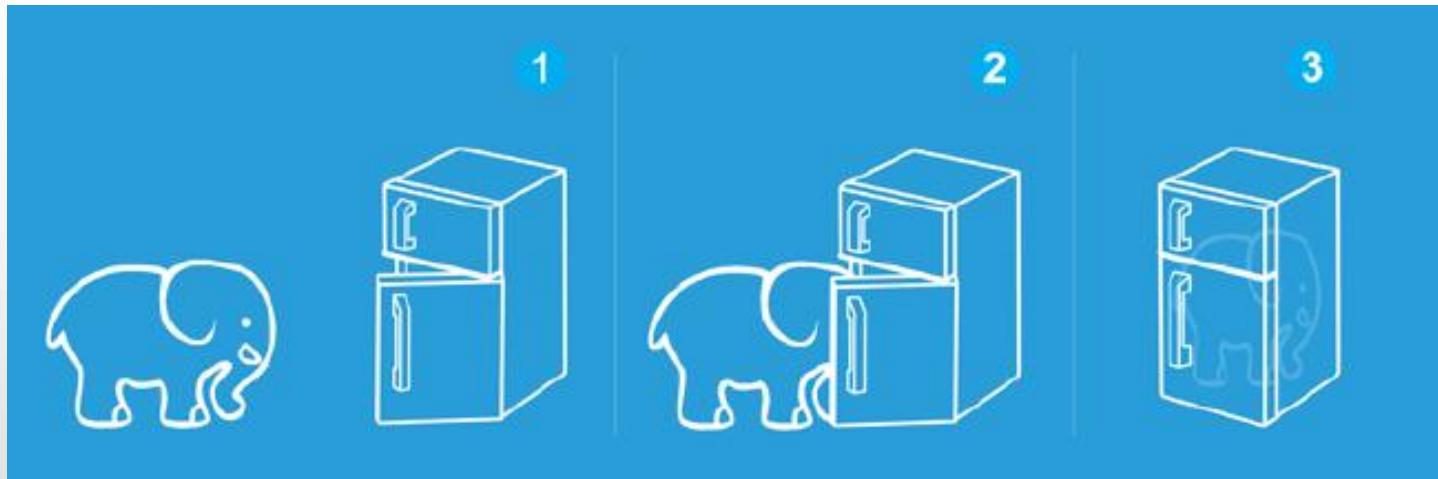
最近，类脑智能的研究逐渐展开，脑科学的研究人员开始和人工智能领域的计算机和数学研究相结合。以构造符合人脑思维的人工智能新理论和新方法。



## 如何实现人工智能(AI)？

◆ 如何让机器像人一样“思考”？

深度学习的简单框架：大象和冰箱的故事。

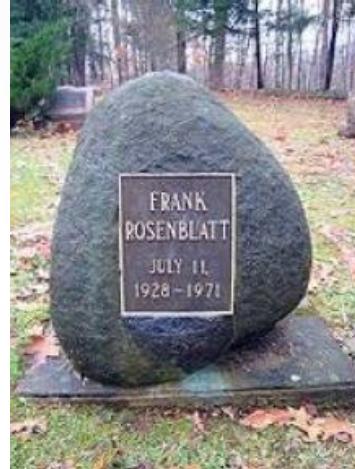


# 第一章：绪论

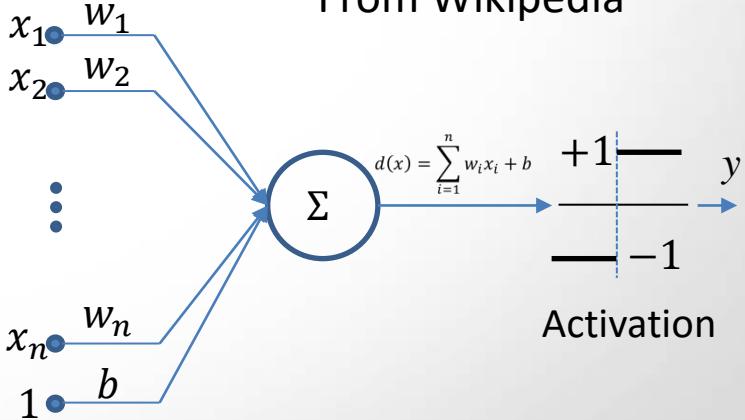
## 人工智能(AI)起源(60年前， 1950):

Rosenblatt 提出“感知器”概念，并在刚刚萌芽的AI界产生了不小的争论。

关于“认知”，由于“感知器”的提出，纽约时报称之为“可使电子计算机能够散步、说话、看、写、自我复制、产生意识的胚胎。”



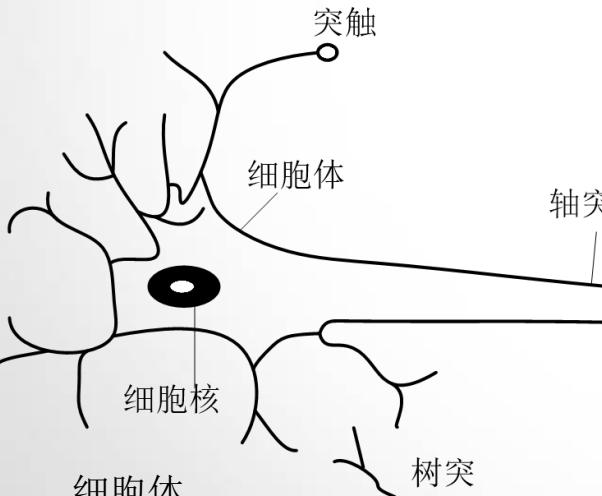
From Wikipedia



# 第一章：绪论

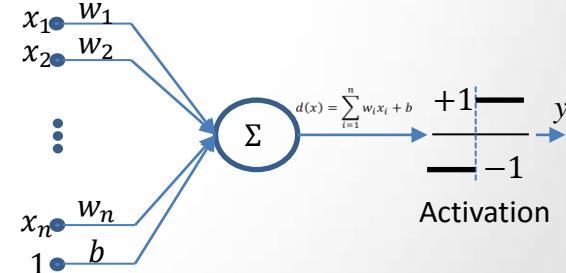
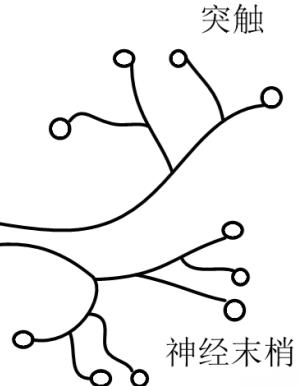
人工智能(AI)起源(60年前，1950):

生物神经元（神经细胞）结构：



生物神经元结构

每个神经元由一个细胞体组成，其包含一个细胞核。从细胞体分支扩展出许多为**树突**的神经纤维和一根长的**轴突**。一个神经元与10到10万个其他神经元相连，连接处称为**突触**。信号通过复杂的化学反应从神经元传播到神经元。研究发现，大多数的信息处理在大脑皮质，即大脑外层进行。

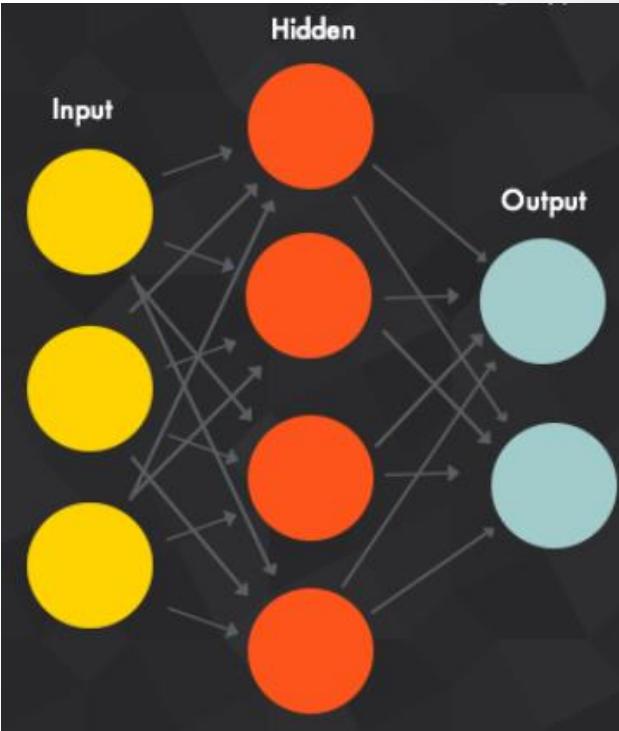
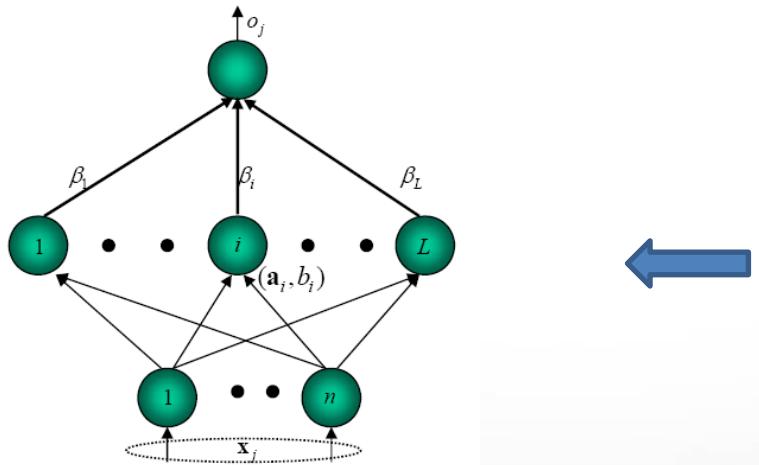


人工神经元结构



## 人工智能(AI)冬天(1970s):

人工智能冬天也是神经网络的冬天，  
Minsky说“两层的神经网络并不能解决简单的XOR问题，导致AI winter的来临”





# 第一章：绪论

## 人工智能(AI)冬天(1970s):

人工智能冬天也是神经网络的冬天，  
Minsky说“两层的神经网络并不能解决简单的XOR问题，导致AI winter的来临”

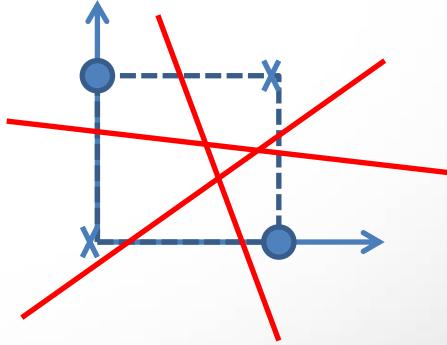
XOR(异或)问题：

0,0: 0

0,1: 1

1,0: 1

1,1: 0



无法找到一条直线将+1和-1分开

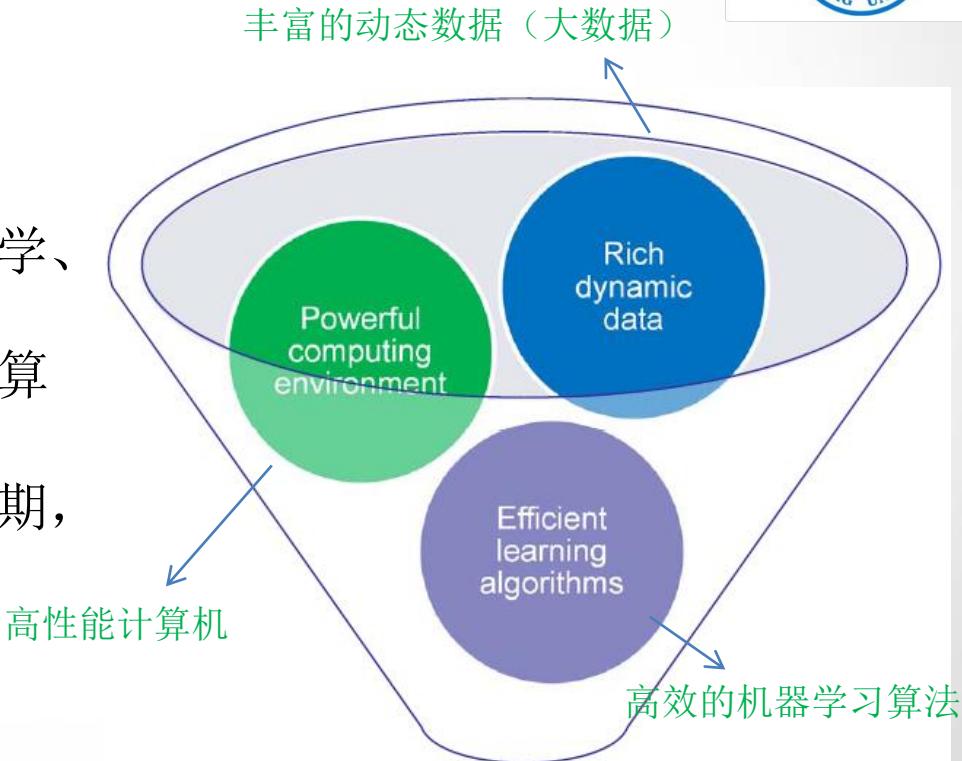


## 人工智能(AI)基础：

数学、计算科学、控制、神经科学、心理学、语言学、哲学

过去60年，人工智能发展重点在算法和模型上；

而今天，人工智能发展处于巅峰期，取决于三点：





## 人工智能(AI)最新发展：

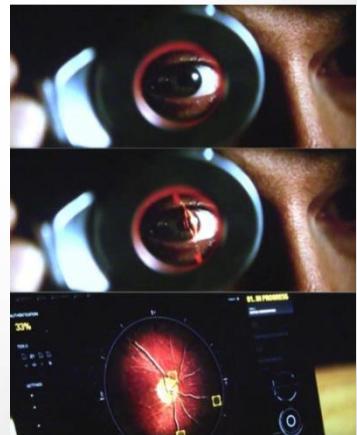
机器视、听、触、嗅、感觉及思维方式的模拟：指纹识别、人脸识别、虹膜识别、掌纹识别、图像/视频识别、语音识别、机器翻译、自然语言处理、搜索/检索、博弈（围棋、象棋）、无人驾驶等



人脸识别（谍影重重5）



步态分析与识别（碟中谍5）



视网膜识别（碟中谍5）



# 第一章：绪论



利用智能信息处理技术从大规模人脸图像数据中获取特征、知识表达、识别模型等



# 第一章：绪论

## 人工智能(AI)最新发展：

**自然语言处理(NLP)**：计算机科学和人工智能领域的重要方向之一。其研究人与计算机如何使用自然语言进行通信的理论和方法。主要包括两个方面：自然语言理解、自然语言生成。

**难点**：自然语言文本和对话存在不同的歧义性或多义性。

**存在的技术问题**：1. 局限于分析一个孤立的句子，上下文关系和谈话场景对该句子的影响缺乏系统研究。2. 人类理解一个句子不仅仅是凭语法，还运用大量有关知识，比如生活常识和专门技能，而这些知识无法存储在计算机中。



# 第一章：绪论

## 人工智能(AI)最新发展：

**图像视频理解**：人工智能和计算机视觉领域的重要方向。其研究如何利用计算机对图像和视频的内容、场景、语义的理解、标注、分类和识别。目前在视频监控、图像检索、图-文转换等领域应用较广。其实现过程分成4个层：

**数据层**：图像数据的获取（传感器）、压缩和传输（通信）；

**描述层**：特征提取，图像信号(像素)的形式化；

**认知层**：图像理解，包括机器学习和推理；

**应用层**：对特定任务（分类、识别、检测），设计模式识别和机器学习算法。



图像和文本之间的转换  
是一项具有趣味性和挑  
战性的研究



# 第一章：绪论





# 第一章：绪论





# 第一章：绪论

人工智能(AI)最新发展：

博弈：

IBM公司的深蓝(deep blue)第一个在象棋比赛中击败世界冠军(加里.卡斯帕罗夫, 1997)。

深蓝是一台超级计算机，而非智能体（Agent）；

Google公司DeepMind团队开发的AlphaGo在围棋比赛中采用最新的深度学习技术战胜世界冠军(李世石, 2016)。

AlphaGo是一个人工智能程序，是当前人工智能的标志性事件。



# 第一章：绪论

## 人工智能(AI)争议：

Von Neumann: 计算机不会有智能；

Alan Turing: 计算机是能达到人的智力水平的；

McCarthy: 人工智能的主要问题是难解的；

Minsky: 人工智能是最难的科学之一，是思维的社会无统一的知识表示和理论基础

反对派核心观点：计算机只能解决形式化的问题，而客观世界是非形式化的，变化无穷的。



## 智能信息处理 课程：

人工智能涉及多个领域，本课程主要介绍智能信息处理的新方法和新理论，主要包括智能体Agent、机器学习、演化进化算法、神经网络、深度学习、人脸识别、计算机视觉应用。



# 第二章：智能体AGENT



## 第二章：智能体Agent

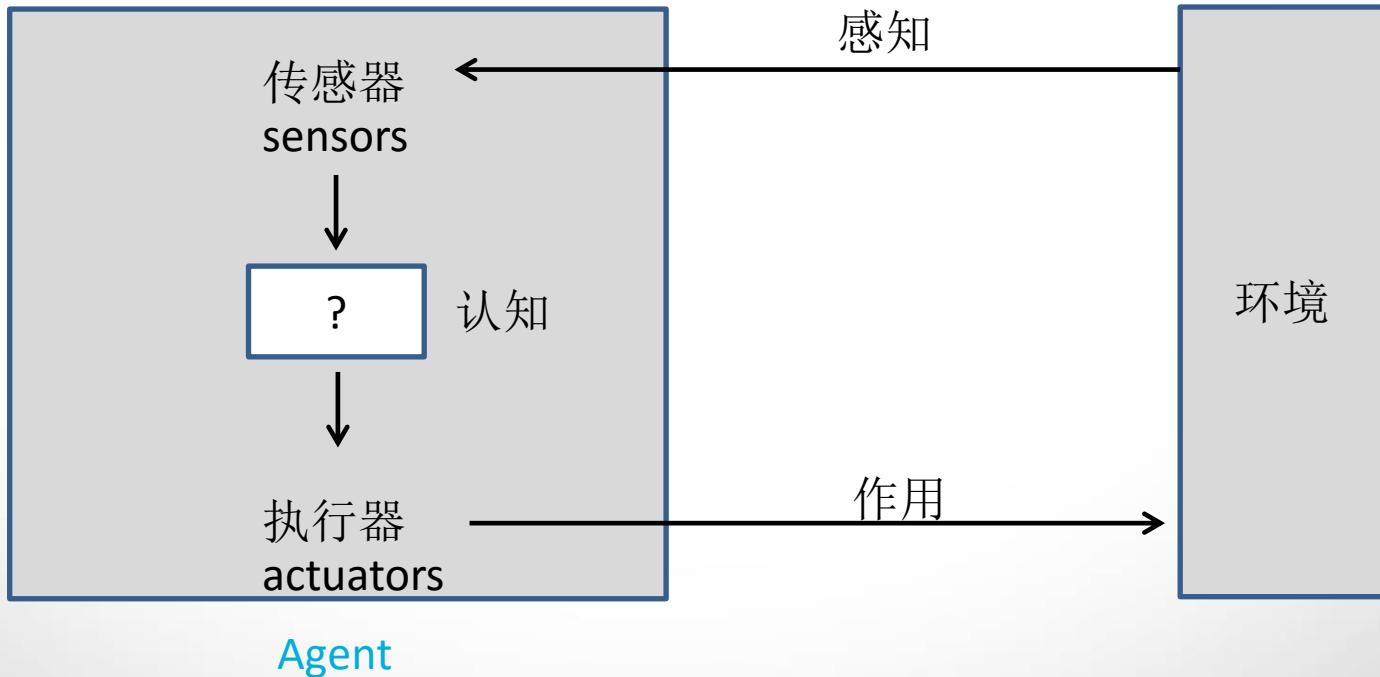
### Agent:

- ◆ 在人工智能领域，Agent有多种翻译，如“智能体”、“智能代理”，“主体”等。它可以看做是一个自动执行的实体，通过传感器感知环境，通过执行器作用于环境。
- ◆ 多Agent系统即是研究多个智能Agent的协调工作，实现问题求解。
- ◆ 构建Agent的任务，就是设计Agent程序，实现从感知到动作的映射。



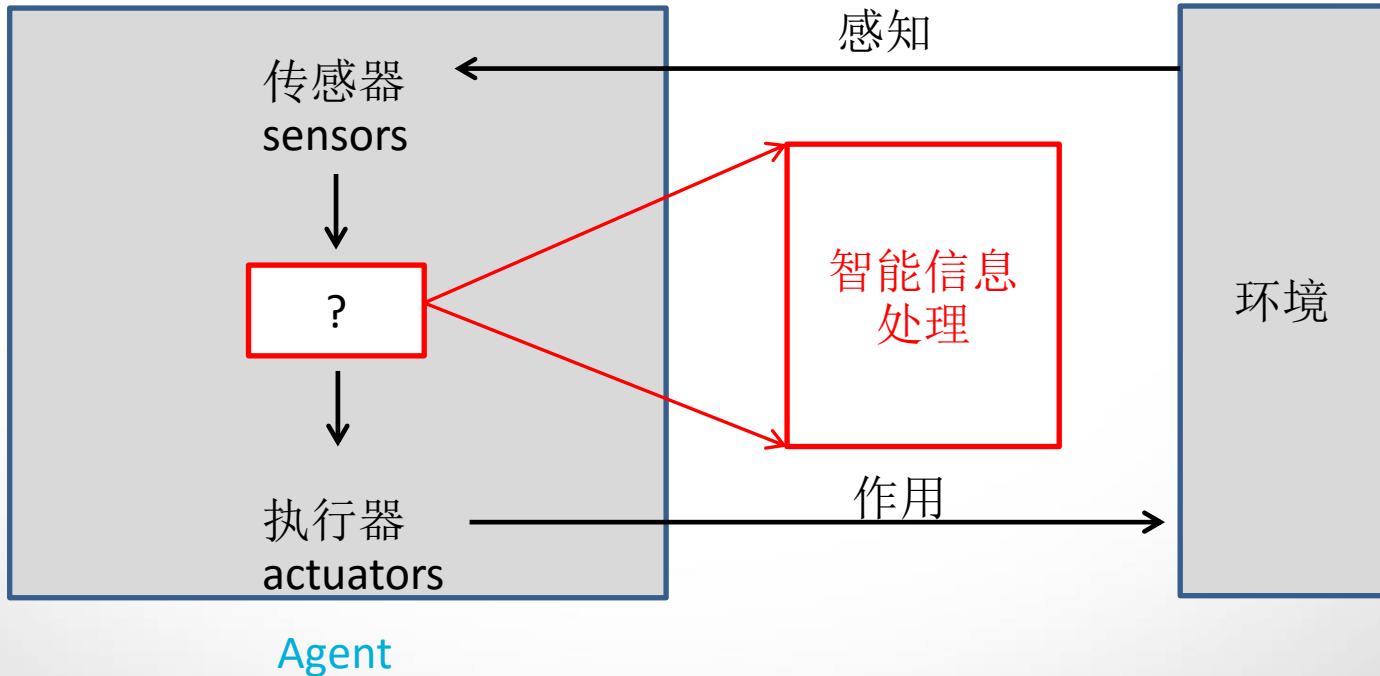
## 第二章：智能体Agent

智能体通过传感器和执行器与环境进行交互的过程：



## 第二章：智能体Agent

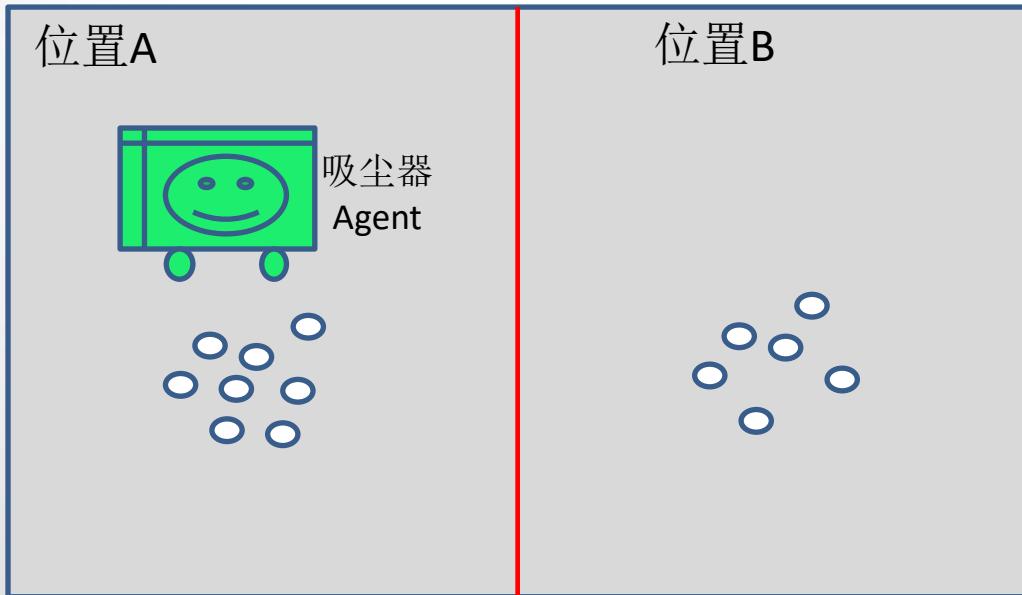
智能体通过传感器和执行器与环境进行交互的过程：





## 第二章：智能体Agent

智能体通过传感器和执行器与环境进行交互的过程：



吸尘器Agent系统

感知	Action
A, clean	right
A, dirty	Suck
B, clean	Left
B, dirty	Suck

简单的Agent函数表格



## 第二章：智能体Agent

如何定义Agent函数表格才是最好的？是什么决定了一个Agent是好还是坏？智能还是愚笨？

### □ 好的行为：理性概念

理性Agent：根据Agent行动的后果，当把Agent置于一个环境中后，它针对收到的感知信息生成一个行动序列，该行动序列会对环境产生一系列的状态变化，如果该系列是渴望的，那么该Agent性能良好。

注意：性能度量是根据环境状态变化进行评估，而不是Agent的状态。



## 第二章：智能体Agent

### 设计Agent要考虑的因素

#### □ 理性Agent

对于吸尘器，我们通过其1小时内的吸尘总量来评估Agent的性能，那么对于理性Agent来说，一边吸尘，一边把灰尘撒到地上，再吸尘，最终可以保证吸尘量的最大化，但这样好吗？

更合适的性能度量，应当奖励保持干净地面的Agent，根据环境的状态来设计性能度量，而不是根据Agent表现出的行为。



## 第二章：智能体Agent

### 设计Agent要考虑的因素

#### □ 理性Agent应具备的四点：

- ◆ 定义成功标准的性能度量。假设性能度量在每个时间步对每块清洁的方格奖励1分。
- ◆ Agent对环境的先验知识。环境的“地形”作为先验是已知的。
- ◆ Agent可以完成的行动。左、右、吸尘。
- ◆ Agent的感知序列。Agent可以正确感知位置及所在方格是否有灰尘。



## 第二章：智能体Agent

### 设计Agent要考虑的因素

#### □ 非理性Agent:

同样的Agent在不同环境下会变的非理性。一旦所有灰尘全部清洁，该吸尘器会毫无必要的来回移动，如果性能度量包含对左右移动惩罚1分，该Agent的性能评价将变的十分糟糕。

一个好的Agent需能够确保所有地方清洁干净后，不在有任何行动。如果再次弄脏，应该不定期检查，重新清洁。如果环境“地形”未知，可能还会去探查其他区域。



## 第二章：智能体Agent

### 设计Agent要考虑的因素

#### □ 理性与全知(完美)Agent:

理性是使期望的性能最大化，而完美是使实际的性能最大化。

理性的选择只依赖于截止当时为止的感知序列，还要确保没有因漫不经心而让Agent进行愚蠢的活动。

根据信息不全的感知序列，采取行动是不理性的，利用全面的感知信息，有助于期望性能最大化。比如智能体过马路。



## 第二章：智能体Agent

### 设计Agent要考虑的因素

#### □ Agent的自主性：

我们定义理性的Agent不仅要收集信息，还应当从感知信息中学习。如果系统设计时，只依赖设计人员的先验知识，Agent缺乏自主性。比如一个吸尘器Agent能够预见灰尘出现的时间和地点，显然会更加“智能”。

Agent应当具备“进化”能力。只有与“学习”相结合，才能设计出适应于不同环境的理性Agent。

“学习”是人工智能必备的能力。



## 第二章：智能体Agent

### 如何设计Agent？

#### ■ Agent的任务环境定义：

通过理性Agent，我们知道：性能度量(**Performance**)、环境(**Environment**)、执行器(**Actuators**)、传感器(**Sensors**)。把这四个因素归结在一起，就是任务环境，也称为**PEAS**描述。这是设计Agent的第一步。

考虑自动驾驶出租车Agent：

出租车任务环境的**PEAS**描述，如下：



## 第二章：智能体Agent

### 出租车任务环境的PEAS描述

Agent类型	Performance 性能度量	Environment 环境	Actuators 执行器	Sensors 传感器
自动驾驶 出租车	安全、快速、 舒适、符合交 通法规、费用 低	马路、其他车 辆、行人、顾 客、道路施工、 障碍物	油门、刹车、 发动机、转向 控制、显示器、 语音合成器与 顾客交流	摄像头、 红外设备、 声呐、加 速计、速 度表、GPS 导航、麦 克风

医疗诊断系统、卫星图像分析系统等Agent



## 第二章：智能体Agent

### 如何设计Agent?

#### ■ Agent任务环境的性质：

- ✓ 完全可观察与部分可观察；

取决于传感器，如果传感器能获取环境的完整状态，那么任务环境是完全可观察的；如果能够检测与行动决策相关的信息，那么是有效、完全可观察的。

如果因噪声、或传感器丢失状态数据，将导致任务环境为部分可观察的。如果没有传感器，则是无法观察的。

- ✓ 单Agent和多Agent

多Agents是以某种竞争或合作的形式存在。比如，国际象棋是双Agent环境，Agent A和B是以竞争的形式，比如A试图最大化自己的性能度量，而最小化对手的性能度量。两个出租车司机，则是以合作的形式避免碰撞，从而实现各自的性能度量最大化。当然，也存在部分竞争，比如两个Agent，但只有一个停车位。



## 第二章：智能体Agent

### 如何设计Agent？

#### ■ Agent任务环境的性质：

- ✓ 确定的与随机的：环境的下一个状态完全取决于当前状态和Agent执行的动作，则是确定的。否则，是随机的，比如突如其来的飞机舱门。
- ✓ 静态的与动态的：环境在Agent计算时会发生变化，比如路况是动态的。
- ✓ 已知的与未知的：比如一个Agent到了一个陌生的城市环境，自然不熟悉路况，到了一个陌生国家，对交通法规也是未知的。



## 第二章：智能体Agent

### Agent的结构？

#### ■ Agent内部工作方式：

AI的任务是设计Agent程序，实现感知信息映射到行动的Agent函数(认知过程)。程序的运行环境，即具有某个物理传感器和执行器的计算装置，也称为体系结构(载体或平台)。

$$\text{Agent} = \text{体系结构} + \text{程序}$$

体系结构可能是一台计算机，或者具有车载计算机、摄像头和其他传感器的自动驾驶汽车。

一般而言，体系架构为程序提供来自传感器的感知信息，并运行程序，并把程序计算结果传送给执行器。



## 第二章：智能体Agent

### Agent程序

#### ■ Agent程序的几种：

- 简单反射Agent：基于当前的感知信息，选择行动，不关注历史信息。比如简单的吸尘器，只建立在当前位置和是否包含灰尘。触发Agent程序的规则，称为“条件--行为规则”。
- 基于模型的反射Agent：根据感知历史，维持智能体内部状态。
- 基于目标的Agent：需要提供目标信息。比如出租车可以选择任何方向（左转、右转、直行），但如果提供目标信息后，可以做出正确的选择。
- 基于效用的Agent：仅靠目标信息，不能够做出最正确的选择，比如要考虑更安全、更快速、更可靠、更便宜的路线。



# 第三章：智能信息处理中的 数学基础



### 第三章：智能信息处理的数学基础

人工智能、智能信息处理、机器学习、模式识别的数学基础主要包括以下几个部分：

- 线性代数
- 矩阵论
- 优化理论
- 概率论
- 信息论
- 计算复杂度理论。

学好这门课，这些数学基础是必不可少的。



## 第三章：智能信息处理的数学基础

### 向量

在线性代数中，标量(scalar)是一个实数，而向量(vector)是指n个实数组成的有序数组，称为n维向量。如果没有特别说明，一个n维向量一般表示为一个列向量，即大小为nx1的列向量。向量的表示形式一般采用粗体、小写。如**a**.如果写成a，则表示一个标量。

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} \in \Re^n \text{ (n维欧几里得空间)}$$

### 常见向量

**全1向量：**指所有元素为1的向量，通常用**1<sub>n</sub>**表示,n表示向量的维度。

$\mathbf{1}_K = [1, \dots, 1]_{K \times 1}^\top$  表示K维的全1向量。



### 第三章：智能信息处理的数学基础

#### 向量的模和范数

向量 $\mathbf{a}$ 的模表示为 $\|\mathbf{a}\|$ ,计算方法为

$$\|\mathbf{a}\| = \sqrt{\sum_{i=1}^n a_i^2}$$

在线性代数中，范数(norm)是一个表示“长度”概念的函数，为向量空间内所有向量赋予非零的正长度或者大小。对于一个 $n$ 维的向量 $\mathbf{x}$ ，其常见的范数有

$L_1$  范数：

$$|\mathbf{x}|_1 = \sum_{i=1}^n |x_i|.$$

$L_2$  范数：

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{\mathbf{x}^T \mathbf{x}}.$$

欧几里得范数，向量的长度，如果向量 $\mathbf{x}$ 满足 $\|\mathbf{x}\| = 1$ ，则称向量是归一化的



## 第三章：智能信息处理的数学基础

### 向量的内积

两个具有相同维数 $n$ 的向量 $\mathbf{x}$ 和 $\mathbf{y}$ 的内积记为 $\mathbf{x}^T \mathbf{y}$ ,是一个标量。

$$\mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i$$

内积也称作标量积或点积，在泛函里面表示为 $\langle \mathbf{x}, \mathbf{y} \rangle$

### 向量的夹角

两个 $n$ 维向量的夹角定义

$$\cos \theta = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

**涵义：**向量内积是两个向量共线性的度量，说明两个向量之间的相似性。  
若 $\mathbf{x}$ 和 $\mathbf{y}$ 正交，则 $\mathbf{x}^T \mathbf{y} = 0$ 。由 $\cos \theta \leq 1$ ,  $\mathbf{x}^T \mathbf{y} \leq \|\mathbf{x}\| \|\mathbf{y}\|$  (柯西-施瓦茨不等式)



### 第三章：智能信息处理的数学基础

#### 向量的外积

两个具有相同维数 $n$ 的向量 $\mathbf{x}$ 和 $\mathbf{y}$ 的外积记为 $\mathbf{xy}$ ,是一个矩阵。

$$\begin{aligned}\mathbf{xy} &= \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} (y_1 \ y_2 \cdots y_n) \\ &= \begin{pmatrix} x_1 y_1 & \cdots & x_1 y_n \\ \vdots & \ddots & \vdots \\ x_n y_1 & \cdots & x_n y_n \end{pmatrix}\end{aligned}$$

向量外积也称作矩阵积或二元积



## 第三章：智能信息处理的数学基础

### 矩阵

一个大小为 $m \times n$ 的矩阵(matrix)是一个由 $m$ 行 $n$ 列元素排列成的矩形阵列。矩阵里的元素可以是数字、符号或数学式。在我们的课程里，矩阵一般默认为数字矩阵。

一个 $n$ 维向量，可以看作是一个 $n \times 1$ 的矩阵。

一个 $m \times n$ 的矩阵**M**，表达为  $\mathbf{M} \in \Re^{m \times n}$  (欧几里得空间)

矩阵的数学符号通常有粗体、大写表示。



## 第三章：智能信息处理的数学基础

### 矩阵的基本运算

如果  $A$  和  $B$  都为  $m \times n$  的矩阵，则  $A$  和  $B$  的加减为

$$(A + B)_{ij} = A_{ij} + B_{ij},$$

$$(A - B)_{ij} = A_{ij} - B_{ij}.$$

$A$  和  $B$  的点乘  $A \odot B \in \mathbb{R}^{m \times n}$  为

$$(A \odot B)_{ij} = A_{ij}B_{ij}.$$

一个标量  $c$  与矩阵  $A$  乘积为

$$(cA)_{ij} = cA_{ij}.$$

若  $A$  是  $m \times p$  矩阵和  $B$  是  $p \times n$  矩阵，则乘积  $AB$  是一个  $m \times n$  的矩阵

$$(AB)_{ij} = \sum_{k=1}^p A_{ik}B_{kj}$$



### 第三章：智能信息处理的数学基础

#### 矩阵的基本运算

$m \times n$  矩阵  $A$  的转置 (Transposition) 是一个  $n \times m$  的矩阵，记为  $A^\top$ ， $A^\top$  第  $i$  行第  $j$  列的元素是原矩阵  $A$  第  $j$  行第  $i$  列的元素，

$$(A^\top)_{ij} = A_{ji}.$$

如果  $A$  是对称矩阵，则  $A = A^\top$

矩阵的向量化是将矩阵表示为一个列向量。这里， $\text{vec}$  是向量化算子。设  $A = [a_{ij}]_{m \times n}$ ，则

$$\text{vec}(A) = [a_{11}, a_{21}, \dots, a_{m1}, a_{12}, a_{22}, \dots, a_{m2}, \dots, a_{1n}, a_{2n}, \dots, a_{mn}]^\top.$$



## 第三章：智能信息处理的数学基础

### 矩阵的基本运算

#### 矩阵的迹

矩阵 $\mathbf{A}$ 的迹，数学表示为 $\text{Tr}(\mathbf{A})$ 或 $\text{Trace}(\mathbf{A})$ ，即对角线元素的和。

$$\text{Tr}(\mathbf{A}\mathbf{A}^T) = \text{Tr}(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_F^2 = \sum_{i,j} A_{i,j}^2$$

#### 矩阵的逆 $\mathbf{A}^{-1}$

#### 矩阵的转置 $\mathbf{A}^T$



## 第三章：智能信息处理的数学基础

### 矩阵的基本运算

#### 矩阵与向量的乘积

$$\begin{bmatrix} m_{11} & \cdots & m_{1d} \\ \vdots & \ddots & \vdots \\ m_{n1} & \cdots & m_{nd} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

可以写另一种表达方式

$$\mathbf{M}\mathbf{x} = \mathbf{y}$$

其中，

$$y_i = \sum_{j=1}^d m_{ij}x_j \text{ (线性组合)}$$



### 第三章：智能信息处理的数学基础

#### 常见矩阵

**对称矩阵**指其转置等于自己的矩阵，即满足  $A = A^\top$ 。

**对角矩阵**（Diagonal Matrix）是一个主对角线之外的元素皆为 0 的矩阵。对角线上的元素可以为 0 或其他值。一个  $n \times n$  的对角矩阵矩阵  $A$  满足：

$$A_{ij} = 0 \text{ if } i \neq j \quad \forall i, j \in \{1, \dots, n\}$$

对角矩阵  $A$  也可以记为  $\text{diag}(a)$ ,  $a$  为一个  $n$  维向量，并满足

$$A_{ii} = a_i.$$



### 第三章：智能信息处理的数学基础

#### 常见矩阵

$n \times n$  的对角矩阵矩阵  $A = \text{diag}(a)$  和  $n$  维向量  $b$  的乘积为一个  $n$  维向量

$$Ab = \text{diag}(a)b = a \odot b,$$

其中  $\odot$  表示点乘，即  $(a \odot b)_i = a_i b_i$ 。

**单位矩阵**是一种特殊的对角矩阵，其主对角线元素为 1，其余元素为 0。  
 $n$  阶单位矩阵  $I_n$ ，是一个  $n \times n$  的方形矩阵。可以记为  $I_n = \text{diag}(1, 1, \dots, 1)$ 。  
一个矩阵和单位矩阵的乘积等于其本身。



### 第三章：智能信息处理的数学基础

## 导数

对于定义域和值域都是实数域的函数 $y=f(x)$ ,若 $f(x)$ 在点 $x_0$ 的某个邻域 $\Delta x$ 内,

极限  $f'(x_0) = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$

存在, 则称函数 $y=f(x)$ 在点 $x_0$ 处可导, 其导数为 $f'(x_0)$

函数  $f(x)$  的导数  $f'(x)$  也可记作  $\nabla_x f(x)$ ,  $\frac{\partial f(x)}{\partial x}$  或  $\frac{\partial}{\partial x} f(x)$ 。



### 第三章：智能信息处理的数学基础

#### 向量和矩阵的导数

□ 对于一个 $p$ 维的向量 $\mathbf{x} \in \Re^p$ , 函数

$y = f(\mathbf{x}) = f(x_1, x_2, \dots, x_p)$ 是一个标量, 则 $y$ 关于 $\mathbf{x}$ 的导数为

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_p} \end{bmatrix} \in \Re^p$$

→ 相当于梯度, 是由 $p$ 个偏导数组成的矢量

□ 如果 $\mathbf{y} = f(\mathbf{x}) = f(x_1, x_2, \dots, x_p) \in \Re^q$ 是一个向量,

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_q(\mathbf{x})}{\partial x_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_1(\mathbf{x})}{\partial x_p} & \dots & \frac{\partial f_q(\mathbf{x})}{\partial x_p} \end{bmatrix} \in \Re^{p \times q}$$



## 第三章：智能信息处理的数学基础

### 导数法则

#### 加（减）法则

$y = f(x), z = g(x)$  则

$$\frac{\partial(y + z)}{\partial x} = \frac{\partial y}{\partial x} + \frac{\partial z}{\partial x}$$

#### 乘法法则

(1) 若  $x \in \mathbb{R}^p$ ,  $y = f(x) \in \mathbb{R}^q$ ,  $z = g(x) \in \mathbb{R}^q$ , 则

$$\frac{\partial y^\top z}{\partial x} = \frac{\partial y}{\partial x} z + \frac{\partial z}{\partial x} y$$



## 第三章：智能信息处理的数学基础

### 导数法则

(2) 若  $\mathbf{x} \in \mathbb{R}^p$ ,  $\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^s$ ,  $\mathbf{z} = g(\mathbf{x}) \in \mathbb{R}^t$ ,  $A \in \mathbb{R}^{s \times t}$  和  $\mathbf{x}$  无关, 则

$$\frac{\partial \mathbf{y}^\top A \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} A \mathbf{z} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}} A^\top \mathbf{y}$$

(3) 若  $\mathbf{x} \in \mathbb{R}^p$ ,  $y = f(\mathbf{x}) \in \mathbb{R}$ ,  $\mathbf{z} = g(\mathbf{x}) \in \mathbb{R}^p$ , 则

$$\frac{\partial y \mathbf{z}}{\partial \mathbf{x}} = y \frac{\partial \mathbf{z}}{\partial \mathbf{x}} + \frac{\partial y}{\partial \mathbf{x}} \mathbf{z}^\top$$



## 第三章：智能信息处理的数学基础

### 导数法则

#### 链式法则

(1) 若  $\mathbf{x} \in \mathbb{R}^p$ ,  $\mathbf{y} = g(\mathbf{x}) \in \mathbb{R}^s$ ,  $\mathbf{z} = f(\mathbf{y}) \in \mathbb{R}^t$ , 则

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{y}}$$

(2) 若  $\mathbf{X} \in \mathbb{R}^{p \times q}$  为矩阵,  $\mathbf{Y} = g(\mathbf{X}) \in \mathbb{R}^{s \times t}$ ,  $\mathbf{z} = f(\mathbf{Y}) \in \mathbb{R}$ ,

$$\frac{\partial \mathbf{z}}{\partial \mathbf{X}_{ij}} = \text{tr} \left( \left( \frac{\partial \mathbf{z}}{\partial \mathbf{Y}} \right)^{\top} \frac{\partial \mathbf{Y}}{\partial \mathbf{X}_{ij}} \right)$$

(3) 若  $\mathbf{X} \in \mathbb{R}^{p \times q}$  为矩阵,  $\mathbf{y} = g(\mathbf{X}) \in \mathbb{R}^s$ ,  $\mathbf{z} = f(\mathbf{y}) \in \mathbb{R}$ , 则

$$\frac{\partial \mathbf{z}}{\partial \mathbf{X}_{ij}} = \left( \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \right)^{\top} \frac{\partial \mathbf{y}}{\partial \mathbf{X}_{ij}}$$



### 第三章：智能信息处理的数学基础

#### 导数法则

如果矩阵  $\mathbf{M} \in \Re^{n \times d}$  的每一个元素都是关于标量参数  $\theta$  的函数，那么矩阵  $\mathbf{M}$  对参数  $\theta$  的导数为

$$\frac{\partial \mathbf{M}}{\partial \theta} = \begin{pmatrix} \frac{\partial m_{11}}{\partial \theta} & \dots & \frac{\partial m_{1d}}{\partial \theta} \\ \vdots & \ddots & \vdots \\ \frac{\partial m_{n1}}{\partial \theta} & \dots & \frac{\partial m_{nd}}{\partial \theta} \end{pmatrix}$$



### 第三章：智能信息处理的数学基础

#### 导数法则

按位运算的向量函数及其导数：

定义  $\mathbf{x} = [x_1, \dots, x_d]^T \in \Re^d$ ,  $\mathbf{z} = [z_1, \dots, z_d]^T \in \Re^d$   
 $\mathbf{z} = f(\mathbf{x})$

其中  $f(\mathbf{x})$  是按位运算的，即  $z_i = f(x_i)$

那么  $f(\mathbf{x})$  对  $\mathbf{x}$  的导数为

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f(x_1)}{\partial x_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{\partial f(x_d)}{\partial x_d} \end{pmatrix}$$



# 第四章：线性建模



## 第四章：线性回归算法

### 线性建模

在智能信息处理中，一个重要且普遍的问题是推断属性变量（自变量）与响应变量（因变量）之间的函数关系，使得给定任何一个属性集合，可以通过该函数关系，预测其响应。

高数中的自变量 $x$ 与因变量 $y$ ，有以下函数关系

$$y=h(x)$$

线性建模的目的：

当给定一组  $(x,y)$  集合时，如何估计函数 $h(\cdot)$ 的表达式？



## 第四章：线性回归算法

### 线性建模

**例1：**建立一个能够执行疾病诊断的模型 $h(\cdot)$ 。

**已知条件：**已知疾病状态（健康、患病）的多个患者，以及对这些患者测量的属性（血压、心率、体重等）

**例2：**通过某用户在电影网页上的点击情况，预测该用户的喜好。

**已知条件：**已知被点击的电影种类（喜剧、动作、恐怖等），以及该用户对每个种类的点击次数。



## 第四章：线性回归算法

### 模型假设

在建立回归模型时，模型假设很重要。



$x$	$y$
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.10	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43



## 第四章：线性回归算法

### 线性建模

线性建模是机器学习中最直接的学习问题：学习属性与响应之间的线性关系。

例：建立苹果的重量预测模型：单属性问题（一维问题）

已知条件：已知N个苹果的重量 $t_1, t_2, \dots, t_N$ ，以及N个苹果的水平宽度 $x_1, x_2, \dots, x_N$ 。

任务：估计苹果的重量预测线性模型 $y = h(x) = \omega_0 + \omega_1 x$



## 第四章：线性回归算法

### 什么是好的模型？

为了选择某种方式下最好的 $\omega_0$ 和 $\omega_1$ 值，使得该直线能够尽可能与所有数据点接近的直线。

度量“好”的方法是采用“平方差”即预测的苹果重量 $y$ 与实际重量 $t$ 之间的平方差，定义为

$$(t_n - h(x_n; \omega_0, \omega_1))^2$$

该式子表示第n个苹果的预测误差，该值越小，说明模型 $h(\cdot)$ 在 $x_n$ 处的值越接近 $t_n$ 。

这个度量有个专业称呼“平方损失函数”(squared loss function)，因此，第n个苹果（样本）的损失可以定义为

$$L(t_n, h(x_n; \omega_0, \omega_1)) = (t_n - h(x_n; \omega_0, \omega_1))^2$$



## 第四章：线性回归算法

什么是好的模型？

对于全部的N个苹果，我们希望所有苹果的损失最小，因此，有平均损失，

$$L = \frac{1}{N} \sum_{n=1}^N L(t_n, h(x_n; \omega_0, \omega_1)) = \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2$$

我们建模的目的是求得最佳的 $\omega_0, \omega_1$ ,使得平均损失 $L$ 达到最小。

上面的问题可以写成专业化的机器学习模型：

$$\min_{\omega_0, \omega_1} L = \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2$$

$$\text{或者 } \{\omega_0, \omega_1\} = \arg \min_{\omega_0, \omega_1} L = \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2$$



## 第四章：线性回归算法

### 求解过程

将损失函数展开

$$\begin{aligned} L &= \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2 \\ &= \\ &\vdots \\ &= \frac{1}{N} \sum_{n=1}^N (\omega_1^2 x_n^2 + 2\omega_1 x_n(\omega_0 - t_n) + \omega_0^2 - 2\omega_0 t_n + t_n^2) \end{aligned}$$

计算损失函数L对 $\omega_0$ 和 $\omega_1$ 的偏导数，并令导数为0，可得出损失最小值时的两个估计参数。



## 第四章：线性回归算法

### 预测过程

目前，根据求解已获得了 $\omega_0$ 和 $\omega_1$ 的最佳估计值 $\hat{\omega}_0$ 和 $\hat{\omega}_1$ ，从而可以写出函数表达式

$$t = \hat{\omega}_0 + \hat{\omega}_1 x = -0.8 + 0.19x$$

此时，对于一个新的样本（苹果），经过测量，发现其宽度为**12cm**，那该苹果的重量应该是多少？

$$t = -0.8 + 0.19 \times 12 = 1.48kg$$



## 第四章：线性回归算法

### 模型的矩阵求解

继续考虑刚才的模型

$$t_n = h(x_n) = \omega_0 + \omega_1 x_n$$

可以写成

$$t_n = h(\mathbf{x}_n) = (\omega_0, \omega_1) \begin{pmatrix} 1 \\ x_n \end{pmatrix} = \mathbf{w}^T \mathbf{x}_n$$

现将 $N$ 个样本的属性向量 $\mathbf{x}_n$ 合并成一个矩阵 $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}$$

也将 $N$ 个样本的目标 $t_n$ 也合并成一个向量

$$\mathbf{t} = (t_1, t_2, \dots, t_N)^T$$



## 第四章：线性回归算法

### 模型的矩阵求解

#### 原损失函数

$$L = \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2$$

可以完全等价地转为向量和矩阵的函数，即

$$\begin{aligned} L &= \frac{1}{N} (\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{N} (\mathbf{t}^T - (\mathbf{X}\mathbf{w})^T)(\mathbf{t} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{t} + \frac{1}{N} \mathbf{t}^T \mathbf{t} \end{aligned}$$

此时，可以通过矩阵向量求导，可以直接获得 $\mathbf{w}$ , 而无需对 $\omega_0, \omega_1$ 分别求偏导。



## 第四章：线性回归算法

### 模型的矩阵求解

$$L = \frac{1}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{t} + \frac{1}{N} \mathbf{t}^T \mathbf{t}$$

此时，可以通过矩阵向量求导，可以直接获得 $\mathbf{w}$ ，而无需对 $\omega_0, \omega_1$ 分别求偏导。

因为  $\frac{\partial L}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial L}{\partial \omega_0} \\ \frac{\partial L}{\partial \omega_1} \end{bmatrix} = \frac{2}{N} \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{X}^T \mathbf{t} = 0$

可以推出

$$\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{t}$$

那么  $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$  (完毕)



## 第四章：线性回归算法

### 模型的矩阵求解

现在已经获得 $w$ 的解析表达式

$$w = (X^T X)^{-1} X^T t = \begin{bmatrix} \omega_0 \\ \omega_1 \end{bmatrix}$$

例：给出3个样本点组成的集合

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 5 \end{bmatrix}, t = \begin{bmatrix} 4.8 \\ 11.3 \\ 17.2 \end{bmatrix}$$

如何建立线性模型的表达式？

**预测：**给定一个属性 $x_{new} = [1 \ x_{new}]^T$ 的新向量，其预测公式：

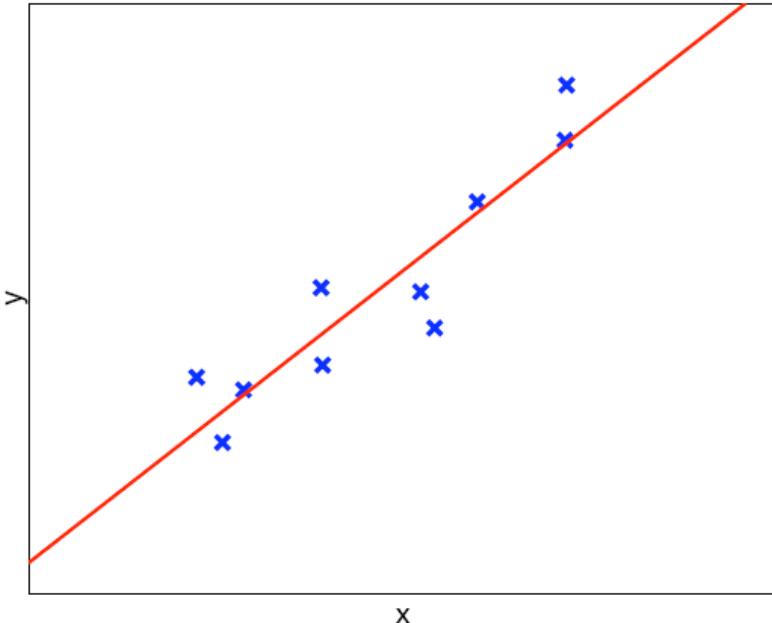
$$t_{new} = \omega_0 + \omega_1 x_{new} = [\omega_0 \ \omega_1] [1 \ x_{new}]^T = w^T x_{new}$$

## 第四章：线性回归算法



Example: Data and best linear hypothesis

$$y = 1.60x + 1.05$$





## 第四章：线性回归算法

### 线性建模-多维问题

多维问题是机器学习、实际应用中的普遍问题，即有多个属性的情况。

例：建立一个能够执行疾病诊断的线性模型：多属性问题（多维）

已知条件：已知疾病状态（健康、患病）的N个患者，以及每个患者的血压值 $x_1$ 、心率 $x_2$ 、体重 $x_3$ 等，通过多个方面的因素，来预测患者的身体状况（响应）。



## 第四章：线性回归算法

### 线性建模-多维问题

如果有d个属性，此时的属性向量 $\mathbf{x}_n$ ，可以表示为

$$\mathbf{x}_n = \begin{pmatrix} 1 \\ x_{1,n} \\ \vdots \\ x_{d,n} \end{pmatrix}$$

那么多维下的预测可以写成

$$\begin{aligned} t_n &= \omega_0 + \omega_1 x_{1,n} + \omega_2 x_{2,n} + \omega_3 x_{3,n} + \cdots + \omega_d x_{d,n} \\ &= \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_d \end{bmatrix}^T \begin{pmatrix} 1 \\ x_{1,n} \\ \vdots \\ x_{d,n} \end{pmatrix} = \mathbf{w}^T \mathbf{x}_n \end{aligned}$$

注意：在上式中， $\omega_0$ 也叫作“偏置项”。样本 $\mathbf{x}_n$ 和系数向量 $\mathbf{w}$ 是d+1维。



## 第四章：线性回归算法

### 线性建模-多维问题：模型的矩阵求解

现将 $N$ 个样本的属性向量 $\mathbf{x}_n = [x_{1,n}, x_{2,n}, \dots, x_{d,n}]^T$ 合并成一个矩阵 $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} = \begin{pmatrix} x_{1,1}, x_{2,1}, \dots, x_{d,1} \\ x_{1,2}, x_{2,2}, \dots, x_{d,2} \\ \vdots \\ x_{1,N}, x_{2,N}, \dots, x_{d,N} \end{pmatrix} \in \Re^{N \times d}$$

也将 $N$ 个样本的目标 $t_n$ 也合并成一个向量

$$\mathbf{t} = (t_1, t_2, \dots, t_N)^T$$

对于{健康、患病}两种响应，通常可以用+1和-1表示目标 $\mathbf{t}$ .



## 第四章：线性回归算法

线性建模-多维问题：模型的矩阵求解

损失函数

$$L = \frac{1}{N} (\mathbf{X}\mathbf{w} - \mathbf{t})^T (\mathbf{X}\mathbf{w} - \mathbf{t}) = \frac{1}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{t} + \frac{1}{N} \mathbf{t}^T \mathbf{t}$$

通过计算  $\frac{\partial L}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial L}{\partial \omega_1} \\ \frac{\partial L}{\partial \omega_2} \\ \vdots \\ \frac{\partial L}{\partial \omega_d} \end{bmatrix} = \frac{2}{N} \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{X}^T \mathbf{t} = 0$

可以推出

$$\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{t}$$

那么  $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$  (完毕)



## 第四章：线性回归算法

线性建模-多维问题：模型的矩阵求解

举例：设有4个患者，测量血压值、心率、体重三个属性。

现将4个样本的属性向量  $\mathbf{x}_n = [x_{1,n}, x_{2,n}, x_{3,n}]^T$  合并成一个矩阵  $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \mathbf{x}_4^T \end{pmatrix} = \begin{pmatrix} x_{1,1}, x_{2,1}, x_{3,1} \\ x_{1,2}, x_{2,2}, x_{3,2} \\ x_{1,3}, x_{2,3}, x_{3,3} \\ x_{1,4}, x_{2,4}, x_{3,4} \end{pmatrix} = \begin{pmatrix} 80, 70, 50 \\ 150, 100, 70 \\ 100, 80, 60 \\ 60, 120, 40 \end{pmatrix} \in \mathbb{R}^{4 \times 3}$$

也将  $N$  个样本的目标  $t_n$  也合并成一个向量

$$\mathbf{t} = (t_1, t_2, t_3, t_4)^T = (1, -1, 1, -1)^T$$

对于{健康、患病}两种响应，分别用+1和-1表示。



## 第四章：线性回归算法

### 线性建模-多维问题：模型的矩阵求解

一般， $\mathbf{x}$ 中的数值很大，需要对属性值进行归一化处理，以便于求解和计算。归一化方法，通常除以属性最大值实现。

$$\mathbf{x} = \begin{pmatrix} 80, 70, 50 \\ 150, 100, 70 \\ 100, 80, 60 \\ 60, 120, 40 \end{pmatrix} \rightarrow \begin{pmatrix} \frac{80}{150}, \frac{70}{120}, \frac{50}{70} \\ \frac{150}{150}, \frac{100}{120}, \frac{70}{70} \\ \frac{100}{150}, \frac{80}{120}, \frac{60}{70} \\ \frac{60}{150}, \frac{120}{120}, \frac{40}{70} \end{pmatrix} \rightarrow \begin{pmatrix} \frac{8}{15}, \frac{7}{12}, \frac{5}{7} \\ 1, \frac{5}{6}, 1 \\ 2, 2, 6 \\ \frac{2}{3}, \frac{2}{3}, \frac{4}{7} \\ \frac{2}{5}, 1, \frac{4}{7} \end{pmatrix}$$

利用解析形式  $\mathbf{w} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{t}$ ，很容易计算出模型的参数  $\mathbf{w}$



## 第四章：线性回归算法

### 普通最小二乘线性回归总结：

- 模型的最优解，可以通过最小化误差的平方和（损失函数）进行计算。

$$L = \frac{1}{N} \sum_{n=1}^N (t_n - h(x_n; \omega_0, \omega_1))^2$$

- 最小二乘法回归算法存在闭解（解析解、精确解）

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}, \text{ 其中 } \mathbf{X} \text{ 是样本数据矩阵, } \mathbf{t} \text{ 是目标值矢量。}$$

存在的问题：当 $\mathbf{X}$ 不是列满秩时， $\mathbf{X}^T \mathbf{X}$ 是奇异矩阵，那么计算 $(\mathbf{X}^T \mathbf{X})^{-1}$ 会产生较大误差，变成了一个不适定问题，该模型缺乏稳定性（对噪声的敏感性很强）。



## 第四章：线性回归算法

### 模型的性能评价

已经介绍了普通最小二乘法的原理，对于由N个样本，计算出的最小二乘模型参数w，如何说明该模型是好的呢？

- 一般地，在机器学习中，为了验证模型的好坏，还需要一组新的样本（测试样本集）进行测试，根据测试准确率，判定模型的好坏。注：测试样本集是独立于N个样本的（也称训练样本）。
- 交叉验证（Leave-one-out cross validation, LOO）：K折交叉验证，也是目前机器学习和人工智能领域具有代表性的验证模型泛化能力的训练方法。



## 第四章：线性回归算法

### 模型的性能评价

**K折交叉验证（Leave-one-out cross validation, LOO）（留一法）：**

将数据集分成相同数量的K份（K折），每份数据分别轮流作为测试集，其余的K-1份作为训练集。执行K次测试的平均精度，作为最后的模型精度。

特别注意的是：当 $K=N$ 时，N折交叉验证，变成了“留一法”，即每个样本分别轮流作为测试样本（测试集中只有一个样本），剩余的 $N-1$ 个样本作为训练集。

**为什么要提出模型性能的评价方式？？**



## 第四章：线性回归算法

### 模型的性能评价

**K折交叉验证（Leave-one-out cross validation, LOO）（留一法）：**

将数据集分成相同数量的K份（K折），每份数据分别轮流作为测试集，其余的K-1份作为训练集。执行K次测试的平均精度，作为最后的模型精度。

特别注意的是：当 $K=N$ 时，N折交叉验证，变成了“留一法”，即每个样本分别轮流作为测试样本（测试集中只有一个样本），剩余的 $N-1$ 个样本作为训练集。

**为什么要提出模型性能的评价方式？？**



## 第四章：线性回归算法

### 广义线性回归

给定一组样本： $\langle \mathbf{x}_i, y_i \rangle_{i=1 \dots m}$ ，我们要拟合下面的假设

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{K-1} w_k \phi_k(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

其中， $\phi_k$  为基函数，可以是多阶的函数，比如平方、立方等。

为了求最佳的预测系数  $\mathbf{w}$ ，我们需要最小化下列损失函数

$$\sum_{i=1}^m (y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2$$

那么最优解  $\mathbf{w}$  可以写为

$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$



## 第四章：线性回归算法

### 广义线性回归

- 由前面的分析 可知， $h_w(x)$ 对于参数 $w$ 来说，是线性模型。但并不意味着， $h_w(x)$ 是关于输入 $x$ 的线性函数，比如 $h_w(x) = wx^2$ 。  
(多项式回归)
- 因此，广义的线性模型可以写成

$$h_w(x) = \sum_{k=0}^{K-1} w_k \phi_k(x) = \mathbf{w}^T \phi(x)$$

其中， $\phi_k$  为基函数。

- 线性假设模型还可以重新写成

$$h_w(x) = \Phi \mathbf{w}$$

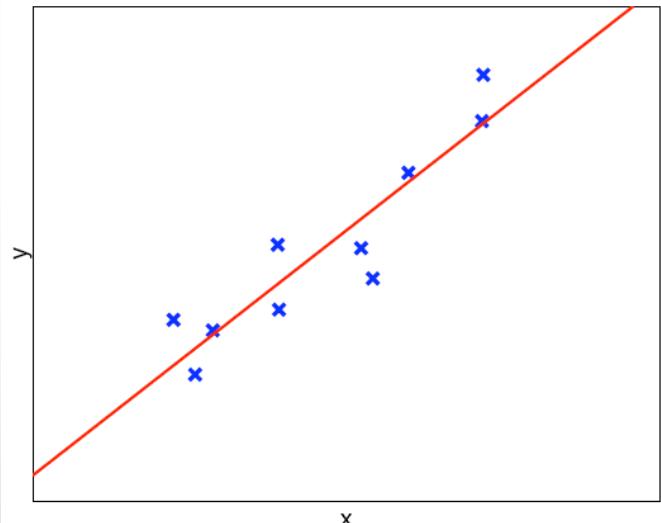
其中， $\Phi$ 是由 $\phi(x_j)$ 构成的矩阵。

→ 关于 $w$ 的  
线性模型

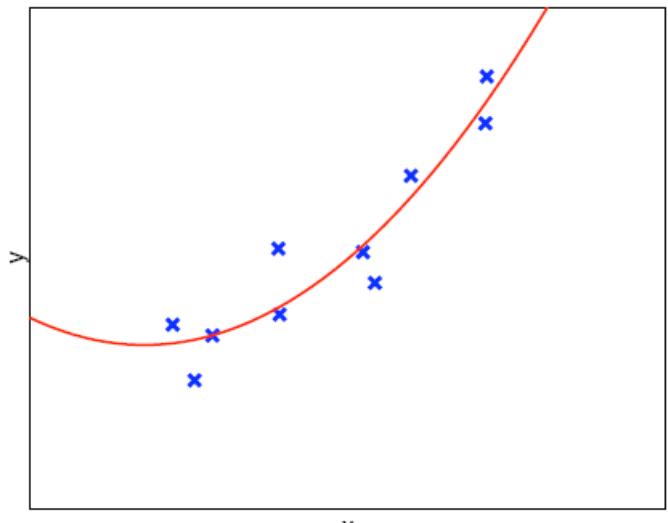


## 广义线性回归

采用不同基函数（不同阶次）的回归模型效果



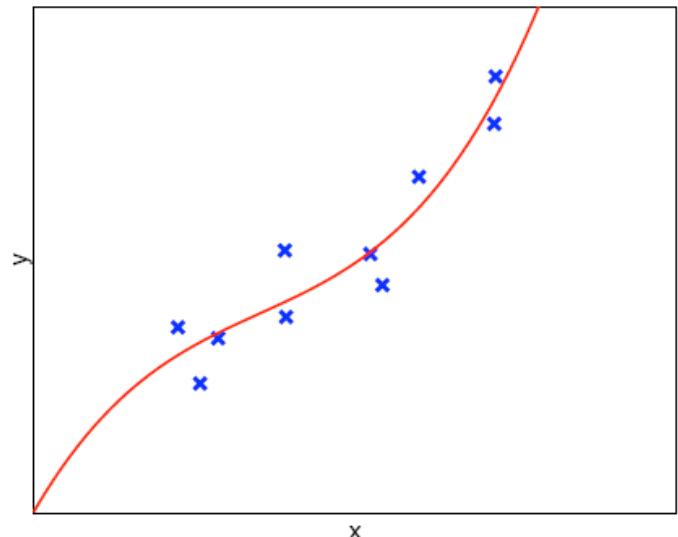
1阶回归 (is it better to fit the data?) 2阶回归



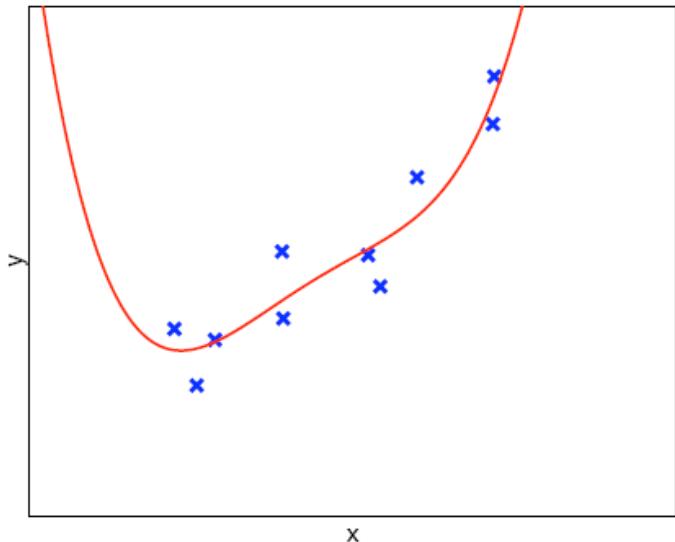


## 广义线性回归

采用不同基函数（不同阶次）的回归模型效果



3阶回归 (is it better to fit the data?)



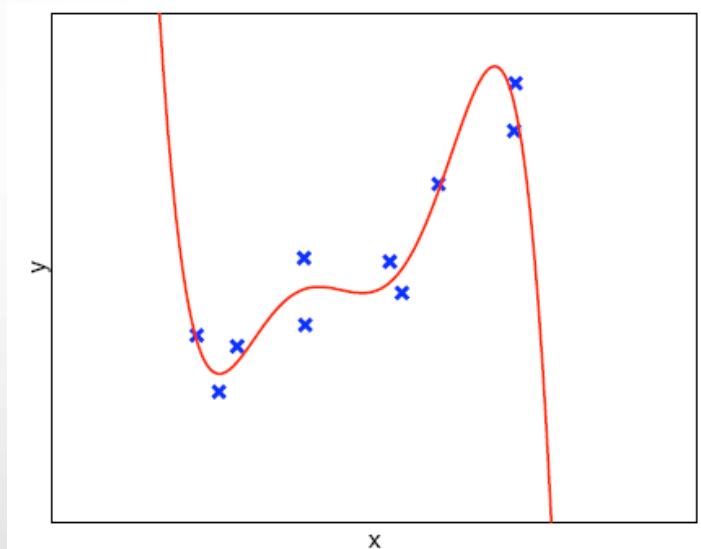
4阶回归



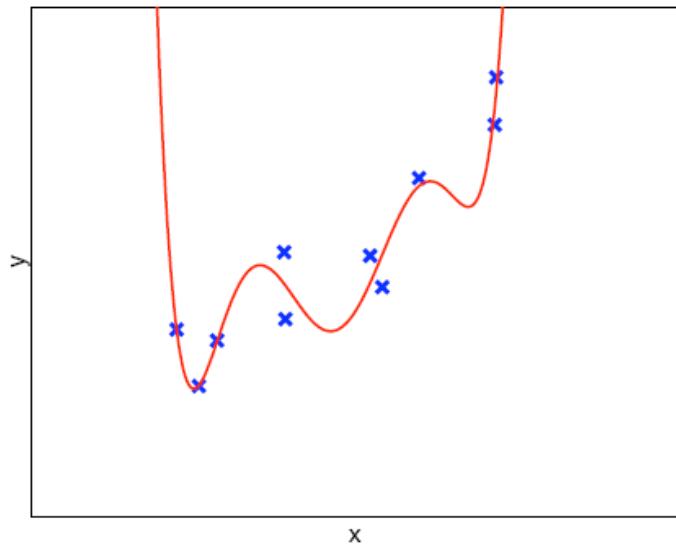
## 第四章：线性回归算法

### 广义线性回归

采用不同基函数（不同阶次）的回归模型效果



5阶回归 (is it better to fit the data?)



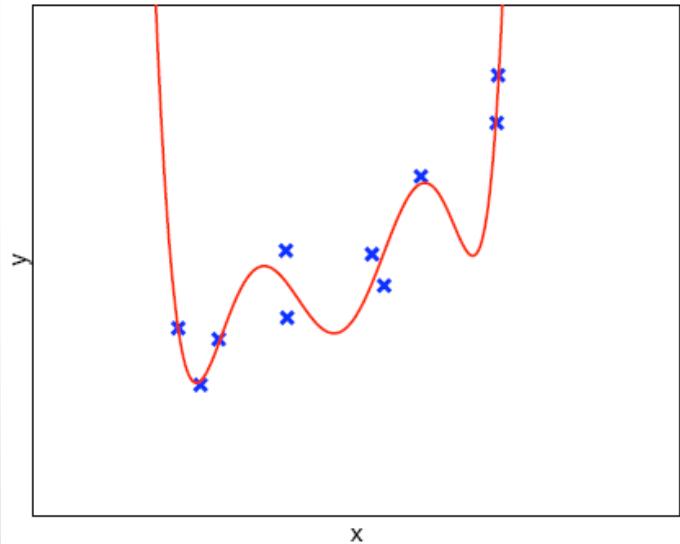
6阶回归



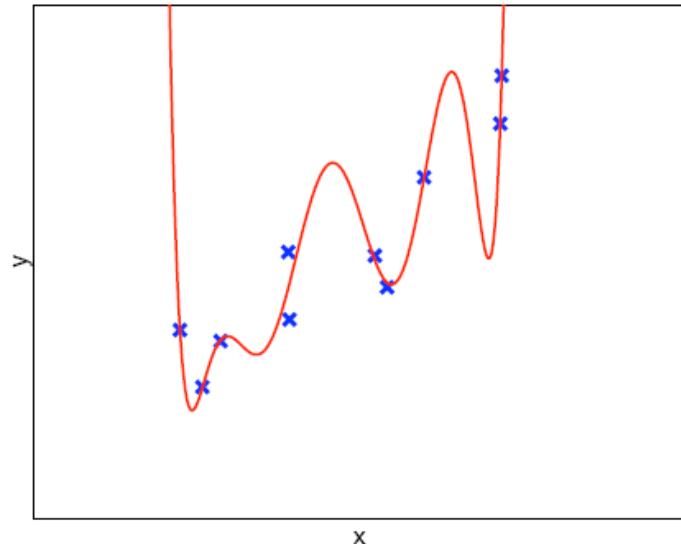
## 第四章：线性回归算法

### 广义线性回归

采用不同基函数（不同阶次）的回归模型效果



7阶回归 (is it better to fit the data?)



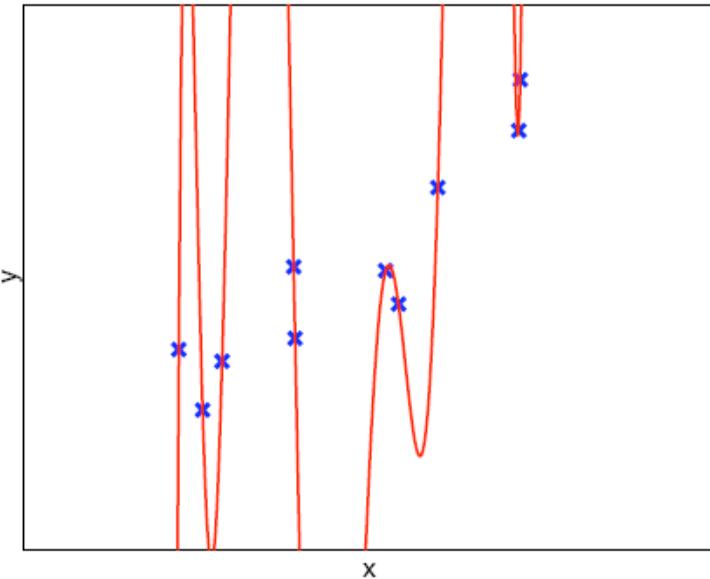
8阶回归



## 第四章：线性回归算法

### 广义线性回归

采用不同基函数（不同阶次）的回归模型效果



9阶回归 (is it better to fit the data?)



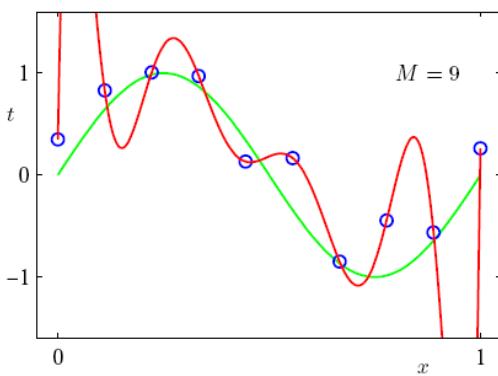
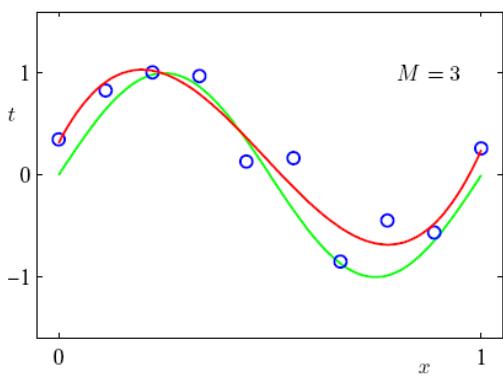
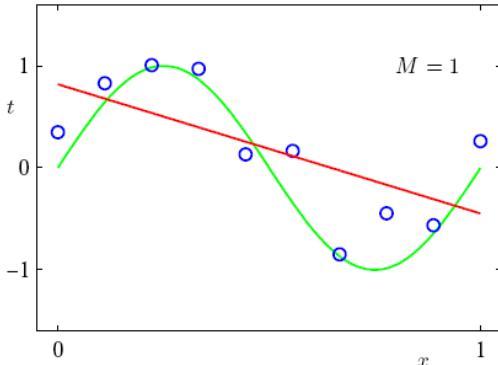
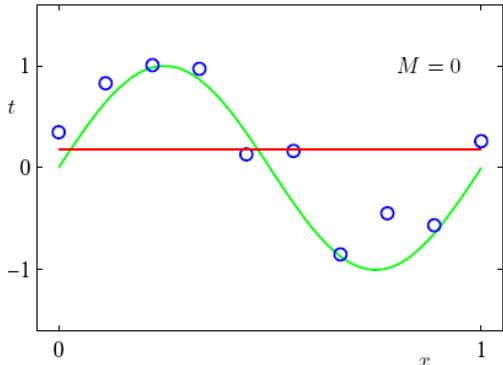
## 第四章：线性回归算法

### 过拟合（overfitting）

- 一般来说，过拟合是所有机器学习算法中的一个非常重要的问题；
- 通常，我们能够找到一种完美的假设，准确无误的预测训练数据。但是，对于新数据却不能够很好的预测（泛化能力很差）。
- 通过刚才的例子看到，如果参数越来越多，模型倾向于“**记忆**”训练数据点，而不是“**推理**”某种规律。

## 第四章：线性回归算法

### 过拟合（overfitting）与欠拟合（underfitting）

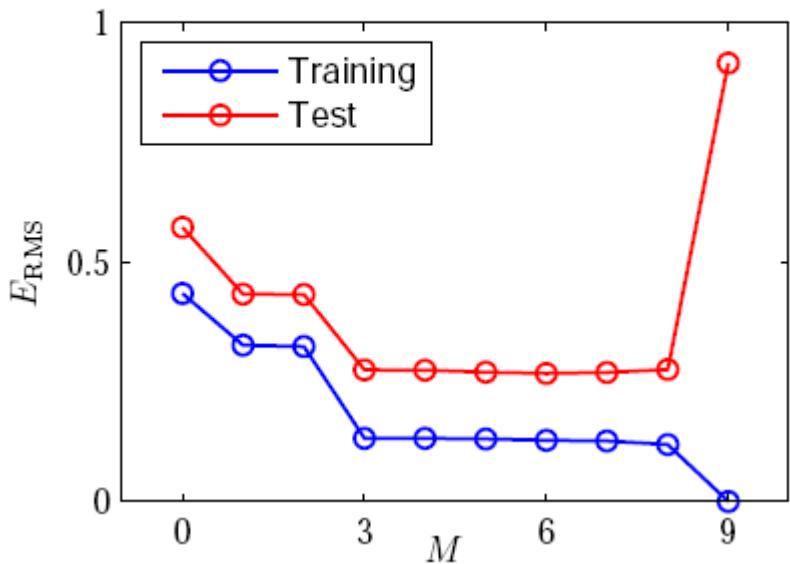


- $M$ 为模型的阶数，反映模型的自由度和复杂度。
- 过拟合是指，训练样本的误差非常低，而新测试样本的预测误差非常大。
- 欠拟合是指，训练样本的预测误差非常高（学习不够）。
- 通常，过拟合是更为常见的问题，因此，在建模过程中，我们需要经验和理论分析，来避免这个问题。

## 第四章：线性回归算法

### 过拟合 (overfitting) 与欠拟合 (underfitting)

## Typical overfitting plot



- 为从图中可以看出， $M=0,1,2$ 时，属于欠拟合（训练误差较大）
- $M=3,4,5,6,7,8$ 时，属于拟合情况良好；但根据模型的几个准则（参数越少、模型越简单越好），显然 $M=3$ 是最佳参数。
- $M=9$ 时，属于过拟合。（训练误差非常小，接近0，但测试误差非常大，）



## 第四章：线性回归算法

在设计模型寻求某种假设时，如何防止过拟合（overfitting）与欠拟合（underfitting）？

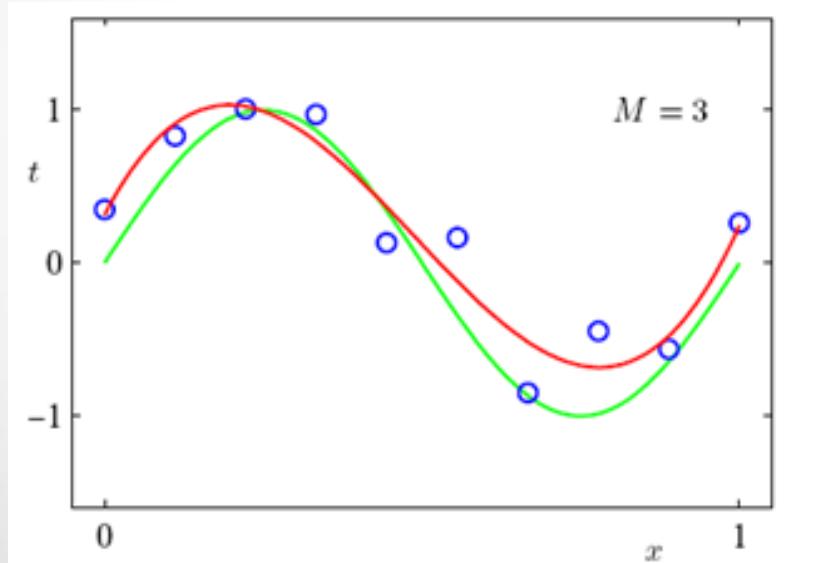
非常有效的交叉验证策略：

- 将整个数据集分成3个独立的部分：训练集、验证集、测试集
  - ✓ 训练集是用于训练某种假设；
  - ✓ 验证集用于检验假设的可靠性，并进一步修正模型（即 $M$ ）；
  - ✓ 测试集用于检验最终模型假设的性能。

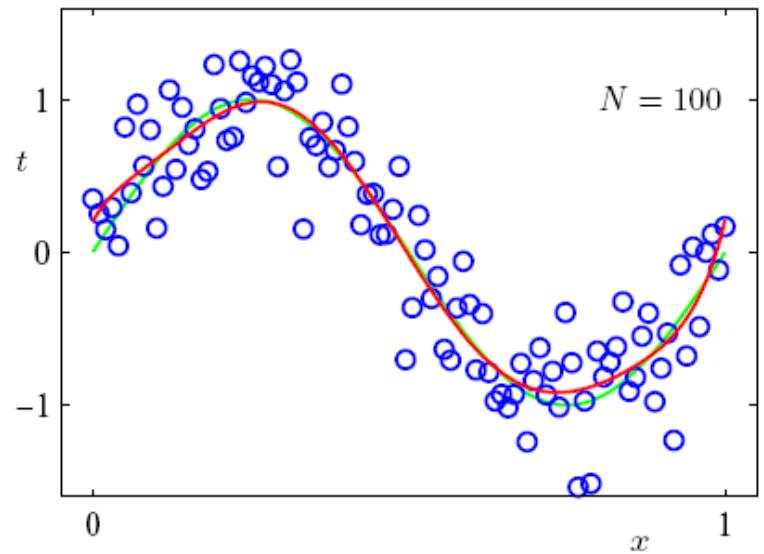
一般地，采用随机分割的形式，执行以上程序N次，每次测试集的准确率的平均值为最终的模型的性能评价指标。



最佳的假设 $h(x)$ 依赖于训练样本数量和模型复杂度



10个训练样本的回归曲线 ( $M=3$ )  
红色表示真实曲线，绿色为假设

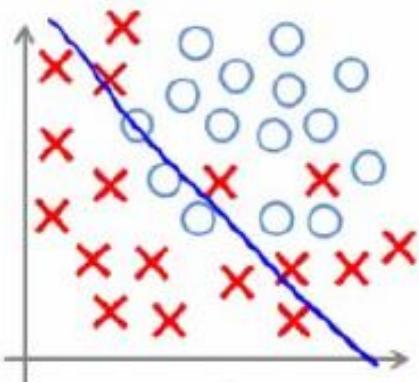


100个训练样本的回归曲线 ( $M=3$ )  
假设与真实更加接近

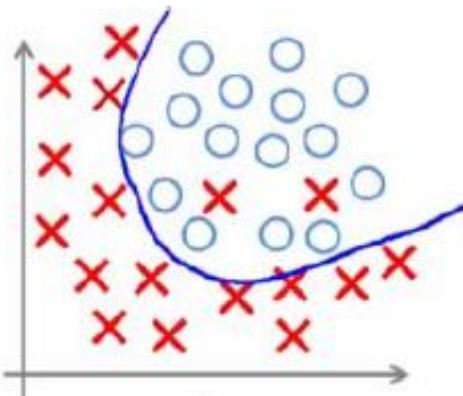
## 第四章：线性回归算法

最佳的假设 $h(x)$ 依赖于训练样本数量和模型复杂度

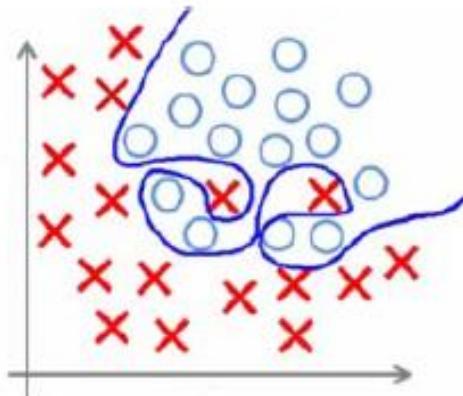
分类问题：



Under-fitting



Appropriate-fitting



Over-fitting



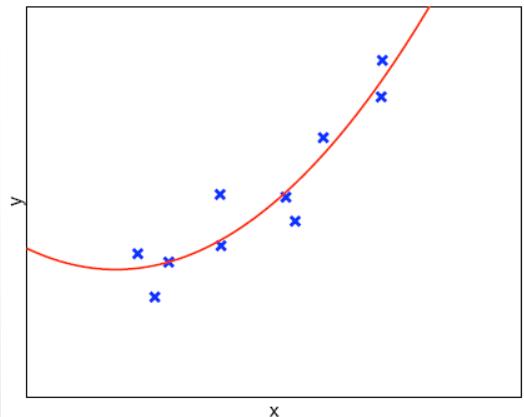
# 第五章：正则化



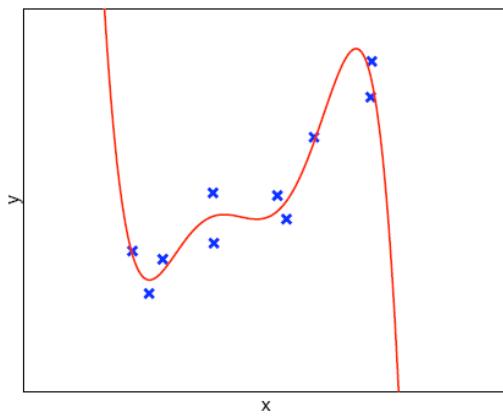
## 第五章：正则化regularization

正则化是机器学习也是智能信息处理中的重要一环，其目的是为了降低模型复杂度，增强模型的抗噪声能力，防止模型对训练数据集的过拟合（过学习）。换句话说，正则化是用于对模型参数的一种约束。

举例：一个2阶和5阶的拟合函数



$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



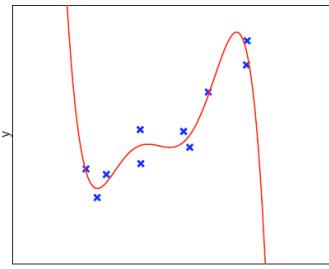
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

## 第五章：正则化regularization



原线性模型的最小化表达式为

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 \quad \rightarrow \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

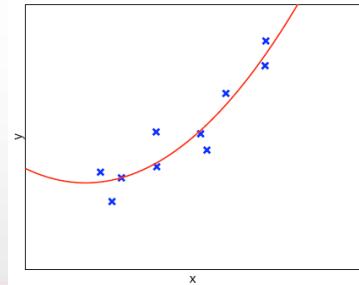


为了获得一个较好的模型，我们希望模型复杂度越小越好（阶数越小），也就是希望得到  $\theta_3, \theta_4, \theta_5 \approx 0$  (参数变少，得到简单的模型假设)  
那么模型应当改进如下

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + 10000 \cdot \theta_3^2 + 10000 \cdot \theta_4^2 + 10000 \cdot \theta_5^2$$



$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$





## 第五章：正则化regularization

正则化是机器学习也是智能信息处理中的重要一环，其目的是为了降低模型复杂度，增强模型的抗噪声能力，防止模型对训练数据集的过拟合（过学习）。换句话说，正则化是用于对模型参数的一种约束。

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + \lambda \cdot \sum_{j=1}^m \theta_j^2$$

正则化项

正则化参数  $\lambda$  是一个可调的参数（误差项和正则项之间的平衡系数），如果该值非常大，则获得的参数就会非常小！容易产生“欠拟合”，因此在算法编程中，可调试该参数。



## 第五章：正则化regularization

正则化是机器学习也是智能信息处理中的重要一环，其目的是为了降低模型复杂度，增强模型的抗噪声能力，防止模型对训练数据集的过拟合（过学习）。换句话说，正则化是用于对模型参数的一种约束。

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + \boxed{\lambda \cdot \sum_{j=1}^m \theta_j^2}$$

正则化项

实际上，正则化是对模型参数的约束，带有约束的模型可以进一步改写



## 第五章：正则化regularization

正则化是机器学习也是智能信息处理中的重要一环，其目的是为了降低模型复杂度，增强模型的抗噪声能力，防止模型对训练数据集的过拟合（过学习）。换句话说，正则化是用于对模型参数的一种约束。

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + \boxed{\lambda \cdot \sum_{j=1}^m \theta_j^2}$$

等价于

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2$$

$$s.t. \sum_{j=1}^m \theta_j^2 \leq \xi$$



正则化项

相当于在原模型基础上，加了一个约束项



## 第五章：正则化regularization

等价性证明



$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + \lambda \cdot \sum_{j=1}^m \theta_j^2$$

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2$$

$$s.t. \sum_{j=1}^m \theta_j^2 \leq \xi$$

对下面的模型利用拉格朗日乘子法构造目标函数

$$\min_{\theta} J(\theta) = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 + \lambda \cdot \left( \sum_{j=1}^m \theta_j^2 - \xi \right)$$



## 第五章：正则化regularization

### □多维下的线性回归模型表达

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  , 其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$  为属性,  $\mathbf{Y}$  为响应标签。  $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2$$

其中,  $\boldsymbol{\theta}$  是模型参数 (向量或矩阵) 。



## 第五章：正则化regularization

### □多维下的正则化线性回归模型表达

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  , 其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$  为属性,  $\mathbf{Y}$  为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2 + \lambda \cdot \|\boldsymbol{\theta}\|_2^2$$

其中,  $\boldsymbol{\theta}$  是模型参数 (向量或矩阵)。

与非正则化相比, 目标函数中多了一项, 即在最小化误差的同时, 还要保持参数值比较小。



## 第五章：正则化regularization

### □ 正则化最小二乘模型（岭回归）

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$ ，其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ，  
 $\mathbf{X}$  为属性， $\mathbf{Y}$  为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^N \left\| \boldsymbol{\theta}^T \mathbf{x}_i - \mathbf{y}_i \right\|_2^2 + \lambda \cdot \left\| \boldsymbol{\theta} \right\|_2^2$$

等价于

$$\min_{\boldsymbol{\theta}} \left\| \mathbf{X}^T \boldsymbol{\theta} - \mathbf{Y} \right\|_F^2 + \lambda \cdot \left\| \boldsymbol{\theta} \right\|_F^2$$

其中， $\boldsymbol{\theta}$  是模型参数（向量或矩阵）。



## 第五章：正则化regularization

### □ 正则化最小二乘模型（岭回归）

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$ ，其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ，  
 $\mathbf{X}$  为属性， $\mathbf{Y}$  为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \left\| \mathbf{X}^T \boldsymbol{\theta} - \mathbf{Y} \right\|_F^2 + \lambda \cdot \left\| \boldsymbol{\theta} \right\|_F^2$$

模型求解：

令  $J(\boldsymbol{\theta}) = \left\| \mathbf{X}^T \boldsymbol{\theta} - \mathbf{Y} \right\|_F^2 + \lambda \cdot \left\| \boldsymbol{\theta} \right\|_F^2$  为目标函数。

求导：

$$\frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \mathbf{X}(\mathbf{X}^T \boldsymbol{\theta} - \mathbf{Y}) + \lambda \cdot \boldsymbol{\theta} = 0 \quad \rightarrow \quad \boldsymbol{\theta} = (\mathbf{X}\mathbf{X}^T + \lambda \cdot \mathbf{I})^{-1} \mathbf{X}\mathbf{Y}$$

直接写出解析解



## 第五章：正则化regularization

### □多维下的正则化线性回归模型表达

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  , 其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$  为属性,  $\mathbf{Y}$  为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2 + \lambda \cdot \|\boldsymbol{\theta}\|_2^2$$

其中,  $\boldsymbol{\theta}$  是模型参数 (向量或矩阵)。

注: 这里正则化项的设计采用的是L2-norm(Tikhonov regularization), 是一种很平滑的约束, 但并不稀疏。



## 第五章：正则化regularization

### □吉洪诺夫正则化(Tikhonov Regularization)

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  ,其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$  为属性,  $\mathbf{Y}$  为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2 + \lambda \cdot \|\Gamma \boldsymbol{\theta}\|_2^2$$

其中,  $\Gamma$  是吉洪诺夫矩阵, 在许多情况下  $\Gamma = \alpha \cdot \mathbf{I}$ 。

对于**L2-Norm**,解的稀疏性不好, (如何能保证求解的稀疏性?)



## 第五章：正则化regularization

### 口多维下的稀疏正则化线性回归模型表达

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  ,其中  $\mathbf{X}=[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y}=[\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$ 为属性,  $\mathbf{Y}$ 为响应标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是 $d$ -维的向量。

$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2 + \lambda \cdot \|\boldsymbol{\theta}\|_0$$

$L_0$ -范数表示向量  $\boldsymbol{\theta}$  中非零元素的个数。

注: 这里正则化项的设计采用的是L0-norm,是一种强稀疏约束。

但求解时, 由于L0的非凸性, 难以求解NP-hard。通常采用松弛化。



## 第五章：正则化regularization

### 口多维下的稀疏正则化线性回归模型表达 (Lasso Regression)

定义一组观测数据集  $(\mathbf{X}, \mathbf{Y})$  , 其中  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ ,  $\mathbf{X}$  为属性,  $\mathbf{Y}$  为响应标签。  $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是  $d$ -维的向量。

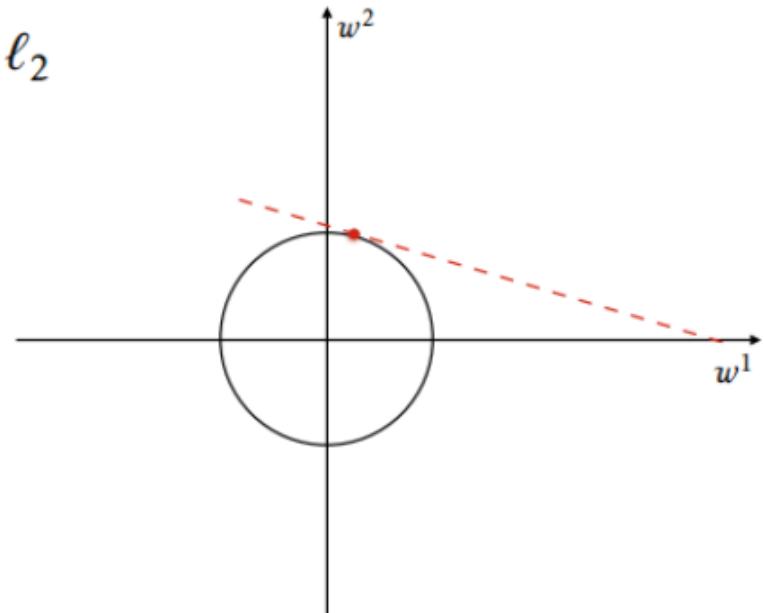
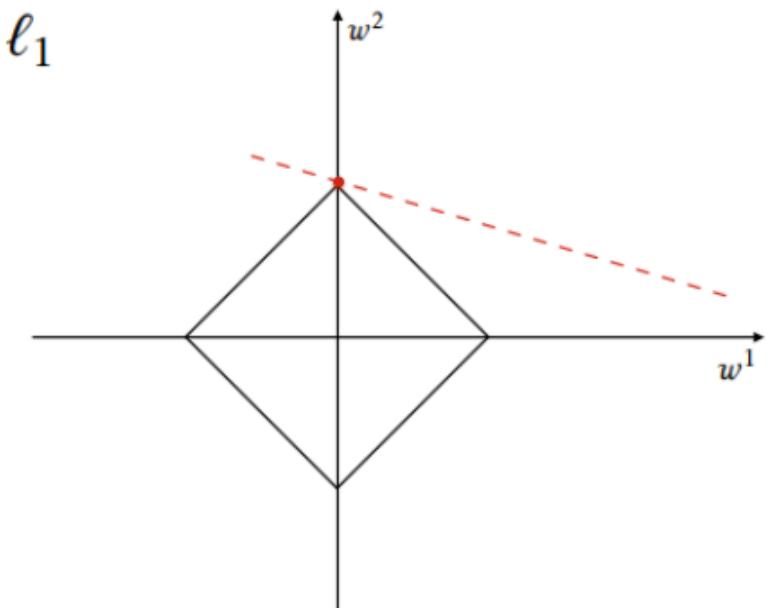
$$\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{i=1}^N \|h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2 + \lambda \cdot \|\boldsymbol{\theta}\|_1$$

L1-范数的表达式。 $\|\boldsymbol{\theta}\|_1 = \sum_{d=1}^D |\theta_d|$

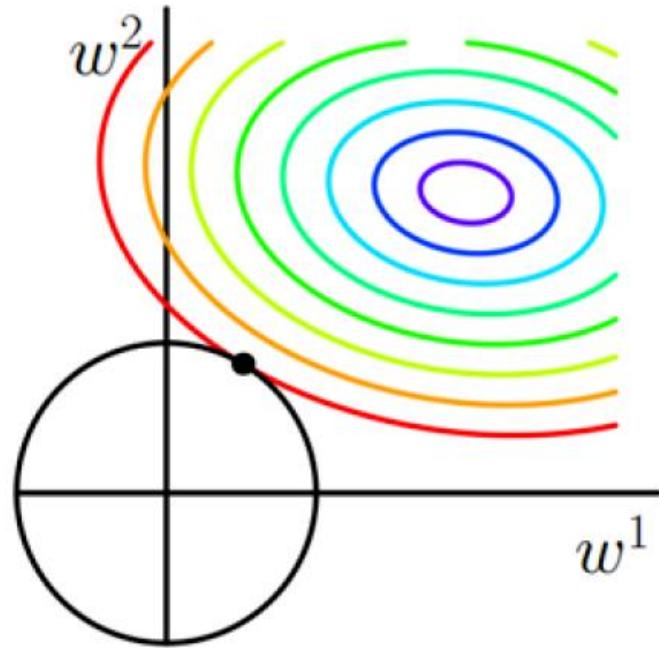
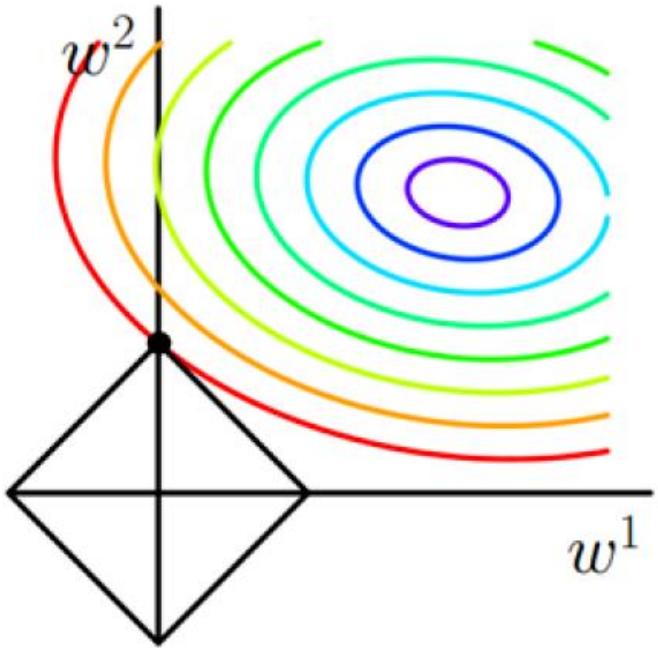
注：这里正则化项的设计采用的是L1-norm,是L0的凸松弛，近似的稀疏解，可以通过Lasso算法迭代求解(虽然凸，但在 $x=0$ 处不严格可导/可微，或者说在 $x=0$ 处不存在导数)。



## 第五章：正则化regularization



## 第五章：正则化regularization



L1有角，解落在角点的可能性更大，而L2没有角，因此不可能落在坐标系上。



## 第五章：正则化regularization

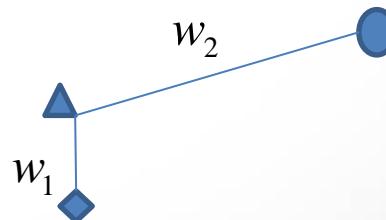
### □半监督正则化（流形正则）

定义一组新的无标签的观测数据集 $\mathbf{X}$ ,其中 $\mathbf{X}=[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ,  $\mathbf{X}$ 为属性,无标签。 $\mathbf{x}_i$  ( $i=1, \dots, N$ ) 是 $d$ -维的向量。

因为样本无标签, 则不能再用损失函数对函数 $f$ 进行定义。

**不妨做出以下假设:** 在欧式空间内, 距离较近的两个样本 $\mathbf{x}_1, \mathbf{x}_2$ , 其具有相同响应即 $f(\mathbf{x}_1)=f(\mathbf{x}_2)$ 的概率更大。

设两点之间的连线,  $w$ 存在一个权重  
即重要性系数。





## 第五章：正则化regularization

### □半监督正则化（流形正则）

按照上面的分析，半监督正则项的表达式可以写成

$$R(f) = \sum_{i,j} w_{i,j} (f(x_i) - f(x_j))^2$$

其中， $w_{i,j}$  是连接样本 $x_i$ 和 $x_j$ 之间的权重（重要性系数），属于给定的值，其定义方法可以根据欧式距离来表达。比如，

$$w_{i,j} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

显然，权重矩阵 $\mathbf{W}$ 是一个对称阵。



## 第五章：正则化regularization

□基于半监督正则化（流形正则）的线性建模

按照上面的分析，半监督正则项的表达式可以写成

$$\min_f \sum_k (f(x_k) - y_k)^2 + \alpha \cdot \|f\|_2 + \beta \cdot R(f)$$

$$\min_f \sum_k (f(x_k) - y_k)^2 + \alpha \cdot \|f\|_2 + \beta \cdot \sum_{i,j} w_{i,j} (f(x_i) - f(x_j))^2$$



## 第五章：正则化regularization

### 口正则化、泛化能力、过拟合三者关系

正则化的提出是用于提升模型的泛化能力，避免过拟合问题。

1. 泛化能力是指，训练模型不仅要保证训练集的误差最小化，同时还要保证测试集的性能，通常称为“偏置-方差”平衡问题（或者折中问题）。偏置是针对训练集，方差针对测试集。
2. 过拟合是指模型复杂度较高，导致训练集偏离程度很小（偏置很小），但测试误差的方差很大。
3. 欠拟合是指模型复杂度较低，模型越简单，训练集偏差较大；
4. 寻找泛化、欠拟合、过拟合之间的平衡问题，就是“偏置-方差”折中问题。



# 第六章：概率建模



## 第六章：概率建模

### 口产生式思考

对于任意一个样本，我们定义模型如下：

$$t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$$

其中， $\varepsilon_n$ 是一个随机变量。所谓产生式考虑，是指第n次的观测响应，是通过 $\mathbf{w}^T \mathbf{x}_n$ 生成。同时，伴随着一个随机变量，可正可负。

$\varepsilon_n$ 是一个随机变量，存在某种概率分布。

我们假设， $\varepsilon_n \sim \mathcal{N}(\mu, \sigma^2)$ 高斯分布（正态分布）

$$p(\varepsilon_n) = \mathcal{N}(\mu, \sigma^2)$$

这种假设会在后面给出解释。



## 第六章：概率建模

### 口产生式思考

对于任意一个样本，我们定义模型如下：

$$t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$$

现在，模型由两部分组成：

1. 数据生成的决定性部分， $\mathbf{w}^T \mathbf{x}_n$ ，表示一种趋势或倾向；
2. 随机组成部分 $\varepsilon_n$ ，可以理解成噪声。

注：这里的随机变量 $\varepsilon_n$ （或噪声）并非限定为高斯分布，也并非限定为可加性噪声。但为了便于解释和理解，选择可加性高斯分布噪声，可使我们获得最有参数 $\hat{\mathbf{w}}$ 的精确表达式。



## 第六章：概率建模

### 口产生式思考

将模型重新定义如下：

$$t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n, \text{ 其中, } \varepsilon_n \sim \mathcal{N}(\mu, \sigma^2)$$

这里令 $\mu=0$ , 即 $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$

由于 $\mathbf{w}^T \mathbf{x}_n$ 是一个常量, 因此加上一个随机变量 $\varepsilon_n$ 后,  $t_n$ 也是一个随机变量(这是赋予随机变量的原因)。

而且,  $t_n \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$  (为什么? 高斯随机变量加上一个常数, 依然是高斯, 仅均值发生变化), 从而概率密度函数可以写成:

$$p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t_n - \mathbf{w}^T \mathbf{x}_n)^2}{2\sigma^2}}$$

可以看出, 预测响应 $t_n$ 的密度依赖于特定的 $\mathbf{x}_n, \mathbf{w}$ (均值)和 $\sigma^2$ (方差)的值。



## 第六章：概率建模

### 口问题

我们看下面这个概率密度函数

$$p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

对于给定的  $\mathbf{w}, \sigma^2$ , 我们可以计算出在观测  $\mathbf{x}_n$  下, 使得  $p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2)$  最大的  $t_n$ , 即预测值。

对于已知的高斯分布来说, 只要均值和方差确定, 即可绘制出连续或离散的概率密度。

**例:** 设第10个苹果的直径为0.5, 即  $\mathbf{x}_{10}=[0.5, 1]^T$ ; 设  $\mathbf{w}=[0.6, -0.1]^T$ ,  $\sigma^2 = 0.1$ ; 另: 该苹果实际重量  $t_{10}=2$ .

分析: 针对该苹果, 可以绘制其重量的概率密度函数曲线

# 第六章：概率建模

## 口问题

✓ 如果  $\mathbf{w}, \sigma^2$  已经最优

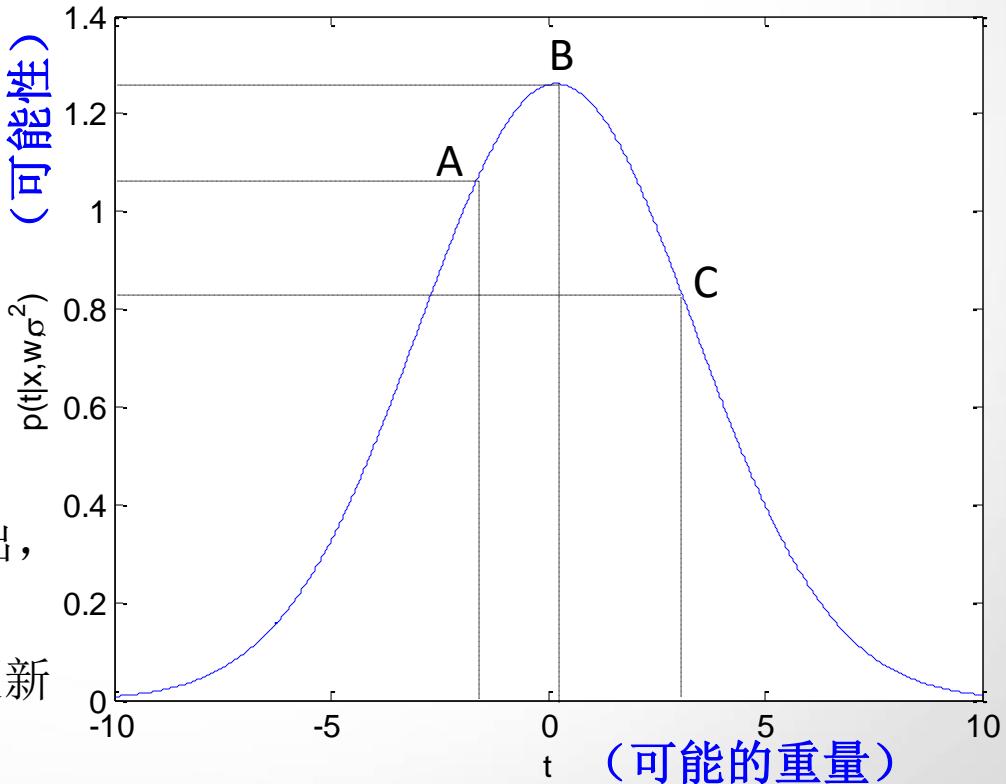
通过右边的概率密度函数可以得出：

1. B点的可能性最大，即为苹果重量的最佳估计值  $\mathbf{w}^T \mathbf{x}_{10}$
2. C点的可能性最小。

✓ 如果  $\mathbf{w}, \sigma^2$  还不是最优

实际重量  $t_{10}=2$ ，根据图可以看出，最佳的可能值是在B和C之间。

但是数据不可能变，因此需要更新分布参数。



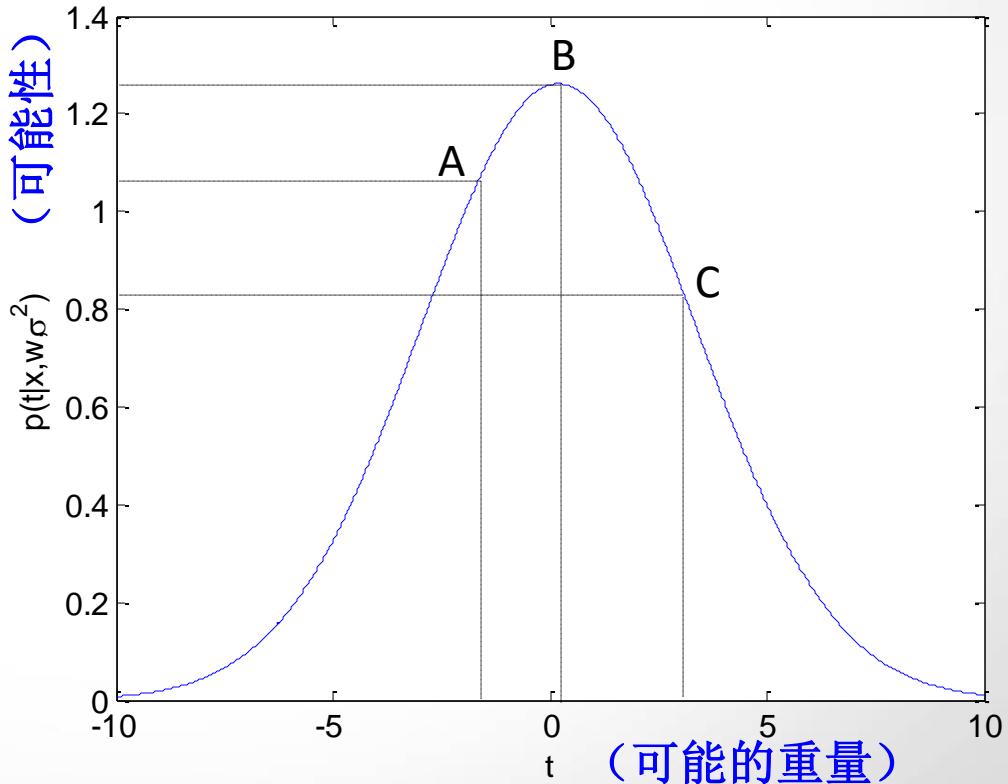
# 第六章：概率建模

## □ 定义

这种通过寻找参数，来最大化似然值得方式，称为基于似然函数最大化的线性建模——**概率建模**，是机器学习中的一种重要方法。

## □ 数据集的似然值

一般来说，我们感兴趣的是整个数据集的所有样本的似然值，而非单个样本的似然值。





## 第六章：概率建模

### □ 数据集的似然值

- 对于单个样本的概率密度表达式为

$$p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

- 如果有N个数据样本点，那么我们感兴趣的是它们的联合条件概率密度

$$p(t_1, t_2, \dots, t_N | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N, \mathbf{w}, \sigma^2)$$

- 根据向量和矩阵表示法，可以写成紧缩的表达形式：

$$p(\mathbf{t} | \mathbf{X}, \mathbf{w}, \sigma^2)$$

- 假设数据的噪声是独立的，即噪声的联合概率密度可表达为

$$p(\boldsymbol{\varepsilon}) = p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N) = \prod_{n=1}^N p(\varepsilon_n)$$

这一独立性假设，可以使密度进行分解更易操作的对象，使问题求解更简单



## 第六章：概率建模

### □ 数据集的似然值

➤ 基于噪声的独立性假设，对N个数据样本点，它们的联合条件概率密度

$$\begin{aligned} p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) &= p(t_1, t_2, \dots, t_N | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N, \mathbf{w}, \sigma^2) = \prod_{n=1}^N p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) \\ &= \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \end{aligned}$$

注：这里不是说 $\mathbf{t}$ 是独立的，否则不存在对数据的建模。而是条件独立，可以根据模型 $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$ 理解。当 $\mathbf{w}^T \mathbf{x}_n$ 给定时， $t_n$ 独立，即条件独立。  
或者说，只有当模型确定了， $t_n$ 才是一个独立的随机变量。

另： $\prod_{n=1}^N$  表示连乘符号。



## 第六章：概率建模

### □ 最大似然建模

- 最大似然建模，就是为了寻找 $\mathbf{w}$ 和 $\sigma^2$ ，使得似然值最大化。也就是最大化下列似然函数。

$$\max_{\mathbf{w}, \sigma^2} p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = p(t_1, t_2, \dots, t_N | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N, \mathbf{w}, \sigma^2)$$

$$\propto \max_{\mathbf{w}, \sigma^2} \prod_{n=1}^N p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

换句话说，当给定 $\mathbf{w}$ 和 $\sigma^2$ 时，可以获得数据集的似然值 $L = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2)$ ，但是，该似然值并不是最大的，因此，我们要改变 $\mathbf{w}$ 和 $\sigma^2$ 使得 $L$ 最大。

当然，为什么不改变 $\mathbf{X}$ 呢？数据！！

因为数据集是固定的，不同的参数值会产生不同似然值，建模的目的是要求出最佳的参数值。



## 第六章：概率建模

### □ 最大似然建模

➤ 最大似然建模求解：即求解 $\mathbf{w}$ 和 $\sigma^2$ ，使得似然值最大化。

$$\max_{\mathbf{w}, \sigma^2} \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

很显然，要求解上面的模型，可以最大化似然值得自然对数，用 $\log$ 表示，即，

$$\max_{\mathbf{w}, \sigma^2} \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \propto \max_{\mathbf{w}, \sigma^2} \log \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

➤ 似然函数的自然对数可以简化

$$\begin{aligned} \log \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) &= \log \mathcal{N}(\mathbf{w}^T \mathbf{x}_1, \sigma^2) + \cdots + \log \mathcal{N}(\mathbf{w}^T \mathbf{x}_N, \sigma^2) \\ &= \sum_{n=1}^N \log \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \end{aligned}$$



## 第六章：概率建模

### □ 最大似然建模

➤ 似然函数的自然对数可以简化

$$\begin{aligned}\log L &= \log \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) = \sum_{n=1}^N \log \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \\ &= \sum_{n=1}^N \log \left( \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t_n - \mathbf{w}^T \mathbf{x}_n)^2}{2\sigma^2}} \right) \\ &= \sum_{n=1}^N \left( -\frac{1}{2} \log(2\pi) - \log \sigma - \frac{1}{2\sigma^2} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \right) \\ &= -\frac{N}{2} \log(2\pi) - N \log \sigma - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2\end{aligned}$$



## 第六章：概率建模

### □ 最大似然建模

似然函数的自然对数最大化模型可以简化

$$\max_{\mathbf{w}, \sigma^2} \log L = -\frac{N}{2} - \log(2\pi) - N \log \sigma - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

➤ 利用前面学习的内容，对 $\mathbf{w}$ 求偏导，

$$\begin{aligned} \frac{\partial \log L}{\partial \mathbf{w}} &= \frac{1}{\sigma^2} \sum_{n=1}^N \mathbf{x}_n (t_n - \mathbf{x}_n^T \mathbf{w}) = \frac{1}{\sigma^2} \sum_{n=1}^N \mathbf{x}_n t_n - \frac{1}{\sigma^2} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \mathbf{w} \\ &= \frac{1}{\sigma^2} (\mathbf{X}^T \mathbf{t} - \mathbf{X}^T \mathbf{X} \mathbf{w}) \end{aligned}$$

令  $\frac{\partial \log L}{\partial \mathbf{w}} = \mathbf{0}$ ，可以得出  $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$  ( $\mathbf{w}$ 的最大似然解) 似曾相识！？  
将噪声假设为高斯分布，最小化平方损失等同于最大似然解。



## 第六章：概率建模

### □ 最大似然建模

似然函数的自然对数最大化模型可以简化

$$\max_{\mathbf{w}, \sigma^2} \log L = -\frac{N}{2} - \log(2\pi) - N \log \sigma - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

➤ 对 $\sigma$ 求偏导，

$$\frac{\partial \log L}{\partial \sigma} = -\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

令 $\frac{\partial \log L}{\partial \sigma} = 0$ ，可以得出 $\widehat{\sigma^2} = \frac{1}{N} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2 = \frac{1}{N} (\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w})$

最终， $\widehat{\sigma^2} = \frac{1}{N} (\mathbf{t}^T \mathbf{t} - \mathbf{t}^T \mathbf{X}\mathbf{w})$



## 课外任务：

根据前面的内容，请自行推导最大似然建模方法的整个过程。



## 第六章：概率建模

### □ Logistic回归

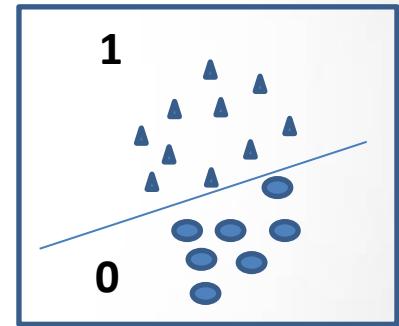
回顾线性回归算法中的二类分类问题

$$h(\mathbf{x}) = w_0 + w_1x_1 + \cdots + w_dx_d$$

分类边界为

$$h(\mathbf{x}) = 0$$

- 当  $h(\mathbf{x}) > 0$ ,  $\mathbf{x}$  的属性为  $Y=1$
- 当  $h(\mathbf{x}) < 0$ ,  $\mathbf{x}$  的属性为  $Y=0$



也就是说，对于一次观测  $\mathbf{x}$ ，其仅仅给出“是”或者“不是”的猜测（**简单、粗暴**），给出的预测是一个“肯定发生”或“肯定不发生”的事件。

如果提供一个关于可能性大小的结果岂不是更好？？换句话说，“可能性”是关于概率的一个概念，如果对于一个观测，能够计算出该观测的事件发生的概率（即  $Y=1$  发生的概率），通常比直接给出“是”或者“不是”更加有用。



## □ Logistic回归

几个特点：

- Logistic回归模型作为一种概率模型，可用于预测某事件发生的概率；
- Logistic回归模型不要求因变量在正态分布的前提下完成。
- 该模型在疾病诊断、预测中应用较广，主要用于分析不同变量因素对疾病的影响。在医学领域，一些随机事件只具有两种互斥结果的离散性随机事件。这类只具有两种互斥结果的离散型随机事件的规律性进行描述的一种概率分布，叫二项分布。
- 二项分布概率公式：如果进行n次伯努利试验，取得成功次数(事件发生)为k( $k=0,1,\dots,n$ )的概率，表示为

$$P(\xi = k) = C_n^k p^k (1 - p)^{n-k}$$

其中，事件发生的概率为p，事件不发生的概率为1-p。 $\xi \sim B(n, p)$ ，当n=1时，二项分布也可称为伯努利分布或0-1分布（1次伯努利试验）



## 第六章：概率建模

### □ Logistic回归

条件概率：

对于观测输入变量 $X$ , 其响应 $Y$ 的条件概率分布表达为

$$P(Y|X)$$

这个概率表示模型的准确性。

设想，如果对于“今天是否下雪”的预测结果是51%，但是并没有下雪。那么这个预测也要比预测为下雪的可能性为99%要好！

设 $P(Y = 1|X = x) = p(x; w)$ , 其中 $w$ 为参数，那么有

$$P(Y = 0|X = x) = 1 - p(x; w)$$

Y的取值为0  
或1



## 第六章：概率建模

### □ Logistic回归

条件概率：

对于一个观测 $x_i$ ,其响应 $Y = y_i$ 的条件概率可以写成

$$P(Y = y_i | X = x_i) = p(x_i; \mathbf{w})^{y_i} (1 - p(x_i; \mathbf{w}))^{1-y_i}$$

由于N次观测的独立性，联合条件概率分布(似然)可以写为

$$\prod_{i=1}^N P(Y = y_i | X = x_i) = \prod_{i=1}^N p(x_i; \mathbf{w})^{y_i} (1 - p(x_i; \mathbf{w}))^{1-y_i}$$

N个观测的似然函数

Y的取值为0  
或1



## 第六章：概率建模

### □ Logistic回归

主要思想： logistic回归模型的输出是某次观测事件发生的概率，通过观测样本 $x$ 的特征与最佳估计参数相乘、求和后，然后利用 Logistic函数（sigmoid）进行非线性计算（概率），然后与阈值进行比较，得出该观测 $x$ 的输出。

相乘、求和：

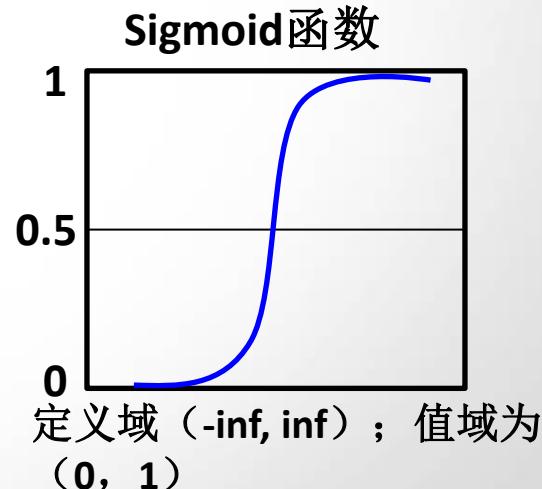
$$h(x) = w_0 + w_1x_1 + \dots + w_dx_d$$

条件概率 $P(y = 1|x)$ 的计算表达式为

$$P(y = 1|x) = p(x) = \frac{1}{1 + e^{-h(x)}}$$

条件概率 $P(y = 0|x)$ 的计算表达式为

$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - p(x) = \frac{1}{1 + e^{h(x)}}$$





## 第六章：概率建模

### □ Logistic回归

$P(y = 1|x) = \frac{1}{1+e^{-h(x)}}$  为事件发生的概率；

$P(y = 0|x) = \frac{1}{1+e^{h(x)}}$  为事件不发生的概率；

那么，事件发生与事件不发生的概率比值为

$$\frac{P(y = 1|x)}{P(y = 0|x)} = \frac{p(x)}{1 - p(x)} = \frac{1 + e^{h(x)}}{1 + e^{-h(x)}} = e^{h(x)}$$

两边同时取对数，

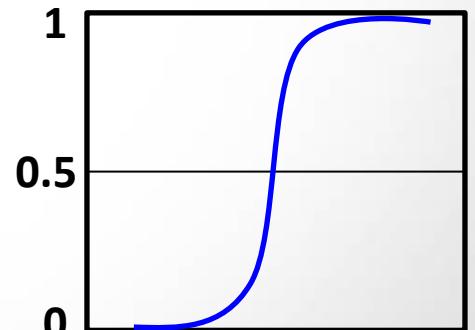
$$\log \frac{p(x)}{1 - p(x)} = h(x) = w_0 + w_1x_1 + \dots + w_dx_d$$

得到线性函数（即 Logistic 回归模型）。

利用上式，可以求出  $p(x) = \frac{e^{h(x)}}{1+e^{h(x)}} = \frac{1}{1+e^{-h(x)}}$  介于 (0,1)。

采用Logit变换，建立与属性变量之间的关系。  
当 $0 < p < 1$ 时，logit变换取任意值

Sigmoid函数



定义域 (-inf, inf)；值域为 (0, 1)



## 第六章：概率建模

### □ Logistic回归

□  $p(x) = \frac{e^{h(x)}}{1+e^{h(x)}} = \frac{1}{1+e^{-h(x)}}$  介于 (0,1)。

根据观测样本x的概率表示可以看出，当 $p(x) > 0.5$ , Y=1;

当 $p(x) < 0.5$ , Y=0。

□ 这也同时意味着， $h(x) > 0$ 时，Y=1； $h(x) < 0$ 时，Y=0。 1

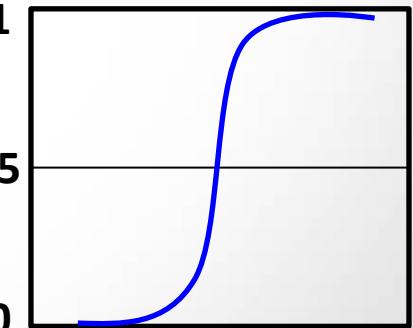
即，将两类分开的判决边界为 $h(x) = 0$ 。

□ 但Logistic回归不同于线性分类器，不是为了找分类边界，0.5

而是找出一个概率值，对“可能性”进行建模。

在Logistic回归模型中，关键是要求出参数w。

Sigmoid函数



0  
定义域 (-inf, inf) ; 值域为  
(0, 1)



## 第六章：概率建模

### □ Logistic回归

#### 最大似然函数：

对于N次观测，似然函数可以写为

$$\prod_{i=1}^N P(Y = y_i | X = x_i) = \prod_{i=1}^N p(x_i; \mathbf{w})^{y_i} (1 - p(x_i; \mathbf{w}))^{1-y_i}$$

注：N次观测的独立性。 $\mathbf{w}$ 是模型参数。

我们的目标是能够求出使这一似然函数取最大值时的参数估计 $\hat{\mathbf{w}}$ 。

$$\hat{\mathbf{w}} = \max_{\mathbf{w}} \prod_{i=1}^N P(Y = y_i | X = x_i) = \prod_{i=1}^N p(x_i; \mathbf{w})^{y_i} (1 - p(x_i; \mathbf{w}))^{1-y_i}$$

接下来，如何解这个模型是关键。



## 第六章：概率建模

### □ Logistic回归

#### 最大似然函数：

对似然函数取自然对数，可得出对数似然函数 $L(\mathbf{w})$ ，

$$\begin{aligned}L(\mathbf{w}) &= \sum_{i=1}^N [y_i \log p(x_i; \mathbf{w}) + (1 - y_i) \log(1 - p(x_i; \mathbf{w}))] \\&= \sum_{i=1}^N \left[ \log(1 - p(x_i; \mathbf{w})) + y_i \log \frac{p(x_i; \mathbf{w})}{1-p(x_i; \mathbf{w})} \right] \\&= \sum_{i=1}^N \left[ \log \left(1 - \frac{1}{1+e^{-h(x)}}\right) + y_i h(x) \right] \\&= \sum_{i=1}^N \left[ -\log(1 + e^{h(x)}) + y_i h(x) \right] \\&= \sum_{i=1}^N \left[ -\log \left(1 + e^{w_0 + \mathbf{w}^T \mathbf{x}_i}\right) \right] + \sum_{i=1}^N y_i (w_0 + \mathbf{w}^T \mathbf{x}_i)\end{aligned}$$

为了估计能使 $L(\mathbf{w})$ 取得最大值的参数 $\mathbf{w} = [w_0, w_1, \dots, w_d]$ ，对函数求导。



## 第六章：概率建模

### □ Logistic回归

#### 最大似然函数：

化简后的对数似然函数 $L(w)$ ,

$$L(w) = \sum_{i=1}^N \left[ -\log(1 + e^{w_0 + w^T x_i}) \right] + \sum_{i=1}^N y_i (w_0 + w^T x_i)$$

为了估计能使 $L(w)$ 取得最大值的参数 $w=[w_0, w_1, \dots, w_d]$ , 分别求函数对 $w_0, w_1, \dots, w_d$ 的导数, 显然存在 $d+1$ 个方程。

$$\begin{aligned}\frac{\partial L(w)}{\partial w_j} &= -\sum_{i=1}^N \frac{1}{1 + e^{w_0 + w^T x_i}} \cdot e^{w_0 + w^T x_i} \cdot x_{ij} + \sum_{i=1}^N y_i x_{ij} \\ &= \sum_{i=1}^N (y_i - p(x_i; w)) x_{ij}\end{aligned}$$

注：此时导数表达式是一个**超越方程**，这里不能另该导数为0，不存在闭合解。**怎么办？**



## 第六章：概率建模

### □ Logistic回归

#### Newton-Raphson (牛顿-拉斐森) 优化算法：

我们考虑一个单变量函数的情况，即 $f(\beta)$ ,求该函数的最优解 $\beta^*$ 。我们假设函数 $f(\cdot)$ 是平滑的，那么意味着该函数在 $\beta^*$ 处的导数为0，二阶导数大于0（极值理论）。

那么根据泰勒展开， $f(\beta)$ 的表达式可以写成

$$f(\beta) \approx f(\beta^*) + \frac{1}{2}(\beta - \beta^*)^2 \frac{d^2 f}{d\beta^2} |_{\beta=\beta^*}$$

既然 $\beta^*$ 是最小值点，那么 $f(\beta) > f(\beta^*)$ ,因此， $\frac{d^2 f}{d\beta^2} |_{\beta=\beta^*} > 0$ 。

而且，根据上式， $\frac{df}{d\beta} |_{\beta=\beta^*} = 0$

换句话说，在最优解附近， $f(\beta)$ 是一个近二次的函数表达式。



## 第六章：概率建模

### □ Logistic回归

#### Newton-Raphson (牛顿-拉斐森) 优化算法：

既然在最优解附近， $f(\beta)$ 是一个近二次的函数表达式，而我们的目的是求出这个最优解，因此，可以将 $f(\beta)$ 进行二次泰勒展开（为什么不更高次？？）

首先，猜测一个初始最优点 $\beta^{(0)}$ ，在该点，二次泰勒展开：

$$f(\beta) \approx f(\beta^{(0)}) + (\beta - \beta^{(0)}) \frac{df}{d\beta} \Big|_{\beta=\beta^{(0)}} + \frac{1}{2} (\beta - \beta^{(0)})^2 \frac{d^2 f}{d\beta^2} \Big|_{\beta=\beta^{(0)}}$$

为了方便，上式可以重新写成，

$$f(\beta) \approx f(\beta^{(0)}) + (\beta - \beta^{(0)}) f'(\beta^{(0)}) + \frac{1}{2} (\beta - \beta^{(0)})^2 f''(\beta^{(0)})$$

此时， $f(\beta)$ 是关于 $\beta$ 的二次函数，是可导的，因此便可以利用导数=0求解。



## 第六章：概率建模

### □ Logistic回归

Newton-Raphson (牛顿-拉斐森) 优化算法：

$$f(\beta) \approx f(\beta^{(0)}) + (\beta - \beta^{(0)})f'(\beta^{(0)}) + \frac{1}{2}(\beta - \beta^{(0)})^2 f''(\beta^{(0)})$$

若要求函数 $f(\beta)$ 的最小值，我们令

$$\frac{df(\beta)}{d\beta} = f'(\beta^{(0)}) + f''(\beta^{(0)})(\beta - \beta^{(0)}) = 0$$

从而可以计算出

$$\beta = \beta^{(0)} - \frac{f'(\beta^{(0)})}{f''(\beta^{(0)})}$$

令此时的最优解为 $\beta = \beta^{(1)}$  (并非最优)

$$\beta^{(1)} = \beta^{(0)} - \frac{f'(\beta^{(0)})}{f''(\beta^{(0)})}$$



## 第六章：概率建模

### □ Logistic回归

**Newton-Raphson** (牛顿-拉斐森) 优化算法：

当计算出新的次优解 $\beta^{(1)}$ ， 函数 $f(\beta)$ 的泰勒展开为

$$f(\beta) \approx f(\beta^{(1)}) + (\beta - \beta^{(1)})f'(\beta^{(1)}) + \frac{1}{2}(\beta - \beta^{(1)})^2 f''(\beta^{(1)})$$

若要求函数 $f(\beta)$ 的最小值，我们令

$$\frac{df(\beta)}{d\beta} = f'(\beta^{(1)}) + f''(\beta^{(1)})(\beta - \beta^{(1)}) = 0$$

从而可以计算出

$$\beta = \beta^{(1)} - \frac{f'(\beta^{(1)})}{f''(\beta^{(1)})}$$

令此时的最优解为 $\beta = \beta^{(2)}$  (并非最优)

$$\beta^{(2)} = \beta^{(1)} - \frac{f'(\beta^{(1)})}{f''(\beta^{(1)})}$$



## 第六章：概率建模

### □ Logistic回归

Newton-Raphson (牛顿-拉斐森) 优化算法：

为了找到最优解，通常迭代多次，设为T次。

当计算出新的次优解 $\beta^{(t)}$ ，函数 $f(\beta)$ 的泰勒展开为

$$f(\beta) \approx f(\beta^{(t)}) + (\beta - \beta^{(t)})f'(\beta^{(t)}) + \frac{1}{2}(\beta - \beta^{(t)})^2 f''(\beta^{(t)})$$

若要求函数 $f(\beta)$ 的最小值，我们令

$$\frac{df(\beta)}{d\beta} = f'(\beta^{(t)}) + f''(\beta^{(t)})(\beta - \beta^{(t)}) = 0$$

从而可以计算出

$$\beta = \beta^{(t)} - \frac{f'(\beta^{(t)})}{f''(\beta^{(t)})}$$

令此时的最优解为 $\beta = \beta^{(t+1)}$ （并非最优）

解更新的迭代公式

$$\beta^{(t+1)} = \beta^{(t)} - \frac{f'(\beta^{(t)})}{f''(\beta^{(t)})}$$



## 第六章：概率建模

### □ Logistic回归

Newton-Raphson (牛顿-拉斐森) 优化算法 (多个变量的情况——多维) :

一维时，解更新的迭代公式

$$\beta^{(t+1)} = \beta^{(t)} - \frac{f'(\beta^{(t)})}{f''(\beta^{(t)})}$$

对于多维时，模型由  $f(\beta) = \beta_0 + \beta_1 x$  转化为  $f(\beta) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$

那么如何计算多维的函数关于变量的一阶导数和二阶导数？？



## 第六章：概率建模

### □ Logistic回归

Newton-Raphson (牛顿-拉斐森) 优化算法 (多个变量的情况——多维) :

对于多维时，模型由  $f(\beta) = \beta_0 + \beta_1 x$  转化为  $f(\beta) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$   
回顾第三章 (数学基础) 中，关于对矩阵或向量的导数。

$$f'(\beta^{(t)}) = \nabla f(\beta^{(t)}) = \left[ \frac{\partial f}{\partial \beta_1}, \frac{\partial f}{\partial \beta_2}, \dots, \frac{\partial f}{\partial \beta_d} \right]^T$$

$$f''(\beta^{(t)}) = \nabla^2 f(\beta^{(t)}) = \begin{bmatrix} \frac{\partial^2 f}{\partial \beta_1^2} & \cdots & \frac{\partial f}{\partial \beta_1 \partial \beta_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial \beta_d \partial \beta_1} & \cdots & \frac{\partial^2 f}{\partial \beta_d^2} \end{bmatrix} |_{\beta=\beta^{(t)}} = H \text{ (Hessian矩阵)}$$



## 第六章：概率建模

### □ Logistic回归

**Newton-Raphson** (牛顿-拉斐森) 优化算法 (多个变量的情况——多维) :

对于多维时，函数 $f(\beta)$ 的解更新优化方法为：

$$\beta^{(t+1)} = \beta^{(t)} - \frac{f'(\beta^{(t)})}{f''(\beta^{(t)})}$$



$$\beta^{(t+1)} = \beta^{(t)} - H^{-1}(\beta^{(t)}) \nabla f(\beta^{(t)})$$

迭代方法完全相同，只是在写法上，变成了矩阵向量的方式。



## 第六章：概率建模

### □ Logistic回归

#### 最大似然函数：

化简后的对数似然函数 $L(w)$ ,

$$L(w) = \sum_{i=1}^N \left[ -\log(1 + e^{w_0 + w^T x_i}) \right] + \sum_{i=1}^N y_i (w_0 + w^T x_i)$$

为了估计能使 $L(w)$ 取得最大值的参数 $w=[w_0, w_1, \dots, w_d]$ , 分别求函数对 $w_0, w_1, \dots, w_d$ 的导数, 显然存在 $d+1$ 个方程。

$$\begin{aligned}\frac{\partial L(w)}{\partial w_j} &= - \sum_{i=1}^N \frac{1}{1 + e^{w_0 + w^T x_i}} \cdot e^{w_0 + w^T x_i} \cdot x_{ij} + \sum_{i=1}^N y_i x_{ij} \\ &= \sum_{i=1}^N (y_i - p(x_i; w)) x_{ij}\end{aligned}$$

注：此时导数表达式是一个**超越方程**，这里不能另该导数为0，不存在闭合解。**怎么办？**



## 第六章：概率建模

### □ Logistic回归

最大似然函数：

Hessian矩阵H的计算：

$$\frac{\partial^2 L(\mathbf{w})}{\partial w_j^2} = - \sum_{i=1}^N x_{ij}^2 \cdot p(x_i; \mathbf{w}) \cdot (1 - p(x_i; \mathbf{w}))$$

$$\frac{\partial^2 L(\mathbf{w})}{\partial w_j \partial w_l} = - \sum_{i=1}^N x_{ij} x_{il} \cdot p(x_i; \mathbf{w}) \cdot (1 - p(x_i; \mathbf{w}))$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - H^{-1}(\mathbf{w}^{(t)}) \nabla f(\mathbf{w}^{(t)})$$

迭代公式



## 第六章：概率建模

### □ 牛顿法存在的问题

- ✓ 经典牛顿法虽然具有二次收敛性，但是要求初始点需要尽量靠近极小点，否则有可能不收敛。
- ✓ 计算过程中需要计算目标函数的二阶偏导数，难度较大。
- ✓ 目标函数的Hessian矩阵无法保持正定，会导致算法产生的方向不能保证是 $f$ 在 $x_k$ 处的下降方向，从而令牛顿法失效；
- ✓ 如果Hessian矩阵奇异，牛顿方向可能根本是不存在的。

### □ 补充：正定矩阵的定义、性质、判定方法

**定义：**对于n阶方阵A，如果对任何非零向量z，都有 $z^T A z > 0$ ，就称A为正定矩阵。

**性质：**正定矩阵一定是非奇异的、可逆的矩阵。

**判定：**对称阵A为正定的充分必要条件为A的特征值均大于0。



## 第六章：概率建模

### □ 本章小结

- Logistic回归在本质上是线性回归。
- Logistic回归解决的是因变量的离散型问题，即分类问题（二类问题）。
- Logistic回归适合于当因变量属于二项分布这类问题。
- Logistic回归模型的输出是介于0-1之间的数，是一个概率值，基于sigmoid函数。
- 该模型的求解采用最大似然估计和牛顿法。
- 建模过程不需要对属性变量进行高斯分布的假设，但依然需要条件独立假设。



## 课外任务：

根据前面的内容，请自行推导Logistic回归建模方法的整个过程。



# 第七章：朴素贝叶斯学习



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

➤ **Thomas Bayes**的“逆概”问题。

**正向概率：**假设袋子里面有 $N$ 个白球， $M$ 个黑球，你伸手进去摸一把，摸出黑球的概率是多大？

**逆向概率：**如果我们事先并不知道袋子里面黑白球的比例，而是闭着眼睛摸出一个（或好几个）球，观察这些取出来的球的颜色之后，那么我们可以就此对袋子里面的黑白球的比例作出什么样的推测？

解决这个问题要靠什么？

一个字“猜”！



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

- 贝叶斯学习是一种基于概率的学习方法，能够计算显式的假设概率。它基于假设的先验概率，给定假设下观察到不同数据的概率，以及观察到数据本身的概率。
- 1.  $P(h)$ : 表示没有训练样本数据前，假设 $h$ 拥有的初始概率，称为假设 $h$ 的先验概率。如果，我们不清楚这个先验知识，在实际中通常简单地将每一种假设都赋予相同的概率。
- 2.  $P(D)$ : 表示观察到训练数据 $D$ 的先验概率，通常作为常数处理，因为不同的假设拥有相同的 $P(D)$ 。
- 3.  $P(D|h)$ : 表示假设 $h$ 成立时观察到数据 $D$ 的概率。
- 4.  $P(h|D)$ : 后验概率，即对于给定的一个训练样本 $D$ ，假设 $h$ 成立的概率。在机器学习中，我们感兴趣的是这个后验概率。



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

贝叶斯公式：

$$P(h|D) = \frac{P(h) * P(D|h)}{P(D)}$$

这个公式是怎么来的？

例：一所学校里面有 60% 的男生， 40% 的女生。男生总是穿长裤，女生则一半穿长裤一半穿裙子。有了这些信息之后我们可以容易地计算“随机选取一个学生，他（她）穿长裤的概率和穿裙子的概率是多大”，这个就是前面说的“正向概率”的计算。然而，假设你走在校园中，迎面走来一个穿长裤的学生，你能够推断出他（她）是男生的概率是多大吗？

分析：如果知道有多少个穿长裤的学生，并知道穿长裤的人里面有多少男生就可以了。

设学校里面共有N人，其中：

- ◆ 穿长裤的男生的数量是  $N * P(\text{男}) * P(\text{裤}|\text{男})$
- ◆ 穿长裤的女生的数量是  $N * P(\text{女}) * P(\text{裤}|\text{女})$

那么，  $P(\text{男}|\text{裤}) = N * P(\text{男}) * P(\text{裤}|\text{男}) / [N * P(\text{男}) * P(\text{裤}|\text{男}) + N * P(\text{女}) * P(\text{裤}|\text{女})] = P(\text{男}) * P(\text{裤}|\text{男}) / P(\text{裤})$

通式：  $P(B|A) = P(B) * P(A|B) / P(A)$

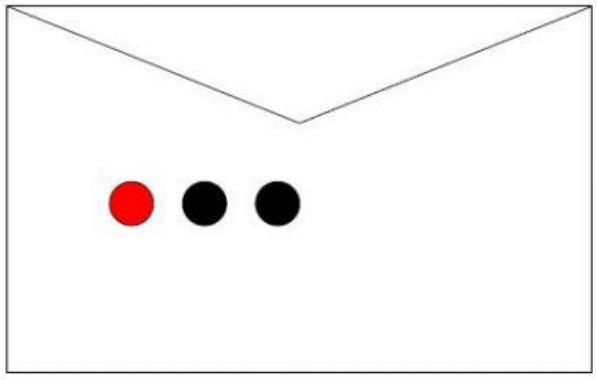
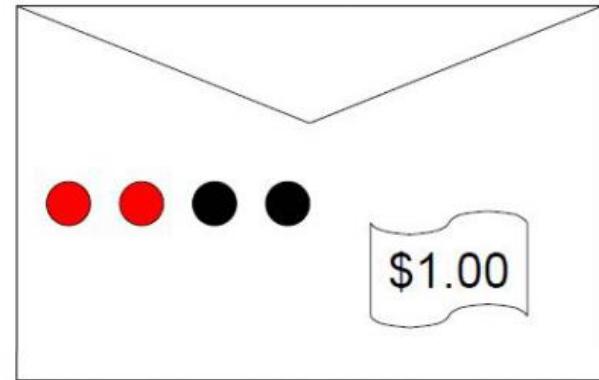


## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

贝叶斯公式：

$$P(h|D) = \frac{P(h) * P(D|h)}{P(D)}$$



- 如果摸到一个红球，那么，这个信封有1美元的概率是？
- 如果摸到一个黑球，那么，这个信封有1美元的概率是？

逆概问题



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

- ◆ c1、c2表示左右两个信封。
- ◆  $P(R)$ ,  $P(B)$ 表示摸到红球、黑球的概率。
- ◆  $P(R)=P(R|c1)*P(c1) + P(R|c2)*P(c2)$ : 全概率公式
- ◆ 后验:  $P(c1|R)=P(R|c1)*P(c1)/P(R)$
- ◆  $P(R|c1)=2/4$
- ◆  $P(R|c2)=1/3$
- ◆  $P(c1)=P(c2)=1/2$

如果摸到一个红球，那么，这个信封有1美元的概率是

$$P(c1|R)=P(R|c1)*P(c1)/P(R)=0.6$$

如果摸到一个黑球，那么，这个信封有1美元的概率是

$$P(c2|B)=P(B|c2)*P(c2)/P(B)=3/7$$



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

贝叶斯公式：

$$P(h|D) = \frac{P(h) * P(D|h)}{P(D)}$$

贝叶斯学习的目的是对于训练样本D，找出一个**最靠谱**的猜测h(假设)，使得后验概率达到最大。

也就是计算下列模型的最优解。

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D)$$

其中H是各种“猜测”的集合。 $h_{MAP}$ 是最佳的“最靠谱”的猜测，满足后验最大。

将上面的模型写完整：

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} \frac{P(h) * P(D|h)}{P(D)} = \operatorname{argmax}_{h \in H} P(h) * P(D|h)$$

注： $P(D)$ 是不依赖于 $h$ 的常量。



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

**最大后验假设：**在许多学习场景中，学习器考虑候选假设集合 $H$ ，并在其中寻找给定数据 $D$ 时，**可能性**最大的假设 $h \in H$ ,即为最大后验假设（Maximum a Posteriori, MAP），也称为MAP假设，定义为 $h_{MAP}$ 。

确定MAP假设 $h_{MAP}$ 的方法，是利用贝叶斯公式计算集合 $H$ 中每个候选假设的后验概率。即

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} \frac{P(h) * P(D|h)}{P(D)}$$



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

贝叶斯模型：

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h) * P(D|h)$$

- ◆ 当设计者不清楚先验时，可将所有假设的先验 $P(h)$ 均设为相同的值，即 $P(h) = \frac{1}{H}$ ，此时，模型变成了 $h_{MAP} = \operatorname{argmax}_{h \in H} P(D|h)$ ， $P(D|h)$ 常被称为给定 $h$ 时，数据 $D$ 的似然，而使 $P(D|h)$ 最大的假设，称为极大似然假设。
- ◆ 可以看出，贝叶斯模型实际上是先验与数据似然函数的乘积。如何理解先验？先验很多时候，能够提供有力的“人为”帮助，提升模型的分类能力和鲁棒。

模型的抽象含义是：对于给定观测数据 $D$ ，一个猜测 $h$ 是好是坏，取决于“这个猜测本身的可能性大小（先验概率 $P(h)$ ， Prior）”和“这个猜测生成我们观测到的数据的可能性大小”（似然 $P(D|h)$ ， Likelihood）的乘积



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习

#### 先验的理解

例：用户输入一个错误单词**Thet**，请判断用户实际真正想输入的单词：“**They**” or “**That**”？

分析：假设“**They**”的后验概率 $P(\text{They}|\text{Thet})=P(\text{Thet}|\text{They})*P(\text{They})/P(\text{Thet})$ ；

假设“**That**”的后验概率 $P(\text{That}|\text{Thet})=P(\text{Thet}|\text{That})*P(\text{That})/P(\text{Thet})$ ；

其中， $P(\text{They})$ 和 $P(\text{That})$ 为假设的先验概率。

1. 若根据实际使用频率，单词**That**的使用频率要高于**They**，也就是说，用户实际想输入**That**的可能性更高，即  $P(\text{They}) < P(\text{That})$
2. 若根据键盘中字母的排列，字母t和字母y是紧密靠近的，因此，用户实际想输入**they**的可能性更高，即 $P(\text{They}) > P(\text{That})$

根据不同的先验知识，先验概率的值不同，因此先验知识的形式是多样的。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯学习

朴素贝叶斯到底“朴素”在哪？也就是朴素贝叶斯的假设是，当类假设 $h$ 给定时，特征之间相互条件独立。

换句话说，在给定假设 $h$ 的情况下，观察到联合的 $d_1, d_2, \dots, d_n$ 的概率等于单独特征（属性）的概率乘积：

$$P(d_1, d_2, \dots, d_n | h) = \prod_i P(d_i | h)$$

代入到贝叶斯模型中，有

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h | D) = \operatorname{argmax}_{h \in H} \frac{P(h) * \prod_i P(d_i | h)}{P(D)} = \operatorname{argmax}_{h \in H} P(h) * \prod_i P(d_i | h)$$

“朴素”即“特征独立假设”的好处就是大大降低了估计带来计算复杂度。



# 第七章：朴素贝叶斯学习

## 口 朴素贝叶斯学习

目标值PlayTennis的14个训练样例

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

问题：结合朴素贝叶斯方法，对下面的新实例（4个特征属性）进行分类：

**Outlook=Sunny**

**Temperature=Cool**

**Humidity=High**

**Wind=Strong**

**PlayTennis=Yes or No?**



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯学习

$$h_{MAP} = \operatorname{argmax}_{h \in H=\{\text{Yes}, \text{No}\}} P(h) * \prod_i P(d_i|h)$$

分析：结合朴素贝叶斯分类器模型，分别把 $h=\text{Yes}$ 和 $h=\text{No}$ 两种假设下的后验概率计算出来，最大值对应的假设就是我们的答案。

◆ 当 $h=\text{Yes}$ 时，

$$\begin{aligned} P(h) * \prod_i P(d_i|h) &= P(\text{Yes}) * P(\text{Sunny}|\text{Yes}) * P(\text{Cool}|\text{Yes}) * P(\text{High}|\text{Yes}) * p(\text{Strong}|\text{Yes}) \\ &= \frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{3}{9} = 0.0053 \end{aligned}$$

◆ 当 $h=\text{No}$ 时，

$$\begin{aligned} P(h) * \prod_i P(d_i|h) &= P(\text{No}) * P(\text{Sunny}|\text{No}) * P(\text{Cool}|\text{No}) * P(\text{High}|\text{No}) * p(\text{Strong}|\text{No}) \\ &= \frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{3}{5} = 0.0206 \end{aligned}$$

显然， $P(\text{No}) * \prod_i P(d_i|\text{No}) = 0.0206 > P(\text{Yes}) * \prod_i P(d_i|\text{Yes}) = 0.0053$ ，所以， $\text{PlayTennis}=\text{No}$ , 今天的天气不适合打网球



## 第七章：朴素贝叶斯学习

### □ 贝叶斯学习方法的特性

1. 贝叶斯方法允许假设做出不确定性的预测（如：某一癌症病人有85%的机会康复）。
2. 先验知识可以与观察数据一起决定假设的最终概率。在贝叶斯学习中，先验知识的形式可以是：1) 每个候选假设的先验概率；2) 每个候选假设在观察数据上的概率分布。
3. 新的样本分类可以由多个假设一起作出预测，用它们的概率来加权（最优贝叶斯）。



## 第七章：朴素贝叶斯学习

### □ 采用朴素贝叶斯对垃圾邮件进行分类

1. 样本：1000封邮件，每个邮件被标记为垃圾邮件或者正常邮件。设有400封垃圾邮件，600封正常邮件。
2. 分类目标（问题）：给定第1001封邮件，确定它是垃圾邮件还是非垃圾邮件？
3. 方法：朴素贝叶斯。

该如何建模和实现？

(一) 分析：令待分类的这封邮件为 $D=[d_1, d_2, \dots, d_n]$ ，由n个单词组成。垃圾邮件定义为 $h1$ ，正常邮件定义为 $h2$ 。设400封垃圾邮件中的单词总数为 $N1$ ，600封正常邮件中的单词总数为 $N2$ 。



## 第七章：朴素贝叶斯学习

### □ 采用朴素贝叶斯对垃圾邮件进行分类

1. 样本：1000封邮件，每个邮件被标记为垃圾邮件或者正常邮件。设有400封垃圾邮件，600封正常邮件。
2. 分类目标（问题）：给定第1001封邮件，确定它是垃圾邮件还是非垃圾邮件？
3. 方法：朴素贝叶斯。

该如何建模和实现？

（二）如果能够计算出垃圾邮件 $h_1$ 假设的后验概率和正常邮件 $h_2$ 假设的后验概率，两者概率最大的假设，即为该封邮件的类别。采用贝叶斯公式。

$$\text{数学描述: } P(h_1|D) = \frac{P(D|h_1)P(h_1)}{P(D)}, P(h_2|D) = \frac{P(D|h_2)P(h_2)}{P(D)}$$

$P(D)$ 为与 $h$ 无关的常量，因此，只需计算 $P(D|h_1)P(h_1)$ 和 $P(D|h_2)P(h_2)$ 即可。



## 第七章：朴素贝叶斯学习

### □ 采用朴素贝叶斯对垃圾邮件进行分类

1. 样本：1000封邮件，每个邮件被标记为垃圾邮件或者正常邮件。设有400封垃圾邮件，600封正常邮件。
2. 分类目标（问题）：给定第1001封邮件，确定它是垃圾邮件还是非垃圾邮件？
3. 方法：朴素贝叶斯。

该如何建模和实现？

(三) 为了计算 $P(D|h_1)P(h_1)$ 和 $P(D|h_2)P(h_2)$ ，可以先计算先验概率 $P(h_1)$  和 $P(h_2)$ 。

$$P(h_1) = \frac{400}{1000} = 0.4; \quad P(h_2) = \frac{600}{1000} = 0.6$$

那如何计算 $P(D|h_1)$ 和 $P(D|h_2)$ ？

$$P(D|h_1) = P(d_1, d_2, \dots, d_n | h_1)$$

然后呢？问题变得复杂。由于数据的稀疏性以及语言的多样性，完全出现两封完全相同的邮件（文本）是几乎不可能的。也就是说， $n$ 个单词 $(d_1, d_2, \dots, d_n)$ 在所有邮件中同时出现的概率几乎为0。那怎么办？



## 第七章：朴素贝叶斯学习

### □ 采用朴素贝叶斯对垃圾邮件进行分类

1. 样本：1000封邮件，每个邮件被标记为垃圾邮件或者正常邮件。设有400封垃圾邮件，600封正常邮件。
2. 分类目标（问题）：给定第1001封邮件，确定它是垃圾邮件还是非垃圾邮件？
3. 方法：朴素贝叶斯。

该如何建模和实现？

（四）当给定某假设时，采用特征的条件独立性假设，即朴素贝叶斯。

那如何计算 $P(D|h_1)$ 和 $P(D|h_2)$ ？

$$\begin{aligned} P(D|h_1) &= P(d_1, d_2, \dots, d_n|h_1) = P(d_1|h_1) * P(d_2|h_1) * \dots * P(d_n|h_1) \\ &= \prod_{i=1}^n P(d_i|h_1) \end{aligned}$$

现在问题变得非常简单，计算 $P(d_i|h_1)$ 的概率，只需要统计单词 $d_i$ 在400封垃圾邮件中出现的频次，然后相乘。同理， $P(D|h_2)$ 的概率也是这么计算。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

基本定义：

1. 设 $\mathbf{x} = \{x_1, x_2, x_3, \dots, x_d\}$ 为一个待分类的样本，由 $d$ 个特征属性构成；
2. 有类别集合 $C = \{y_1, y_2, \dots, y_C\}$ ；
3. 计算 $P(y_1|\mathbf{x}), P(y_2|\mathbf{x}), \dots, P(y_C|\mathbf{x})$ ；
4. 如果 $P(y_k|\mathbf{x}) = \max\{P(y_1|\mathbf{x}), P(y_2|\mathbf{x}), \dots, P(y_C|\mathbf{x})\}$ , 则 $\mathbf{x}$ 属于 $y_k$ 这个类别。

在以上4个步骤中，只要完成第3步，即可实现样本 $\mathbf{x}$ 的分类。

如何实现第3步？

利用前面所学习的朴素贝叶斯学习算法！



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

方法的基本定义：

1. 设 $\mathbf{x} = \{x_1, x_2, x_3, \dots, x_d\}$ 为一个待分类的样本，由 $d$ 个特征属性构成；
2. 有类别集合 $C = \{y_1, y_2, \dots, y_C\}$ ；
3. 计算 $P(y_1|\mathbf{x}), P(y_2|\mathbf{x}), \dots, P(y_C|\mathbf{x})$ ；
4. 如果 $P(y_k|\mathbf{x}) = \max\{P(y_1|\mathbf{x}), P(y_2|\mathbf{x}), \dots, P(y_C|\mathbf{x})\}$ , 则 $\mathbf{x}$ 属于 $y_k$ 这个类别。

在以上4个步骤中，只要成功完成第3步，即可实现样本 $\mathbf{x}$ 的分类。

如何实现第3步？

利用前面所学习的朴素贝叶斯学习算法！



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

第3步计算 $P(y_1|\mathbf{x}), P(y_2|\mathbf{x}), \dots, P(y_C|\mathbf{x})$ 条件概率的具体步骤：

1. 已知类别的训练样本集（如14天的天气数据，以及是否打了网球）
2. 统计每个类别下，各个特征属性的条件概率（出现频次），即

$$P(x_1|y_1), P(x_2|y_1), \dots, P(x_d|y_1);$$

$$P(x_1|y_2), P(x_2|y_2), \dots, P(x_d|y_2);$$

⋮

$$P(x_1|y_C), P(x_2|y_C), \dots, P(x_d|y_C);$$

3. 计算各类别的先验概率 $P(y_1), P(y_2), \dots, P(y_C)$ （统计各类别出现的频率）
4. 根据朴素贝叶斯方法，各个特征属性是条件独立的。利用贝叶斯定理(公式)，

$$\text{我们要计算的 } P(y_i|\mathbf{x}) = \frac{P(\mathbf{x}|y_i)P(y_i)}{P(\mathbf{x})} = \frac{P(x_1|y_i) \cdot P(x_2|y_i) \cdots P(x_d|y_i) \cdot P(y_i)}{P(\mathbf{x})} = \frac{P(y_i) \cdot \prod_{j=1}^d P(x_j|y_i)}{P(\mathbf{x})},$$

其中， $P(\mathbf{x})$ 是数据的先验，可以当做常数处理，不影响后验概率的比较。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

前面的例子中，考虑了特征值属性为离散值时的情况，在实际问题中，也经常会遇到特征值属性为连续值的情况。

那该如何计算完成上述条件概率的计算？即

$$P(x_1|y_i), P(x_2|y_i), \dots, P(x_d|y_i), \forall i = 1, \dots, C$$

➤ 当特征值为连续值时，通常假设特征属性服从高斯分布，即

$$P(\mathbf{x}|y_i) = N(\mu_i, \sigma_i)$$

$$\text{因此, } P(x_j|y_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_j - \mu_i)^2}{2\sigma_i^2}}$$

➤ 如何计算每一类各特征属性的均值以及方差？即 $\mu_i$ 和 $\sigma_i$ ( $i = 1, \dots, C$ )直接利用期望和方差公式即可。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

#### ✓ Laplacian校验(零概率问题)

当特征属性为离散值时，计算条件概率 $P(x_j|y_i)$ 即在类 $y_i$ 条件下统计属性 $x_j$ 出现的次数，可能会遇到出现次数为0，从而导致该条件概率为0，会影响分类器的性能（训练样本较少时会遇到该问题）。

这种情况下，可以利用Laplacian校验，即统计每一类下的某特征属性出现的次数时，全部计数加1，可避免0概率的出现。

当训练样本数量充分大时，不影响分类器的结果。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

#### ➤ 分类器性能评价

在机器学习中，基本上分类器的评价是通过分类器正确率(识别率)进行衡量。

分类器正确率：指被正确分类的样本数占所有样本数的比率。

分类正确率的计算通常是基于一部分新的测试样本集，而非训练样本集。

因为训练集往往会因为过拟合，而得出较高的正确率，但不能作为分类器的评价指标。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

#### ➤ 最小错误率贝叶斯分类器

利用后验概率 $P(y_i|\mathbf{x})$ ，当 $P(y_k|\mathbf{x})$ 为所有后验概率中最大值时，待分类样本 $\mathbf{x}$ 属于 $y_k$ 类；即 $P(y_k|\mathbf{x}) = \max_{i=1,\dots,c} P(y_i|\mathbf{x})$

#### ➤ 最大似然比贝叶斯分类器

对于两类问题，当  $P(\mathbf{x}|y_i)P(y_i) > P(\mathbf{x}|y_j)P(y_j)$  时，判决 $\mathbf{x}$ 属于 $y_i$ 类；也就是当  $\frac{P(\mathbf{x}|y_i)}{P(\mathbf{x}|y_j)} > \frac{P(y_j)}{P(y_i)}$  时，判决 $\mathbf{x}$ 属于 $y_i$ 类。其中， $\frac{P(\mathbf{x}|y_i)}{P(\mathbf{x}|y_j)}$  为似然比。 $\frac{P(y_j)}{P(y_i)}$  为判决门限。

#### ➤ 最小风险贝叶斯分类器



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

#### ➤ 最小风险贝叶斯分类器

顾名思义，对于待分类样本，有多种决策，具有最小风险的决策（分类）即为最终的分类。作如下定义：

1. 决策 $\omega_i$ : 把待识别样本 $\mathbf{x}$ 分类到 $y_i$ 类中；
2. 损失 $\rho_{ij}$ : 把真实属于 $y_i$ 类的样本 $\mathbf{x}$ 错误分类到 $y_j$ 类中；
3. 条件风险 $R(\omega_i|\mathbf{x})$ : 对待识别样本 $\mathbf{x}$ 采取决策 $\omega_i$ ，产生的可能风险；条件风险的计算公式如下： $R(\omega_i|\mathbf{x}) = \sum_{j=1}^c \rho_{ij} P(y_j|\mathbf{x})$
4. 最小风险贝叶斯分类器就是计算具有最小风险的决策 $\omega_k$

$$R(\omega_k|\mathbf{x}) = \min_{i=1,\dots,c} R(\omega_i|\mathbf{x})$$

那么待识别样本 $\mathbf{x}$ 采取决策 $\omega_k$ ，也就是将被分类到 $y_k$ 类中。



## 第七章：朴素贝叶斯学习

### □ 朴素贝叶斯分类器

#### ➤ 最小风险贝叶斯分类器

当  $\rho_{ij} = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases}$  时，

有

$$\min_{i=1, \dots, c} R(\omega_i | \mathbf{x}) = \min_{i=1, \dots, c} \sum_{j=1}^c \rho_{ij} P(y_j | \mathbf{x}) = \min_{i=1, \dots, c} \sum_{j=1, j \neq i}^c P(y_j | \mathbf{x})$$

$$= \min_{i=1, \dots, c} 1 - P(y_i | \mathbf{x}) \quad \text{注: } \sum_{j=1}^c P(y_j | \mathbf{x}) = 1$$

$$= \max_{i=1, \dots, c} P(y_i | \mathbf{x}) \quad \text{注: 最小错误率贝叶斯分类器}$$

可见，最小错误率分类器是最小风险贝叶斯分类器的一种特例。



# 第八章：分类器

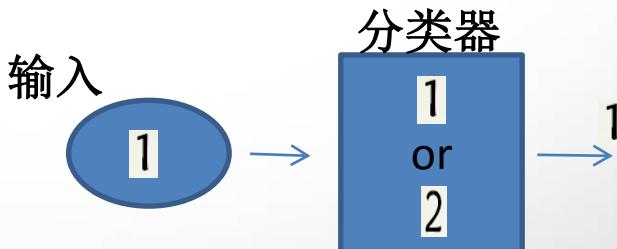


## 第八章：分类器

### □ 定义

- 对于一组训练样本集 $x_1, x_2, \dots, x_N$ , 每个样本由D个属性组成, 形成D-维矢量。其中, 每个样本分别采用一个标签 $y_1, y_2, \dots, y_N$ 描述所属类别, 样本集由C个类别组成。对于给定的新样本 $x_{new}$ , 判定其所属的类别, 被称为**分类**。
- 利用训练集生成的分类模型/方法/算法, 被称为**分类器(Classifier)**。

分类问题是机器学习的核心问题, 也是人工智能、模式识别、计算机视觉、图像理解、文本分类、数据挖掘等应用领域要解决的关键问题。





## 第八章：分类器

### □ 分类器种类

机器学习发展至今，衍生出大量的分类器模型，本课程主要介绍以下几类分类器：

#### ➤ 概率分类器（参见前面的章节）

贝叶斯分类器

朴素贝叶斯分类器

Logistic回归

#### ➤ 非概率分类器

K-近邻分类器（欧式距离）

大间隔分类器（支持向量机）

线性判别分类器（投影）

神经网络分类器（多层感知器、BP网络、深度卷积网络—第九章讲）



## 第八章：分类器

不同于概率分类器，对于每个样本 $x$ ，计算 $P(y=c|x)$ ，提供类别 $y=c$ 的可能性，非概率分类器输出的是一个指定的类别，即 $y(x)=c$ 。

### □ K-近邻分类器（K-nearest neighbors, KNN）

K-近邻分类器是基于欧式距离提出的较为直观的分类器方法，思想极其简单，不需要对训练集进行训练生成模型，完全无参数。

假设有N个训练样本，每个样本分别由属性x和标记y进行描述。对于一个新样本 $x_{new}$ ，如何利用KNN进行分类？

Step 1: 分别计算 $x_{new}$ 与训练集中 $x_1, x_2, \dots, x_N$ 之间的距离；

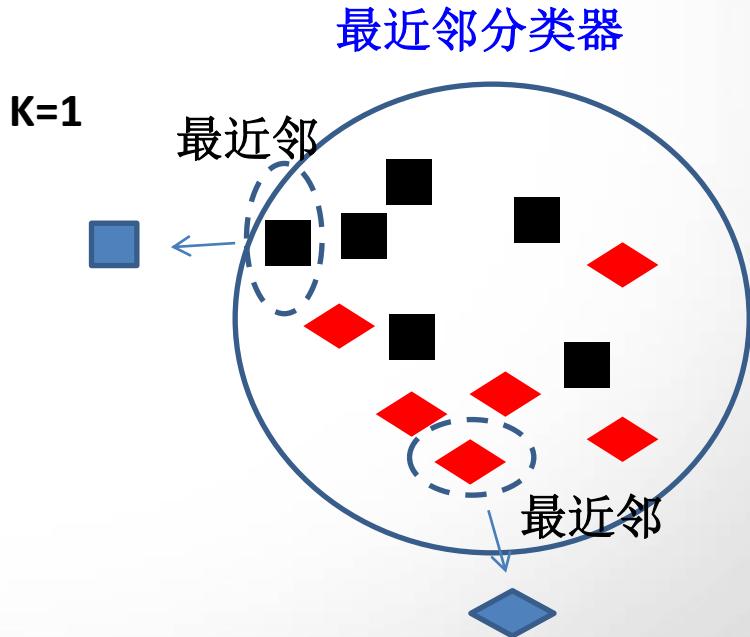
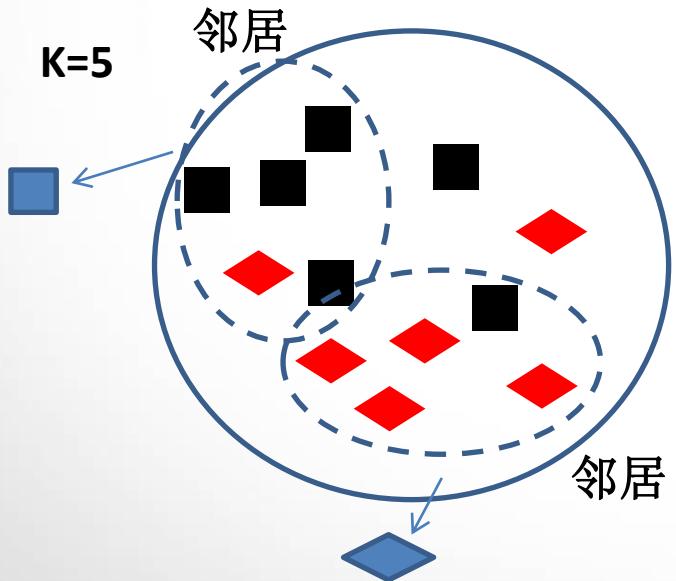
Step 2: 找出与 $x_{new}$ 最近的K个样本 $x_i$ ；

Step 3: 根据“服从多数”原则， $x_{new}$ 的类别即为K个样本中最多的类。

## 第八章：分类器

### □ K-近邻分类器（K-nearest neighbors, KNN）

KNN分类器的思想用一个图进行描述





## 第八章：分类器

### □ K-近邻分类器

#### 距离计算

样本 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 之间的欧式距离表达式

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{l=1}^d (x_i^l - x_j^l)^2}$$

#### K值的选择

- K值通常是一个奇数，即选取奇数个邻居点，从而可以通过投票确定类别；
- 如果K太小，分类器容易被噪声干扰；比如K=1时，最近邻点如果被噪声污染，则分类错误。
- 如果K太大，可能会永远无法进行正确的分类（各类样本不均衡问题），比如两类问题，第I类有10个样本，第II类有30个样本，如果K>21，则新样本将会一直被识别为第II类。



## 第八章：分类器

### □ 其他距离

距离通常用于度量两个样本之间的相似性，在分类器中经常用到。

- ✓ 马氏距离（协方差）
- ✓ 曼哈顿距离（城市间街道）
- ✓ 切比雪夫距离 ( $\max_l |x_i^l - x_j^l|$ )
- ✓ 余弦距离（向量夹角  $\cos\theta = \mathbf{x}_i \cdot \mathbf{x}_j / |\mathbf{x}_i| |\mathbf{x}_j|$ ）
- ✓ 汉明距离（信息编码）
- ✓ 相关性（统计协方差  $\rho_{\mathbf{x}_i, \mathbf{x}_j}$ ）

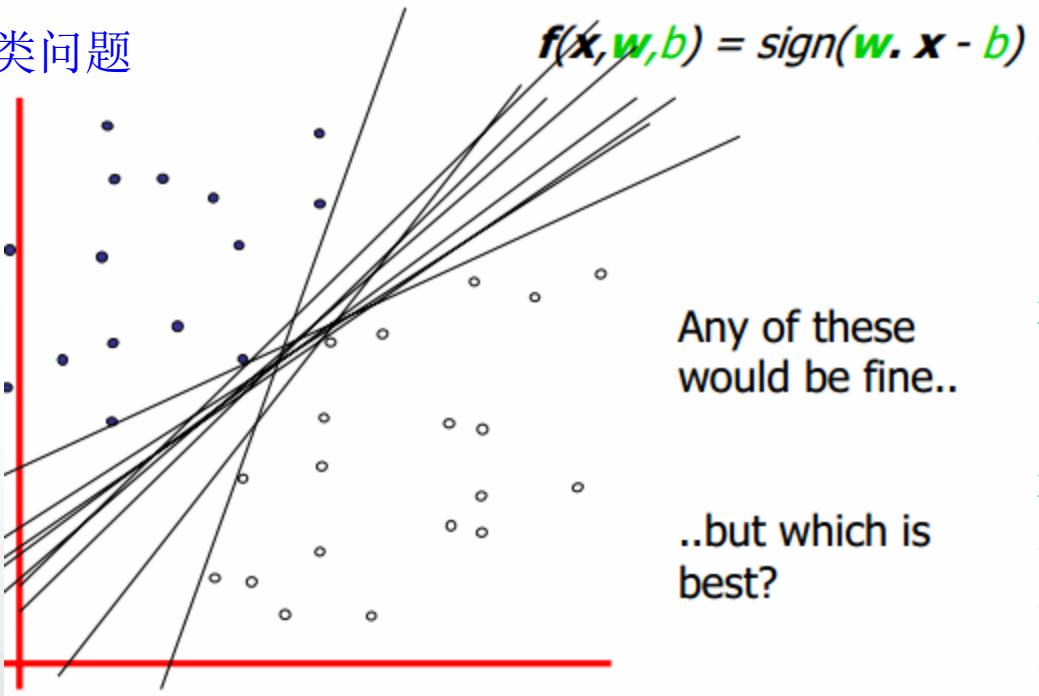
### □ 度量学习

度量学习不同于上述“已知”的距离表达式，而是通过机器学习建模的方式，学习一个最佳的距离度量，使得对特定的问题，具有最佳的识别性能。

## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM, Vapnik'95）

考虑线性分类问题



假设给定一个特征空间上的训练数据集  $T=\{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$ , 其中,  
 $x_i \in \mathbb{R}^n$ ,  $y_i \in \{+1, -1\}$ ,  
 $i=1, 2, \dots, N$ 。 $x_i$  为第  $i$  个特征向量, 也称为实例,  $y_i$  为  $x_i$  的类标记; 当  $y_i=+1$  时, 称  $x_i$  为正例; 当  $y_i=-1$  时, 称  $x_i$  为负例。 $(x_i, y_i)$  称为样本点。



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

给定线性可分训练数据集，通过间隔最大化或等价的求解相应的**凸二次规划问题**学习得到的分离超平面为

$$\mathbf{w}\mathbf{x} + b = 0,$$

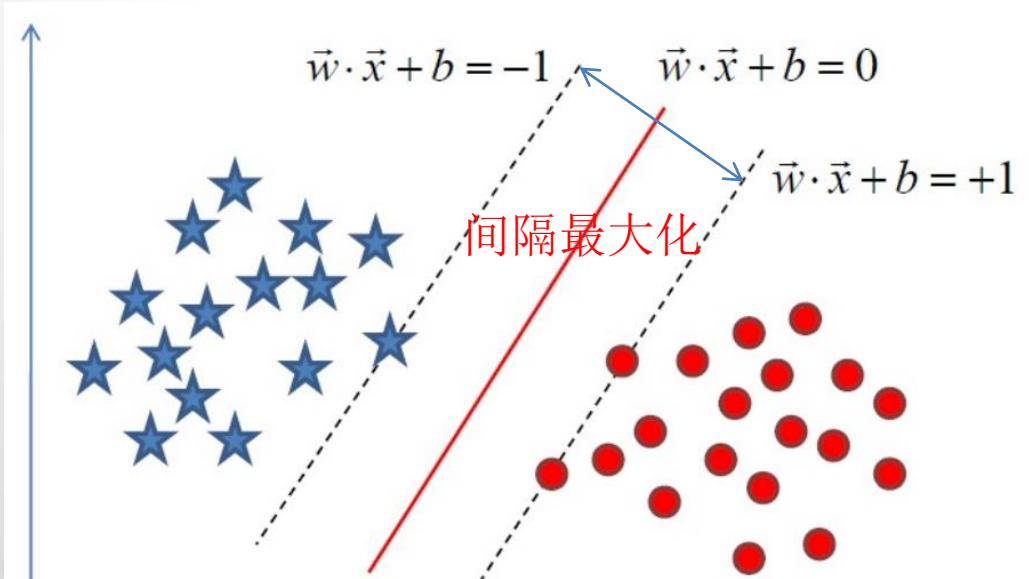
相应的分类决策函数

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}\mathbf{x} + b)$$

该决策函数称为线性可分支持向量机。

## □ 支持向量机（Support Vector Machine, SVM）

### ➤ 线性可分支持向量机



$$x = x_0 + \gamma \frac{w}{\|w\|}$$



定义：函数间隔和几何间隔

**函数间隔：**给定的训练数据集T和超平面( $w, b$ )，定义超平面( $w, b$ )关于样本点( $x_i, y_i$ )的函数间隔为：

$$\hat{\gamma} = y(\mathbf{w}^T \mathbf{x} + b) = yf(\mathbf{x})$$

**几何间隔：**平面法向单位化的函数间隔

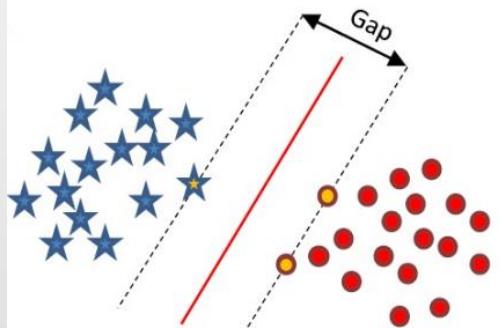
$$\gamma = \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} = \frac{f(\mathbf{x})}{\|\mathbf{w}\|} = \frac{\hat{\gamma}}{\|\omega\|}$$

# 第八章：分类器

## □ 支持向量机（Support Vector Machine, SVM）

### ➤ 线性可分支持向量机

最大间隔分类超平面



$$\max_{w,b} \gamma$$

$$s.t. \quad y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|} \right) \geq \gamma, \quad i = 1, 2, \dots, N$$

$$\max_{w,b} \frac{\hat{\gamma}}{\|w\|}$$

$$s.t. \quad y_i (w \cdot x_i + b) \geq \hat{\gamma}, \quad i = 1, 2, \dots, N$$

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$s.t. \quad y_i (w \cdot x_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

$$\max_{w,b} \frac{1}{\|w\|}$$

$$s.t. \quad y_i (w \cdot x_i + b) \geq 1, \quad i = 1, 2, \dots, N$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

线性可分支持向量机模型

$$\begin{aligned} & \min_{w,b} \frac{1}{2} \|w\|^2 \\ s.t. \quad & y_i(w \cdot x_i + b) \geq 1, \quad i = 1, 2, \dots, N \end{aligned}$$

带有约束的最优化问题。如何求解？



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

拉格朗日乘子法

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1)$$

原问题的对偶问题： 极大极小问题

$$\max_{\alpha} \min_{w,b} L(w,b,\alpha)$$

将拉格朗日函数 $L(w,b,\alpha)$ 分别对 $w$ ,  $b$ 求偏导并令其为0:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

将上式带入拉格朗日函数 $L(w,b,\alpha)$ 中，得到

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i (w^T x_i + b) - 1] \\ &= \frac{1}{2} w^T w - w^T \sum_{i=1}^n \alpha_i y_i x_i - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i \\ &= \frac{1}{2} w^T \sum_{i=1}^n \alpha_i y_i x_i - w^T \sum_{i=1}^n \alpha_i y_i x_i - b \cdot 0 + \sum_{i=1}^n \alpha_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \left( \sum_{i=1}^n \alpha_i y_i x_i \right)^T \sum_{i=1}^n \alpha_i y_i x_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

$$\max_{\alpha} \min_{w,b} L(w,b,\alpha)$$



$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \quad \alpha_i \geq 0, i = 1, 2, \dots, n$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

$$\min_{\alpha} \quad \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i$$

$$s.t. \quad \sum_{i=1}^N \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

1. 模型：

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i$$

$$s.t. \sum_{i=1}^N \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N$$

#### 2. 求得最优解 $\alpha^*$

3. 计算

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^* = y_i - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j)$$

#### 4. 分离超平面

$$w^* x + b^* = 0$$

#### 5. 分类决策函数

$$f(x) = sign(w^* x + b^*)$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

举例：给定3个数据点：正样本点 $x_1=(3,3)^T$ ,  $x_2=(4,3)^T$ , 负样本点 $x_3=(1,1)^T$ , 利用线性可分支持向量机，求线性分类决策函数？

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ = & \frac{1}{2} (18\alpha_1^2 + 25\alpha_2^2 + 2\alpha_3^2 + 42\alpha_1\alpha_2 - 12\alpha_1\alpha_3 - 14\alpha_2\alpha_3) - \alpha_1 - \alpha_2 - \alpha_3 \\ s.t. \quad & \alpha_1 + \alpha_2 - \alpha_3 = 0 \\ & \alpha_i \geq 0, \quad i = 1, 2, 3 \end{aligned}$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

举例：给定3个数据点：正样本点 $x_1=(3,3)^T$ ,  $x_2=(4,3)^T$ , 负样本点 $x_3=(1,1)^T$ , 利用线性可分支持向量机，求线性分类超平面？

- ✓ 将  $\alpha_1 + \alpha_2 = \alpha_3$  带入目标函数，得到关于 $\alpha_1, \alpha_2$ 的函数：

$$s(\alpha_1, \alpha_2) = 4\alpha_1^2 + \frac{13}{2}\alpha_2^2 + 10\alpha_1\alpha_2 - 2\alpha_1 - 2\alpha_2$$

- ✓ 对 $\alpha_1, \alpha_2$ 求偏导并令其为0，易知 $s(\alpha_1, \alpha_2)$ 在点 $(1.5, -1)$ 处取极值。
- ✓ 然而，该点不满足条件 $\alpha_2 \geq 0$ 。
  - 当 $\alpha_1=0$ 时，最小值 $s(0, 2/13) = -2/13$
  - 当 $\alpha_2=0$ 时，最小值 $s(1/4, 0) = -1/4$
- ✓ 于是， $s(\alpha_1, \alpha_2)$ 在 $\alpha_1=1/4$ ,  $\alpha_2=0$ 时达到最小，此时， $\alpha_3 = \alpha_1 + \alpha_2 = 1/4$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性可分支持向量机

举例：给定3个数据点：正样本点 $x_1=(3,3)^T$ ,  $x_2=(4,3)^T$ , 负样本点 $x_3=(1,1)^T$ , 利用线性可分支持向量机，求线性分类超平面？

- ✓ 可见， $\alpha_1=\alpha_3=1/4$ 对应的点 $x_1, x_3$ 是支持向量。
- ✓ 带入公式，求 $w$ 和 $b$ :

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^* = y_i - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j)$$

✓ 分离决策函数为  $f(x) = \text{sign}\left(\frac{1}{2}x_1 + \frac{1}{2}x_2 - 2\right)$

得到 $w_1=w_2=0.5$ ,  $b=-2$

- ✓ 因此，分离超平面为  $\frac{1}{2}x_1 + \frac{1}{2}x_2 - 2 = 0$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性支持向量机

若数据线性不可分，则增加松弛因子 $\xi_i \geq 0$ ，使函数间隔加上松弛变量大于等于1。这样，约束条件变成

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

SVM的模型变成

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i && \boxed{\text{惩罚项}} \\ & s.t. \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, N \\ & \quad \xi_i \geq 0, \quad i = 1, 2, \dots, N \end{aligned}$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性支持向量机

✓ 拉格朗日乘子方程

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i (w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^N \mu_i \xi_i$$

✓ 对  $w, b, \xi$  求偏导

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

$$C - \alpha_i - \mu_i = 0$$

✓ 将三式带入 L 中，得到

$$\min_{w, b, \xi} L(w, b, \xi, \alpha, \mu) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i$$

✓ 对上式求关于  $\alpha$  的极大，得到

$$\max_{\alpha} -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i$$

$$s.t. \sum_{i=1}^N \alpha_i y_i = 0$$

$$C - \alpha_i - \mu_i = 0$$

$$\alpha_i \geq 0 \quad 0 \leq \alpha_i \leq C$$

$$\mu_i \geq 0, \quad i = 1, 2, \dots, N$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

#### ➤ 线性支持向量机

##### 1. 构造SVM模型

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i$$

$$s.t. \sum_{i=1}^N \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, N$$

##### 2. 求解 $\alpha^*$

SMO(Sequential Minimal Optimization)

算法求解, 参见文献

John C. Platt, Sequential Minimal Optimization:

A Fast Algorithm for Training Support Vector machines

##### 3. 计算w和b

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^* = y_i - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_i)$$

##### 4. 求得分离超平面

$$w^* x + b^* = 0$$

##### 5. 分类决策函数

$$f(x) = \text{sign}(w^* x + b^*)$$

$$= \text{sign}\left(\sum_{i=1}^N \alpha_i^* y_i x_i x + b^*\right)$$



## 第八章：分类器

### □ 支持向量机（Support Vector Machine, SVM）

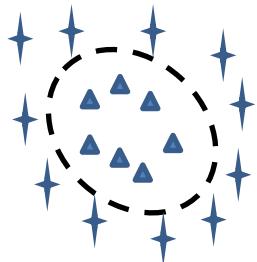
#### ➤ 非线性支持向量机（核函数）

**目的：**将线性不可分的数据(信号)，通过一个非线性函数 $\varphi(\cdot)$ ，映射到高维特征空间，使得线性可分。

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i$$

$$s.t. \sum_{i=1}^N \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, N$$



$$\varphi(x_i) \cdot \varphi(x_j)$$

$$k(x_i, x_j)$$

$$e^{-\sigma \|x_i - x_j\|^2}$$



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 概念

- ✓ 子空间学习是指对于一给定的分类任务，如何为原数据寻找一个低维子空间，使得在该低维子空间内实现更好的分类，而计算复杂度降低；
- ✓ 子空间学习的方法，一般通过投影的方式，实现高维特征（原数据空间）向低维子空间的映射。这个过程也被称为“降维”；

#### ➤ 种类

- ✓ 线性子空间学习
- ✓ 非线性子空间学习

通常情况下，非线性子空间学习方法是通过“核函数”进行建模实现。整个过程，是在线性子空间学习的方法上进行扩展。



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 线性子空间学习

定义原始高维特征集为 $\mathbf{X}$ ，子空间学习从数学上是为了学习一个投影（或变换 $\mathbf{W}$ ），使得特征集 $\mathbf{X}$ 在低维子空间的表达形式为

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X}$$

线性子空间学习的目的是为了获得变换矩阵 $\mathbf{W}$ ，使得低维子空间表达 $\mathbf{Y}$ 能更好的实现分类。

#### ➤ 种类

- ✓ 主成分分析(PCA)；
- ✓ Fisher判别分析（线性判别分析，LDA）；
- ✓ 流形学习（局部保持投影, LPP）；



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 主成分分析(PCA)

主成分分析是一种无监督的降维方法，在整个过程中，不需要知道样本集 $\mathbf{X}$ 的标签。数学上，PCA是一种均方误差最小意义下的投影方法。

定义高维特征集 $\mathbf{X}$ ，存在一个投影（或变换 $\mathbf{W}$ ），即特征集 $\mathbf{X}$ 在低维子空间的表达形式为

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X}$$

使得 $\mathbf{Y}$ 保留高维特征集 $\mathbf{X}$ 中的大部分有效信息（能量），即满足：

$$\max_{\mathbf{W}} \|\mathbf{Y} - \mu\|_F^2 = \max_{\mathbf{W}} \|\mathbf{W}^T \mathbf{X} - \mathbf{W}^T \mathbf{m}\|_F^2$$

$$= \max_{\mathbf{W}} \text{Tr}(\mathbf{W}^T (\mathbf{X} - \mathbf{m})(\mathbf{X} - \mathbf{m})^T) \mathbf{W}$$

$$= \max_{\mathbf{W}} \text{Tr}(\mathbf{W}^T \Sigma \mathbf{W}), s.t. \mathbf{W}^T \mathbf{W} = \mathbf{I} \quad \Sigma = (\mathbf{X} - \mathbf{m})(\mathbf{X} - \mathbf{m})^T \text{ 为协方差矩阵}$$

W的求解采用特征值分解



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ Fisher判别分析(LDA)

与PCA不同，LDA是一种有监督的降维方法，需要利用样本集的标签信息，从而使获得的低维子空间表达，能够更好地实现分类任务。

两者区别在于，PCA是为了找到一种原始数据的最佳表达，而LDA是为了找到一种有利于分类的最佳表达。

比如：在字符识别中，区分字母O和字母Q。PCA方法会保留两个字母最相似的部分即O，而抛弃字母Q的“尾巴”，然而对于LDA则会尽力保住这个“尾巴”，因为更易于区分。

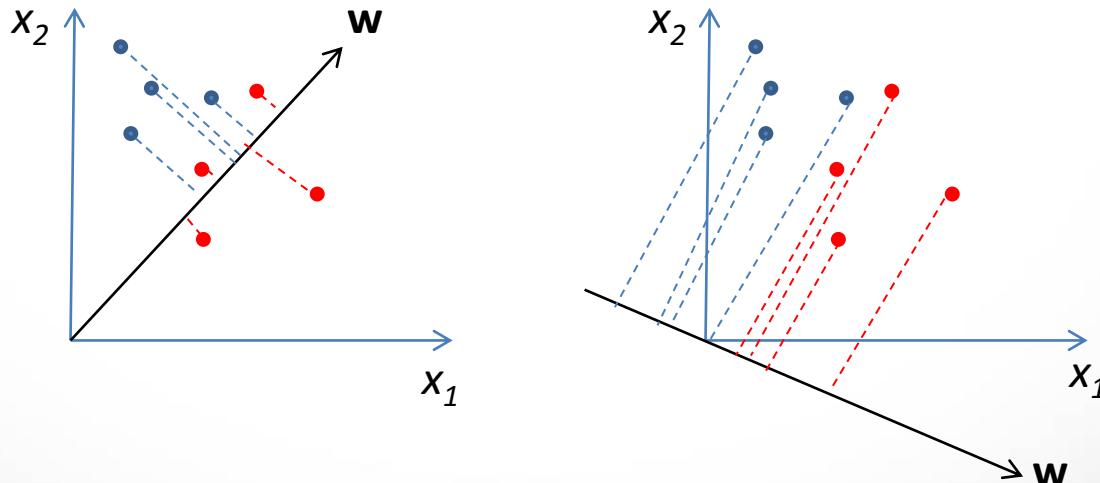
定义高维特征集 $\mathbf{X}$ ，存在一个投影（或变换 $\mathbf{W}$ ），即特征集 $\mathbf{X}$ 在低维子空间的表达形式为

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X}$$

使得 $\mathbf{Y}$ 更加可分类。

## 第八章：分类器

### □ 子空间学习（Subspace Learning） ➤ Fisher判别分析(LDA)



不同方向的投影，投影后的数据点可分能力不同；对于给定的一组特征集，我们是有可能找到能够最大限度的区分各类数据点的投影方向。



## 第八章：分类器

### □ 子空间学习 (Subspace Learning)

#### ➤ Fisher判别分析(LDA)

- ✓ 定义高维特征集  $X = \{x_1, x_2, \dots, x_n\}$ ，其中  $n_1$  个样本属于第一类  $\omega_1$ ，剩余的  $n_2$  个样本属于第二类  $\omega_2$ 。那么存在一个投影（或变换  $W$ ），即特征集  $X$  在低维子空间的表达形式为

$$Y = W^T X$$

使得  $Y$  更加可分类。

那么该如何求  $W$  呢？

- ✓ LDA建模过程：

思想：我们希望经过  $W$  映射后，两类数据的中心间的距离越远越好（异类分离），即类间散度  $J_1$ ；同时，相同类的样本越往中心靠拢越好（同类紧凑），即类内散度  $J_2$ 。

Fisher准则：满足  $\frac{J_1}{J_2}$  最大化！



## 第八章：分类器

### □ 子空间学习 (Subspace Learning)

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X}$$

➤ Fisher判别分析(LDA)

✓ LDA建模过程：

思想：我们希望经过 $\mathbf{W}$ 映射后，两类数据的中心间的距离越远越好（异类分离），即类间散度J1；同时，相同类的样本越往中心靠拢越好（同类紧凑），即类内散度J2。

Fisher准则：满足 $\frac{J_1}{J_2}$ 最大化！

#### 1. 投影后，类间散度J1的构造

$$\text{第1类均值 (中心)} : \boldsymbol{\mu}_1 = \frac{1}{n_1} \sum_{y \in \omega_1} \mathbf{y} = \frac{1}{n_1} \sum_{x \in \omega_1} \mathbf{W}^T \mathbf{x} = \mathbf{W}^T \frac{1}{n_1} \sum_{x \in \omega_1} \mathbf{x} = \mathbf{W}^T \mathbf{m}_1$$

$$\text{第2类均值 (中心)} : \boldsymbol{\mu}_2 = \frac{1}{n_2} \sum_{y \in \omega_2} \mathbf{y} = \frac{1}{n_2} \sum_{x \in \omega_2} \mathbf{W}^T \mathbf{x} = \mathbf{W}^T \frac{1}{n_2} \sum_{x \in \omega_2} \mathbf{x} = \mathbf{W}^T \mathbf{m}_2$$

$$\text{中心间的距离: } J_1 = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2 = \|\mathbf{W}^T \mathbf{m}_1 - \mathbf{W}^T \mathbf{m}_2\|^2$$



## 第八章：分类器

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X}$$

### □ 子空间学习 (Subspace Learning)

➤ Fisher判别分析(LDA)

✓ LDA建模过程:

## 2. 投影后，类内散度J2的构造

$$\text{第1类均值 (中心)} : \boldsymbol{\mu}_1 = \frac{1}{n_1} \sum_{y \in \omega_1} \mathbf{y} = \frac{1}{n_1} \sum_{x \in \omega_1} \mathbf{W}^T \mathbf{x} = \mathbf{W}^T \frac{1}{n_1} \sum_{x \in \omega_1} \mathbf{x} = \mathbf{W}^T \mathbf{m}_1$$

$$\text{第2类均值 (中心)} : \boldsymbol{\mu}_2 = \frac{1}{n_2} \sum_{y \in \omega_2} \mathbf{y} = \frac{1}{n_2} \sum_{x \in \omega_2} \mathbf{W}^T \mathbf{x} = \mathbf{W}^T \frac{1}{n_2} \sum_{x \in \omega_2} \mathbf{x} = \mathbf{W}^T \mathbf{m}_2$$

$$\text{第1类类样本点与其中心间的距离: } J2\_1 = \sum_{y \in \omega_1} \|y - \boldsymbol{\mu}_1\|^2 = \sum_{x \in \omega_1} \|\mathbf{W}^T \mathbf{x} - \mathbf{W}^T \mathbf{m}_1\|^2$$

$$\text{第2类类样本点与其中心间的距离: } J2\_2 = \sum_{y \in \omega_2} \|y - \boldsymbol{\mu}_2\|^2 = \sum_{x \in \omega_2} \|\mathbf{W}^T \mathbf{x} - \mathbf{W}^T \mathbf{m}_2\|^2$$

$$\text{因此, } J2 = J2\_1 + J2\_2 = \sum_{x \in \omega_1} \|\mathbf{W}^T \mathbf{x} - \mathbf{W}^T \mathbf{m}_1\|^2 + \sum_{x \in \omega_2} \|\mathbf{W}^T \mathbf{x} - \mathbf{W}^T \mathbf{m}_2\|^2$$



## 第八章：分类器

### □ 子空间学习 (Subspace Learning)

#### ➤ Fisher判别分析(LDA)

✓ LDA建模过程：

3. 利用Fisher准则（即满足 $\frac{J_1}{J_2}$ 最大化），构造LDA模型

$$\max_w \frac{J_1}{J_2} = \max_w \frac{\|W^T m_1 - W^T m_2\|^2}{\sum_{x \in \omega_1} \|W^T x - W^T m_1\|^2 + \sum_{x \in \omega_2} \|W^T x - W^T m_2\|^2}$$

模型化简：

$$\max_w \frac{\text{Tr}(W^T(m_1 - m_2)(m_1 - m_2)^T W)}{\text{Tr}(W^T(\sum_{x \in \omega_1}(x - m_1)(x - m_1)^T + \sum_{x \in \omega_2}(x - m_2)(x - m_2)^T)W)}$$



$$\max_w \frac{\text{Tr}(W^T S_B W)}{\text{Tr}(W^T S_W W)}$$

特征值分解法求该优化问题

其中， $S_B = (m_1 - m_2)(m_1 - m_2)^T$ 为类间散度矩阵 (between-class scatter)；

$S_W = \sum_{w_x \in \omega_1}(x - m_1)(x - m_1)^T + \sum_{x \in \omega_2}(x - m_2)(x - m_2)^T$ 为类内散度矩阵 (within-class scatter)



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 多重判别分析(MDA)

MDA实质上是LDA的多类推广：

A. 对于多类问题(假设C个类), J2 (类内散度) 的构造可以由LDA直接推广, 即

$$\begin{aligned} J2 &= J2_1 + J2_2 + \cdots + J2_c \\ &= \sum_{x \in \omega_1} \|W^T x - W^T m_1\|^2 + \cdots + \sum_{x \in \omega_c} \|W^T x - W^T m_c\|^2 \\ &= \sum_{c=1}^C \sum_{x \in \omega_c} \|W^T x - W^T m_c\|^2 = \text{Tr}(W^T S_W W) \end{aligned}$$

其中,  $S_W = \sum_{c=1}^C \sum_{x \in \omega_c} (x - m_c)(x - m_c)^T$  为类内散度矩阵 (within-class scatter)



## 第八章：分类器

### □ 子空间学习 (Subspace Learning)

➤ 多重判别分析(MDA)

MDA实质上是LDA的多类推广：

B. 对于多类问题(假设C个类), J1 (类间散度) 的构造:

$$\text{第1类均值 (中心)} : \mu_1 = \frac{1}{n_1} \sum_{y \in \omega_1} y = \frac{1}{n_1} \sum_{x \in \omega_1} W^T x = W^T \frac{1}{n_1} \sum_{x \in \omega_1} x = W^T m_1$$

$$\text{第2类均值 (中心)} : \mu_2 = \frac{1}{n_2} \sum_{y \in \omega_2} y = \frac{1}{n_2} \sum_{x \in \omega_2} W^T x = W^T \frac{1}{n_2} \sum_{x \in \omega_2} x = W^T m_2$$

.

.

$$\text{第C类均值 (中心)} : \mu_C = \frac{1}{n_C} \sum_{y \in \omega_C} y = \frac{1}{n_C} \sum_{x \in \omega_C} W^T x = W^T \frac{1}{n_C} \sum_{x \in \omega_C} x = W^T m_C$$

$$\text{总体均值 } \mu = \frac{1}{N} \sum y = \frac{1}{N} \sum W^T x = W^T \frac{1}{N} \sum x = W^T m$$



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 多重判别分析(MDA)

MDA实质上是LDA的多类推广：

B. 对于多类问题(假设C个类),  $J_1$  (类间散度) 的构造:

$$J_1 = \sum_{c=1}^C n_c \|\mu_c - \mu\|^2 = \sum_{c=1}^C n_c \|W^T m_c - W^T m\|^2 = \text{Tr}(W^T S_B W)$$

其中,  $S_B = \sum_{c=1}^C n_c (m_c - m)(m_c - m)^T$

利用Fisher准则, MDA模型如下:

与LDA相同, 特征值分解法求该优化问题

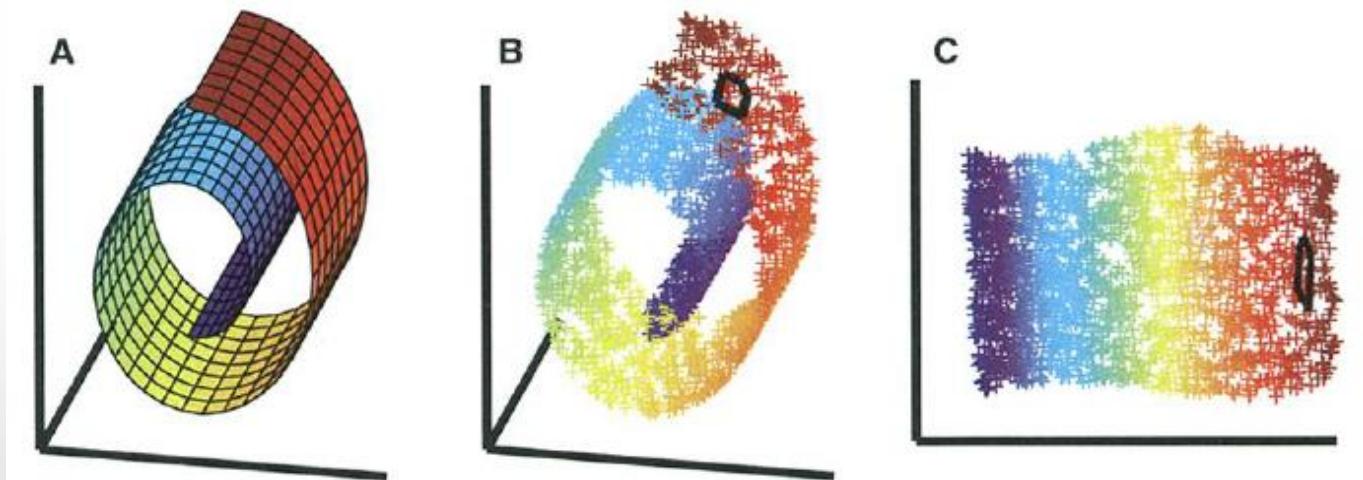
$$\max_W \frac{\text{Tr}(W^T S_B W)}{\text{Tr}(W^T S_W W)}$$

## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 局部保持投影(LPP)

LPP是一种流形思想下的子空间降维方法，前提假设了高维特征数据 $\mathbf{x}$ ，实际上是一种低维的流形结构 $\mathbf{Y}$ 嵌入在高维空间。流形学习的目的映射到低维中，揭示高维数据的本质（局部特性）。





## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 局部保持投影(LPP)

LPP是一种流形思想下的子空间降维方法，前提假设了高维特征数据 $\mathbf{x}$ ，实际上是一种低维的流形结构 $\mathbf{y}$ 嵌入在高维空间。流形学习的目的映射到低维中，揭示高维数据的本质（局部特性）。

局部保持特性是指，在高维空间相近的 $k$ 个样本点簇，经过 $\mathbf{w}$ 投影以后，这个 $k$ 个样本点簇依然保持相近！

数学模型为

$$\min_{\mathbf{W}} \sum_i \sum_{j, \forall \mathbf{x}_j \in N_k(\mathbf{x}_i)} A_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2$$

$$\text{其中, } A_{i,j} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in N_k(\mathbf{x}_i) \text{ and } \mathbf{x}_i \in N_k(\mathbf{x}_j) \\ 0, & \text{otherwise} \end{cases}$$



## 第八章：分类器

### □ 子空间学习（Subspace Learning）

#### ➤ 局部保持投影(LPP)

LPP是一种流形思想下的子空间降维方法，前提假设了高维特征数据 $\mathbf{x}$ ，实际上是一种低维的流形结构 $\mathbf{Y}$ 嵌入在高维空间。流形学习的目的映射到低维中，揭示高维数据的本质（局部特性）。

局部保持特性是指，在高维空间相近的 $k$ 个样本点簇，经过 $\mathbf{W}$ 投影以后，这个 $k$ 个样本点簇依然保持相近！

数学模型为

$$\begin{aligned} \min_{\mathbf{W}} \sum_i \sum_{j, \forall \mathbf{x}_j \in N_k(\mathbf{x}_i)} A_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 &= \min_{\mathbf{W}} \sum_i \sum_{j, \forall \mathbf{x}_j \in N_k(\mathbf{x}_i)} A_{i,j} \|\mathbf{W}^T \mathbf{x}_i - \mathbf{W}^T \mathbf{x}_j\|^2 \\ &= \min_{\mathbf{W}} \mathbf{W}^T \mathbf{L} \mathbf{W} \end{aligned}$$

其中， $\mathbf{L}$ 是拉普拉斯矩阵， $\mathbf{L} = \mathbf{D} - \mathbf{A}$ ， $\mathbf{D}$ 是一个对角阵， $D_{ii} = \sum_i A_{i,:}$



## 第八章：分类器

### □ 子空间学习与分类器

前面介绍了几种不同的子空间学习方法，其目的是在低维空间内实现分类任务，从而降低计算复杂度，提升分类效率和准确率。

**如何实现分类？**结合本章的前两节，讲述的**KNN**和**SVM**方法，可以实现分类，其实现方法基本形式为：

- A. PCA+KNN & PCA+SVM
- B. LDA+KNN & LDA+SVM
- C. MDA+KNN & MDA+SVM
- D. LPP+KNN & LPP+SVM

由于子空间模型的灵活性，可提出不同的子空间改进方案，实现高效的分类/识别任务。



# 第九章：集群智能仿生



## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

- 群智能作为一种新兴的演化计算技术，已成为研究焦点，它与生命进化、昆虫集体行为、鸟群集体行为，有着极为特殊的关系。
- 所谓群体智能，是需要本无智能的个体，通过合作表现出智能行为的特性，在没有集中控制，而且不提供全局模型的前提下，为寻找最优方案或最优解提供了基础。
- 人们把群居昆虫或鸟类的集体行为称作“群智能”或“群体智能”或“集群智能”。
- 主要特点在于：个体的行为很简单，但当它们一起协同工作时，却能够突显出复杂的智能的行为特征。



# 第九章：集群智能仿生

## □ 群体智能（Swarm Intelligence）

### ✓ 优点

灵活性：群体可以适应灵活变化的特性

稳健性：某个体失败，不影响整个群体的行为

自组织行为：不受外部因素的控制

### ✓ 典型算法

遗传算法（起源于生物进化过程，演化算法）：人类对大自然生物进化过程的计算模拟而抽象出的进化算法；

粒子群优化（启发于鸟群觅食）：基于社会系统中群体智能而产生的启发式仿生算法；

蚁群优化（启发于蚂蚁群体活动）

鱼群、蜂群算法、免疫算法、混沌搜索算法等





## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

- 遗传算法（GA, 起源于生物进化过程，演化算法）：人类对大自然生物进化过程的计算模拟而抽象出的进化算法。

**基本思想：**每一个物质个体的基本特征会被后代（子代）继承，但后代又会产生一些不同于父代的新特征，在进化过程中，最适应环境的个体特征会被保留（适者生存）。

**个体表现形式：**个体特征以基因的形式包含在染色体内，通过基因突变或基因杂交，可以产生更适合环境的后代（即更好的解）。

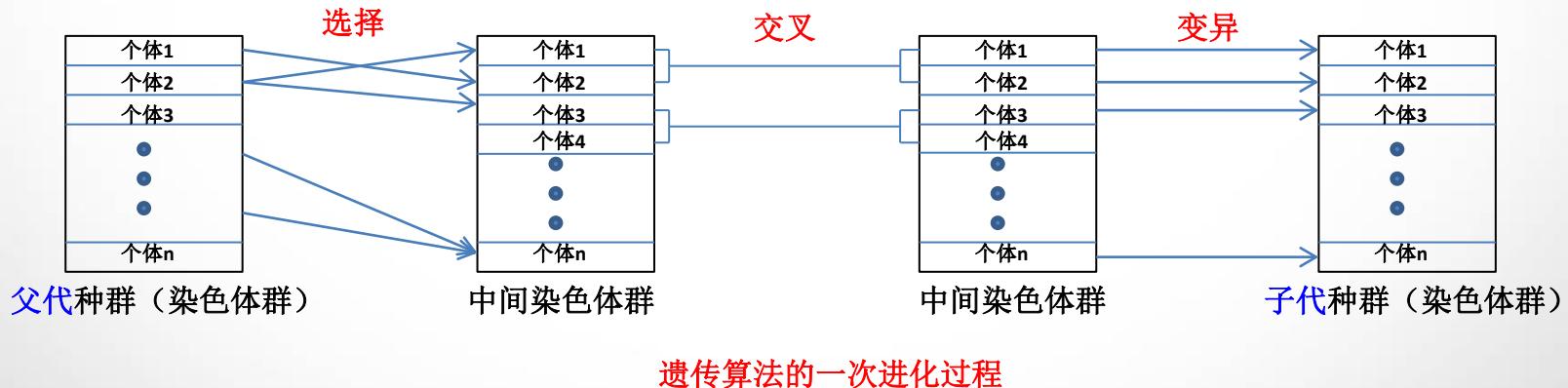
**实现：**基于种群的演化，这里种群是指一群染色体。通过设定某“适应度函数”，从中选择最适应环境的染色体，并进行复制、交叉、变异，产生新一代染色体种群，通过一代代进化，直到染色体不再发生变化，即可找到最优的染色体（最优解）。

# 第九章：集群智能仿生

## □ 群体智能（Swarm Intelligence）

### ➤ 遗传算法（GA）

遗传算法主要由三个算子组成，即选择算子、交叉算子和变异算子，用于产生新的一群染色体。



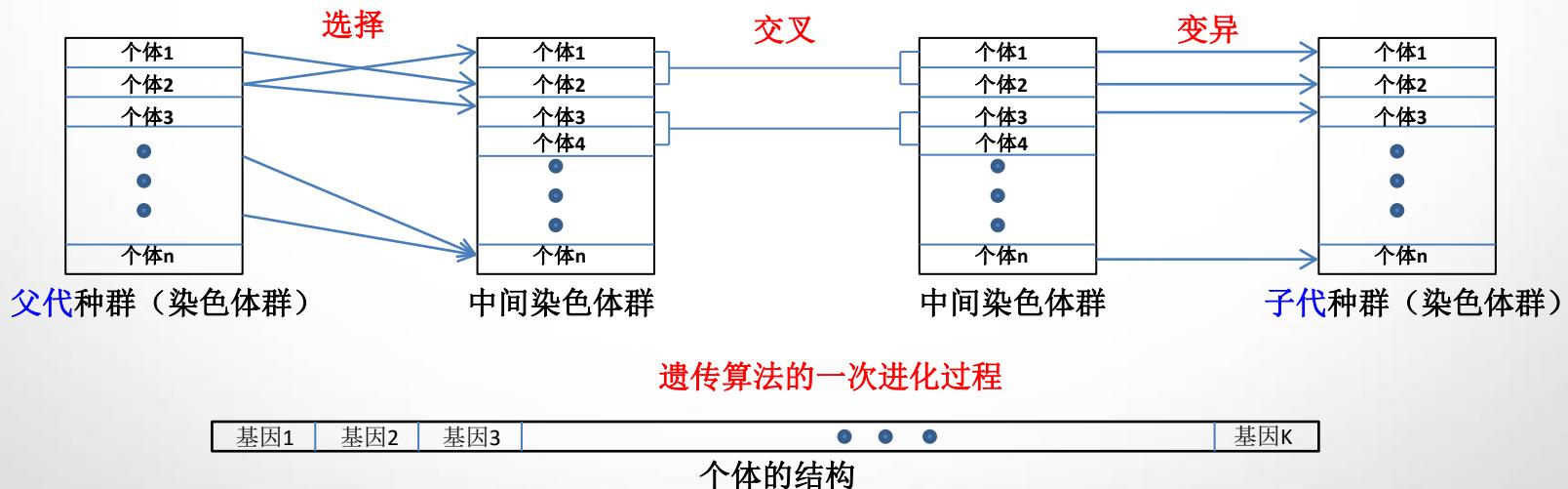
遗传算法如何多次进化呢？ 不断反馈、迭代！

# 第九章：集群智能仿生

## □ 群体智能（Swarm Intelligence）

### ➤ 遗传算法（GA）

遗传算法主要由三个算子组成，即选择算子、交叉算子和变异算子，用于产生新的一群染色体。





## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

#### ➤ 遗传算法（GA）

遗传算法主要由三个算子组成，即选择算子、交叉算子和变异算子，用于产生新的一群染色体。

遗传算法的解搜索过程：

1. **编码**：在演化前，先对问题的解空间表示成基因串结构数据（个体的编码）；
2. **生成初始种群**：随机产生 $n$ 个基因串结构数据，每个基因串为一个个体； $n$ 个个体组成一个群体，作为初始迭代点；（注：每个个体代表一个可行解）
3. **适应度评价**：根据适应度函数，计算每个个体的适应度，评估个体的优劣，即解的优劣；根据具体问题，定义能够客观反映染色体优劣的目标函数；
4. **选择**：从当前群体中选择优良的（适应度高的）个体，使它们有机会被选中，并进入到下一次迭代，舍弃适应度低的个体（适者生存之原则）；



## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

#### ➤ 遗传算法（GA）

遗传算法主要由三个算子组成，即选择算子、交叉算子和变异算子，用于产生新的一群染色体。

遗传算法的解搜索过程：

5. 交叉：遗传操作，信息的交换，交叉方法以某交叉概率Pc,通过第i个个体和第i+1个个体之间进行交叉，其中交叉算子为

$$\begin{cases} X_i^{t+1} = c_i \cdot X_i^t + (1 - c_i) \cdot X_{i+1}^t \\ X_{i+1}^{t+1} = (1 - c_i) \cdot X_i^t + c_i \cdot X_{i+1}^t \end{cases}$$

6. 变异：遗传操作，以变异概率Pm进行基因变异，变异算子为

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + c_i$$



## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

#### ➤ 遗传算法（GA）

##### 控制参数的选择：

(1) 基因串长度 $L$ （个体的长度）

个体的长度取决于特定问题，即需要的可行解的长度（精度）；

(2) 群体规模 $n$

群体规模越大，找到全局最优解的可能性也越大，改善搜索效果。但增加了个体评价的计算，收敛速度较慢；

(3) 交叉概率 $P_c$

交叉概率控制着交叉算子使用的频率，交叉概率越高，群体中个体更新速度较快，但会造成优良基因的丢失。但若交叉概率越小，可能导致搜索陷入局部最优解，而早熟（收敛停滞）

(4) 变异概率 $P_m$

变异是针对某个基因而发生的。如果变异概率较大，则基因变异的数量也较高，会导致随机搜索现象；若变异概率较小，会造成基因丢失而无法恢复。



## 第九章：集群智能仿生



### □ 群体智能（Swarm Intelligence）

#### ➤ 粒子群算法(PSO)

最早由Eberhart和Kennedy于1995年提出，通过模拟鸟群的觅食行为，建立的一种全局优化方法，利用粒子间的合作与竞争，实现最优解的搜索。

**基本思想：**每个粒子会根据自身经验的积累和群体知识的积累，根据当前最佳的个体，更新每个粒子。搜索效率较高，不存在交叉、变异操作。

**个体表现形式：**每一只鸟被视为一个粒子，粒子的表现形式只有两个变量：速度和位置，即鸟飞行的速度和位置。

**实现：**基于种群的演化，这里种群是指一群鸟(粒子)。从随机的粒子群体出发，通过设定某“适应度函数”来评价每个粒子（解或个体）的优劣，通过速度和位置的不断更新，直到搜索结束。在搜索过程中，只有当前最佳的个体能提供信息给其他粒子（这与遗传算法不同）。

# 第九章：集群智能仿生



## □ 群体智能（Swarm Intelligence）

### ➤ 粒子群算法(PSO)

**PSO**算法描述了大量的粒子通过模拟鸟群搜寻食物的运动方式，相互协作，完成最优解的搜索过程。每个粒子以一定的速度飞行，在迭代一定次数后，找到全局最优的位置，那么这个“位置”即为全局最优解。

定义一个**D**维的搜索空间（即解的长度为**D**），粒子的数量为**N**(种群大小)，第*i*个粒子的位置表达为  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})^T$

第*i*个粒子的速度表达为  $\mathbf{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD})^T$

经过搜索后，第*i*个粒子的最佳位置为  $\mathbf{p}_i = (p_{i1}, p_{i2}, \dots, p_{iD})$

整个粒子群的最佳位置为  $\mathbf{P}_g = (p_{g1}, p_{g2}, \dots, p_{gD})$

# 第九章：集群智能仿生



## □ 群体智能 (Swarm Intelligence)

### ➤ 粒子群算法(PSO)

粒子速度的直接更新（不需要交叉变异），因此收敛更快：

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_{1i}(t) \cdot [p_{id}(t) - x_{id}(t)] + c_2 \cdot r_{2i}(t) \cdot [p_{gd}(t) - x_{id}(t)]$$

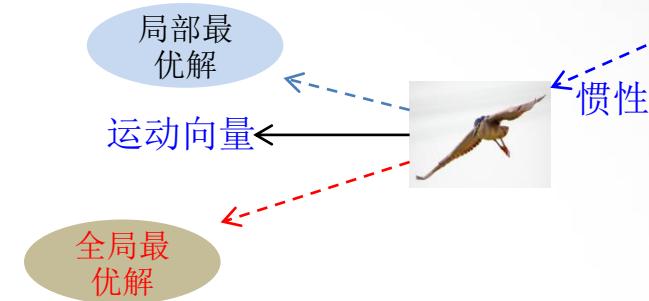
更新方程中涉及三个部分：

第一部分为粒子先前的速度；

第二部分为“自我认知”部分，即第*i*个粒子自身的思考，可理解为第*i*个粒子当前位置与自己最好位置之间的距离；

第三部分为“社会经验”部分，表示粒子间的信息共享与协作，可理解为第*i*个粒子当前位置与群体中最好位置之间的距离；

位置更新： $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), 1 \leq i \leq N, 1 \leq d \leq D$



# 第九章：集群智能仿生

## □ 群体智能（Swarm Intelligence）

### ➤ 粒子群算法(PSO)

粒子速度的更新：

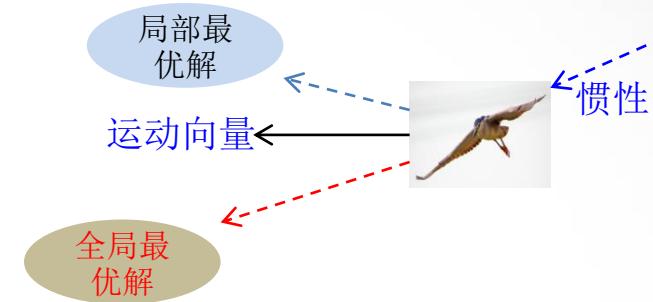
$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_{1i}(t) \cdot [p_{id}(t) - x_{id}(t)] + c_2 \cdot r_{2i}(t) \cdot [p_{gd}(t) - x_{id}(t)]$$

粒子位置更新：

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), 1 \leq i \leq N, 1 \leq d \leq D$$

其中，**c1**和**c2**是正的加速常数，**w**为惯性权重因子，**r1,i**和**r2,i**为区间[0,1]内的随机数

➤ 在实际应用中，可以设计**w**的不同表达形式，产生不同的启发式算法。



# 第九章：集群智能仿生

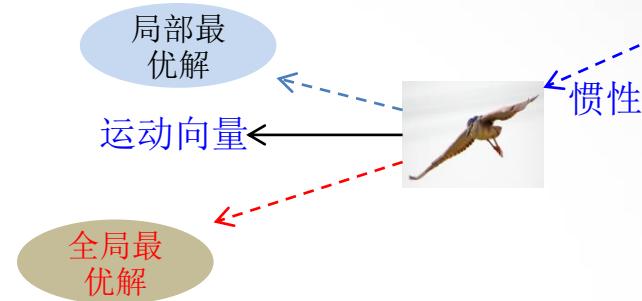


## □ 群体智能 (Swarm Intelligence)

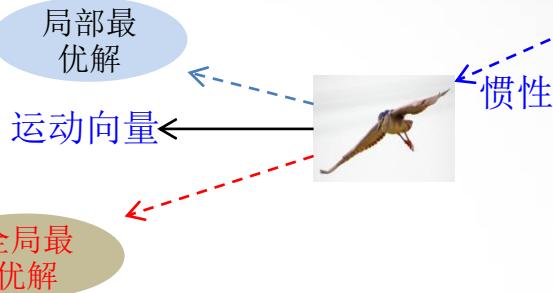
### ➤ 粒子群算法(PSO)

PSO实现过程：

- (1) 初始化粒子群体；规模为 $n$ , 包括位置 $\mathbf{x}$ 和速度 $\mathbf{v}$ 。
- (2) 适应度评价；评价每个粒子的优劣。
- (3) 找每个粒子的最佳位置**Pbest**。对于每个粒子 $x$ , 其当前适应度值如果高于其个体的历史最佳为**pbest**, 则更新**pbest=x**;
- (4) 找全局最佳位置**Gbest**; 找出粒子群适应度最高的粒子位置 $x$ , 如果该粒子适应度比当前全局最佳位置**gbest**高, 那么更新当前全局最佳位置**gbest=x**;
- (5) 利用公式更新粒子（即速度 $\mathbf{v}$ 和位置 $\mathbf{x}$ ）
- (6) 检验收敛条件是否满足。



# 第九章：集群智能仿生



## □ 群体智能 (Swarm Intelligence)

### ➤ 参数控制与分析

如果  $w=0$ , 则粒子没有记忆性;

如果  $c_1=0$ , 则粒子没有“认知”, 只有社会经验, 收敛速度会更快, 但易陷入局部最优;

如果  $c_2=0$ , 则粒子只有“认知”能力, 没有“社会”性, 则粒子之间没有信息共享和合作, 独立搜索, 也就没有了群体智能, 不可能达到最优解

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_{1i}(t) \cdot [p_{id}(t) - x_{id}(t)] + c_2 \cdot r_{2i}(t) \cdot [p_{gd}(t) - x_{id}(t)]$$

粒子间信息共享的部分, 仅利用最优的粒子信息



## 第九章：集群智能仿生

### □ 群体智能（Swarm Intelligence）

#### ➤ 蚁群算法(ACO)

蚂蚁是自然界中常见的一种生物，随着现代仿生学的发展，Dorigo等人首次将蚁群系统发展成了有用的优化和控制算法，即蚁群算法。

**基本思想：**虽然单个蚂蚁的行为极其简单，但群体行为具有自组织性，相互协作的蚂蚁群体能够通过自身释放的“**信息素**”传递信息，很容易找到巢穴到食物源的最短路径，当出现障碍物时，蚂蚁还能够根据对环境的自适应变化，找到最优路径。

**实现：**选择机制、信息素更新机制、协作机制。信息素越多的路径被选中的概率越大（选择机制）；路径越短，信息素增加越快（信息素更新机制）；个体间通过信息素进行交流（协作机制）。蚂蚁算法已成功应用于解决复杂的组合优化问题，如旅行运营商最优路径搜索问题（TSP问题）。

信息素



蚂蚁之间通信的媒介



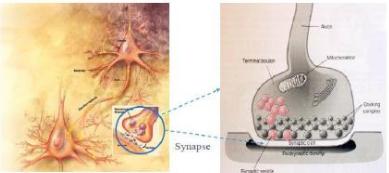
# 第十章：神经网络与深度学习



# 第十章：神经网络与深度学习

## □ 神经网络发展史

- 神经元学说
- 神经元的连接方式被发现

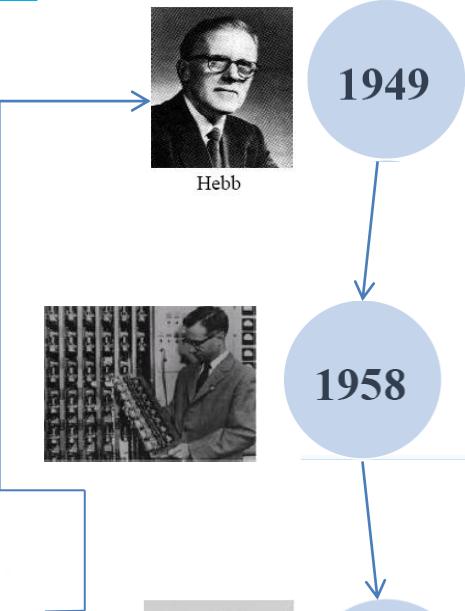
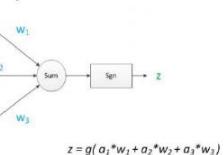


1906

神经元MP模型, McCulloch, Pitts



1943



Marvin Minsky

- Hebb学习规则
- $\Delta W_j = k \times f(W_j^T X)X$

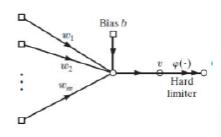
1949



Hebb

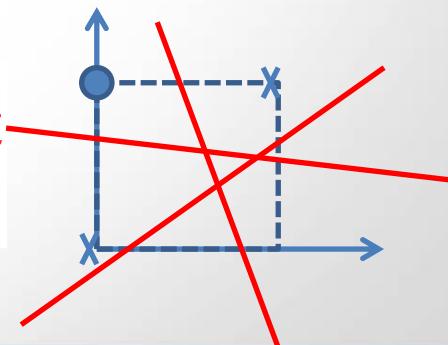
- 感知机(Perceptron)模型

1958



- XOR (异或) 问题
- 计算能力的限制

1969





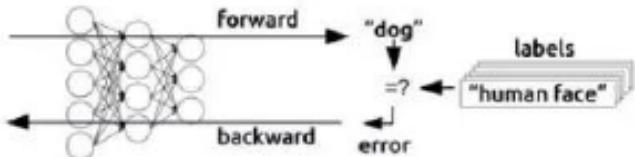
# 第十章：神经网络与深度学习

## □ 神经网络发展史



1986

BP神经网络, Rumehart, Williams, Hinton



G.E. Hinton



1995

■ 支持向量机(SVM), Vapnik

Vladimir Vapnik



2006

G.E. Hinton

- 多层神经网络
- 深度学习

2012

AlexNet在ImageNet比赛  
击败传统机器学习方法

IMAGENET Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)  
Held in conjunction with PASCAL Visual Object Classes Challenge 2012 (VOC2012)

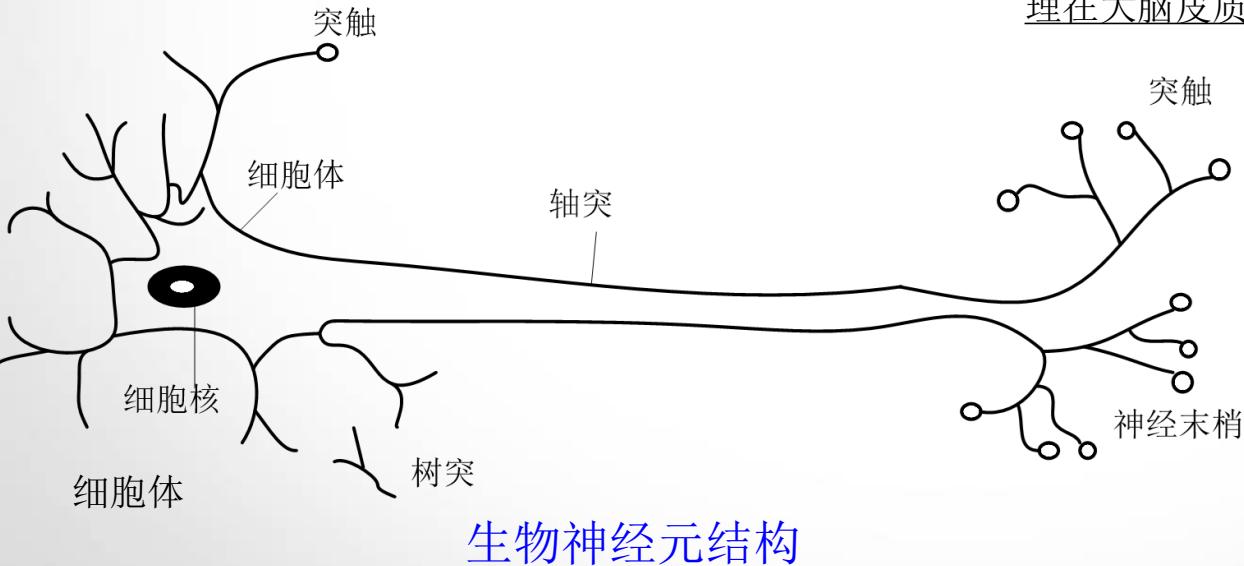
2016

AlphaGo

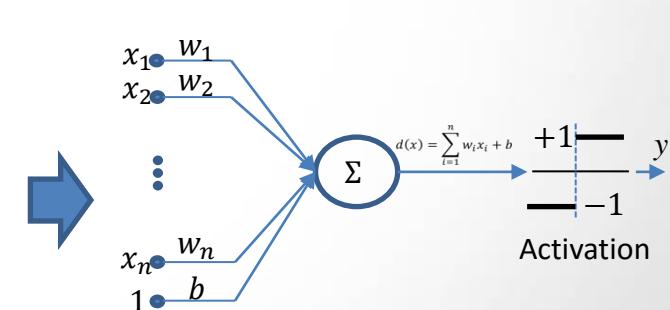


## □ 感知器(Frank Roseblatt)

生物神经元（神经细胞）结构：



每个神经元由一个细胞体组成，其包含一个细胞核。从细胞体分支扩展出许多为树突的神经纤维和一根长的轴突。一个神经元与10到10万个其他神经元相连，连接处称为突触。信号通过复杂的化学反应从神经元传播到神经元。研究发现，大多数的信息处理在大脑皮质，即大脑外层进行。

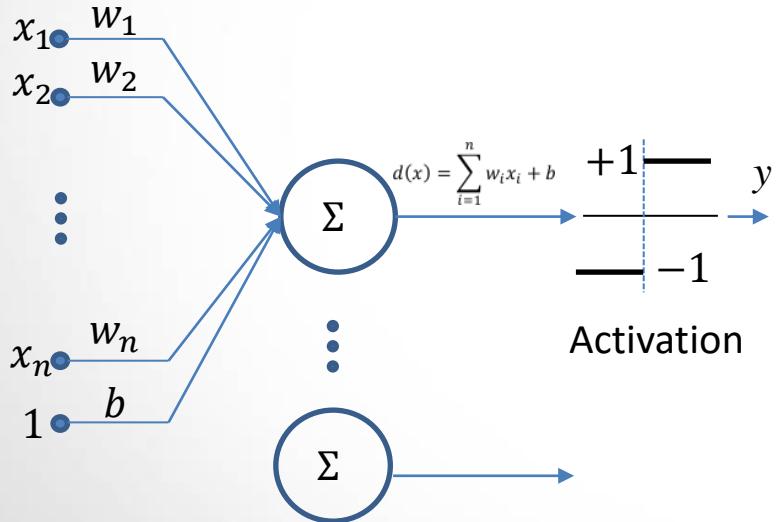


人工神经元结构



# 第十章：神经网络与深度学习

## □ 感知器



如何计算权重w和b呢？

### ✓ 感知器学习规则

采用类梯度下降方法：

- 如果第j个神经元输出是正确的，则 $w_{ij}$ 和偏差 $b_j$ 不变；
- 如果第j个神经元输出是0，而期望值是1，则 $w_{ij}=w_{ij}+x_i$   
 $b_j=b_j+1$ ；
- 如果第j个神经元输出是1，而期望值是0，则 $w_{ij}=w_{ij}-x_i$   
 $b_j=b_j-1$ ；



w和b的学习算法：

设第j个神经元输出为 $y_j$ , 期望值为 $t_j$ , 则学习规则为

$$\begin{aligned} w_{ij} &= w_{ij} + (t_j - y_j)x_i \\ b_j &= b_j + (t_j - y_j)1; \end{aligned}$$

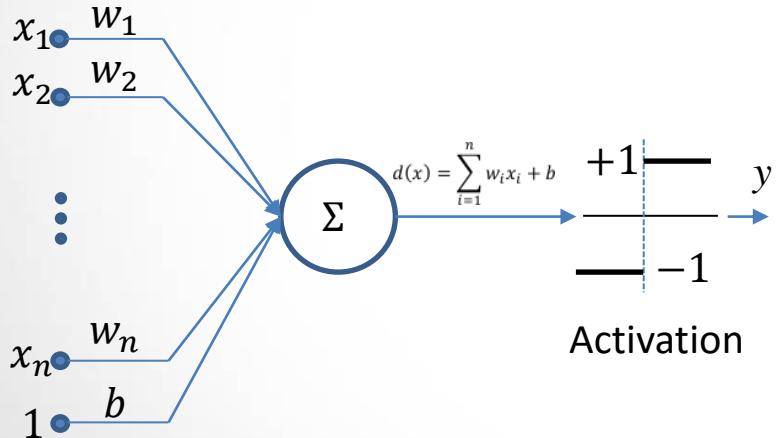
$Y=T$

→ 目标任务



# 第十章：神经网络与深度学习

## □ 感知器



单个神经元的感知器结构

### ✓ 感知器的局限性

1. 感知器的激活函数为阈值函数，输出只能是两个值，因此只能用于简单的分类问题；
2. 感知器是一种线性分类方法，适用于线性可分问题；
3. 收敛速度较慢。

### ✓ 例：利用感知器，对4个二维样本进行分类：

$X = \begin{bmatrix} -2 & -1 & 1 & 0 \\ -1 & -2 & 1 & 2 \end{bmatrix}$ , 目标为  $T = [0 \ 0 \ 1 \ 1]$ ;  
初始权重为  $w = [0, 0]$ ,  $b = 0$