

智能信息处理

主讲：张磊

办公室：主教1030

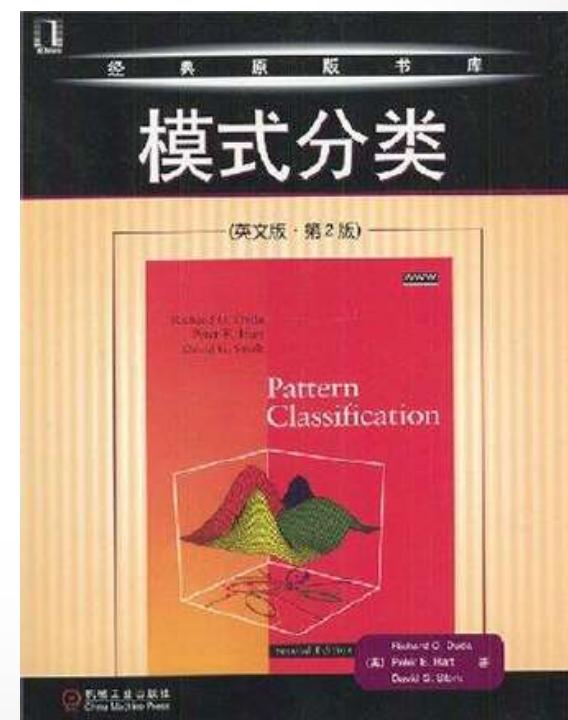
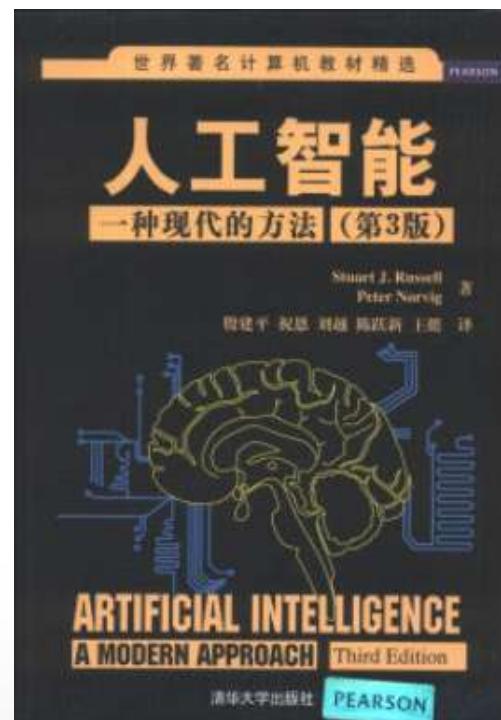
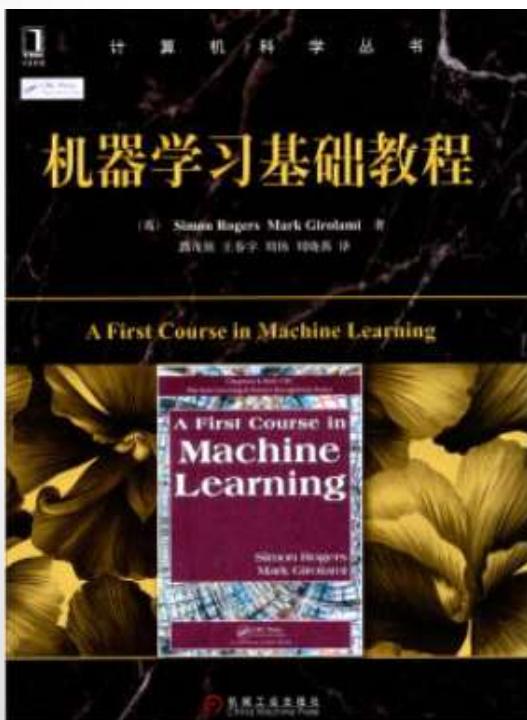
E-mail: leizhang@cqu.edu.cn

Web: <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梦想，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有效见地、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。



第一章：绪论

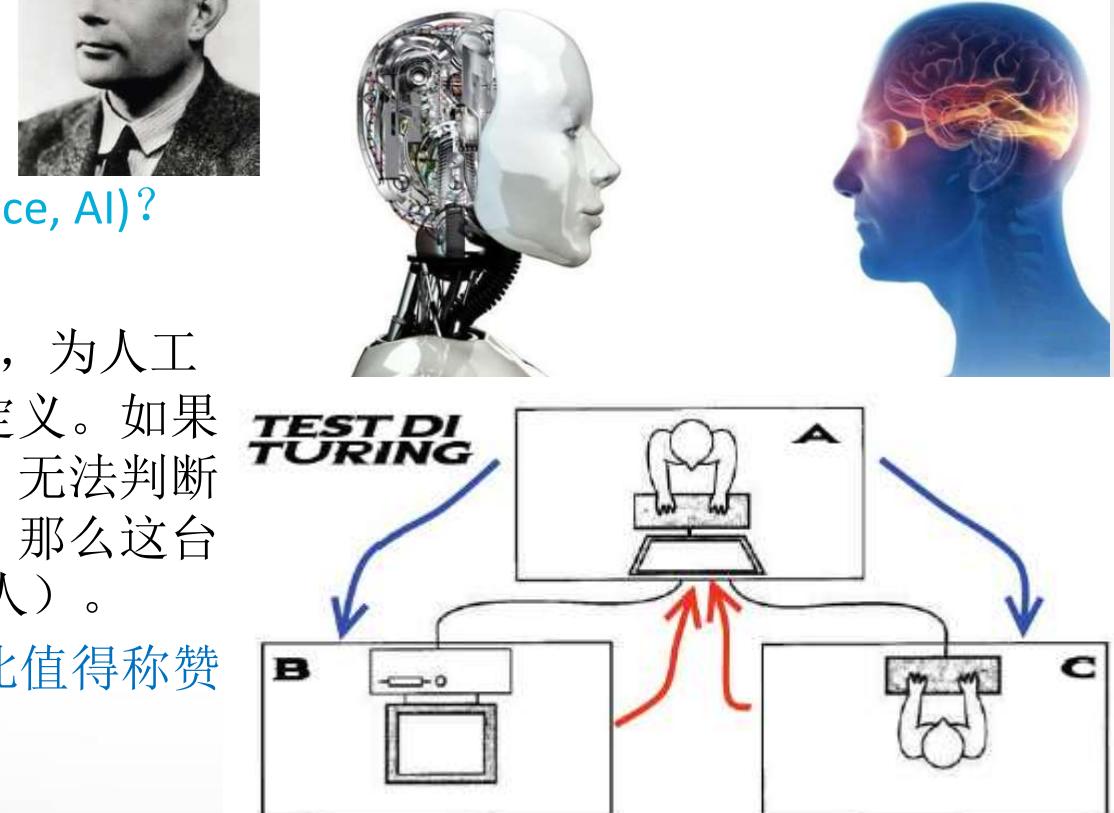


什么是人工智能(Artificial Intelligence, AI)?

◆ 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。

第一章：绪论



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”。



第一章：绪论

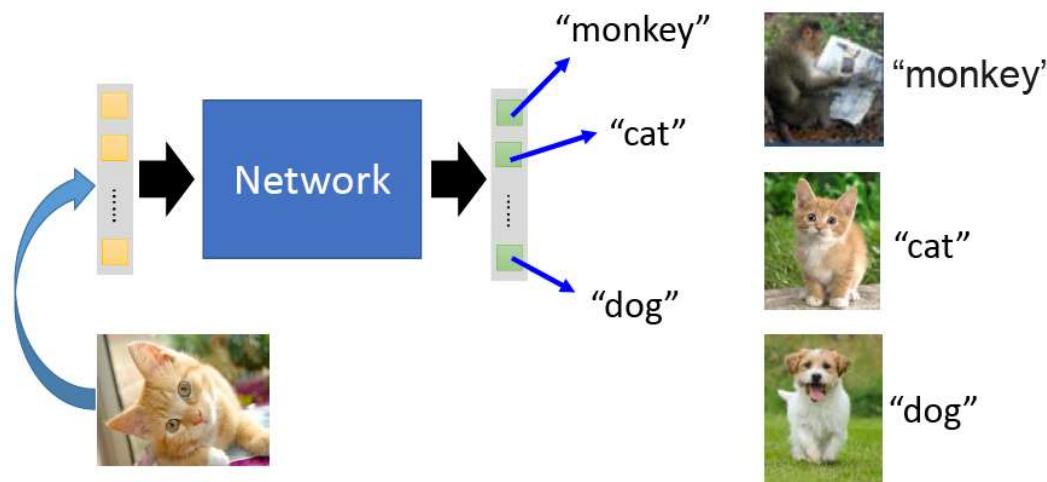
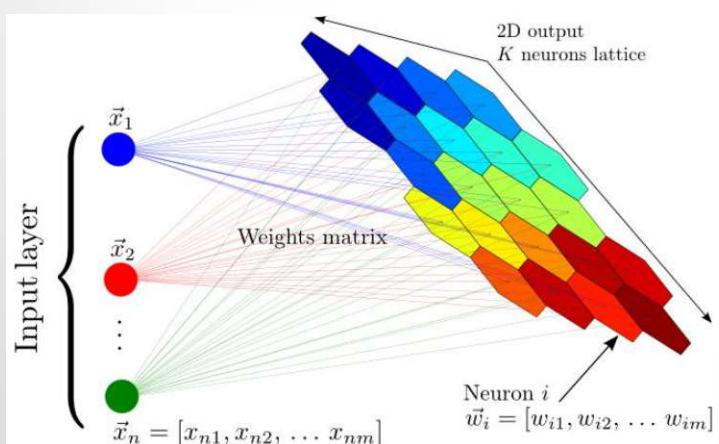


如何实现人工智能(AI)？

◆ 如何让机器像人一样“思考”？

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

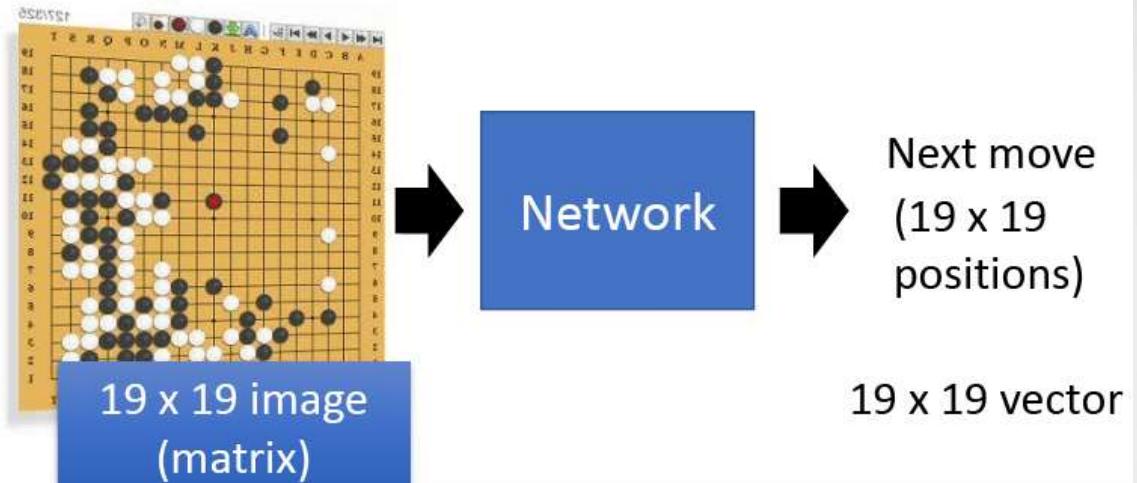
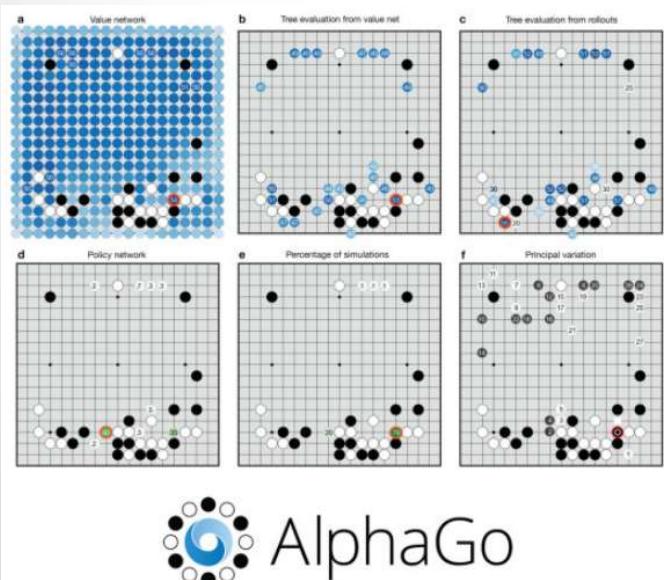


如何实现人工智能(AI)？

◆ 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



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。

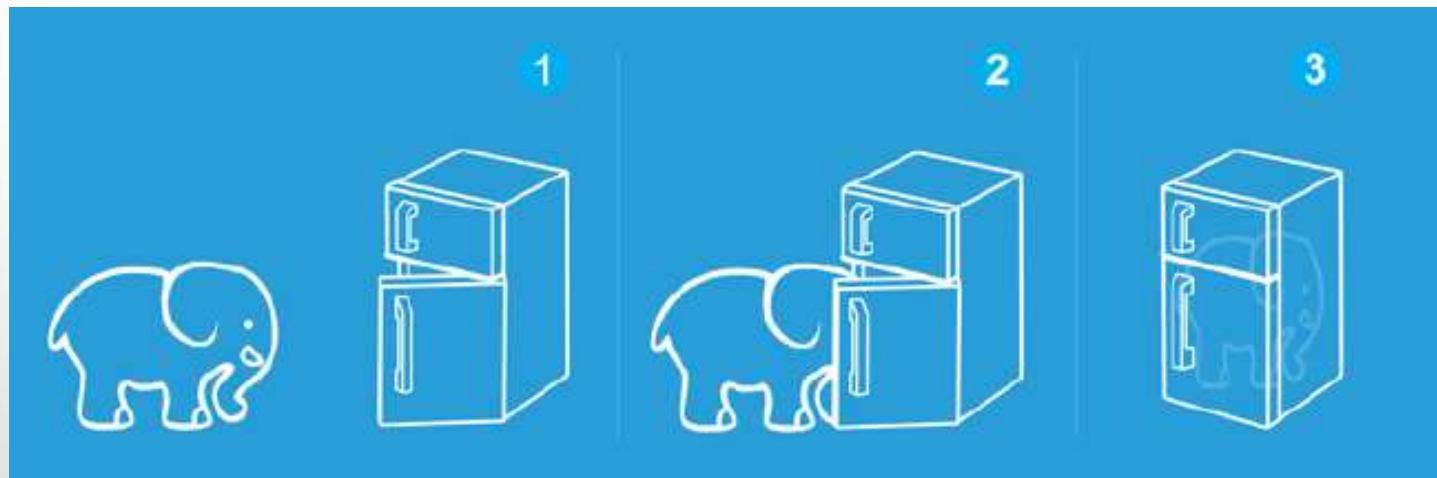
第一章：绪论



如何实现人工智能(AI)?

◆ 如何让机器像人一样“思考”?

深度学习的简单框架：大象和冰箱的故事。

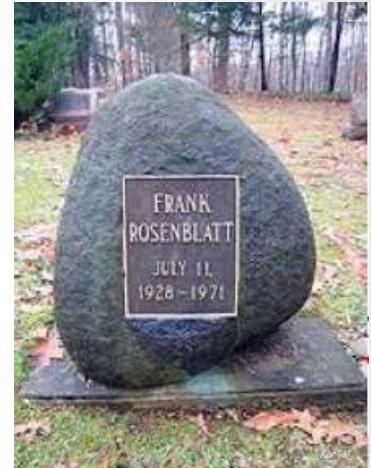


第一章：绪论

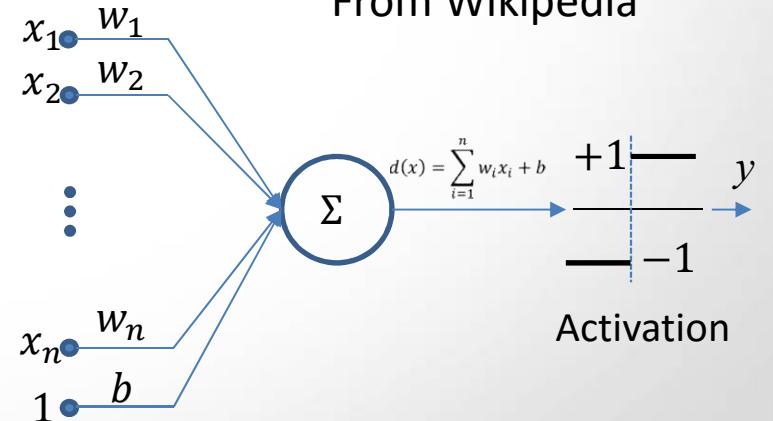
人工智能(AI)起源(60年前， 1950):

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。”

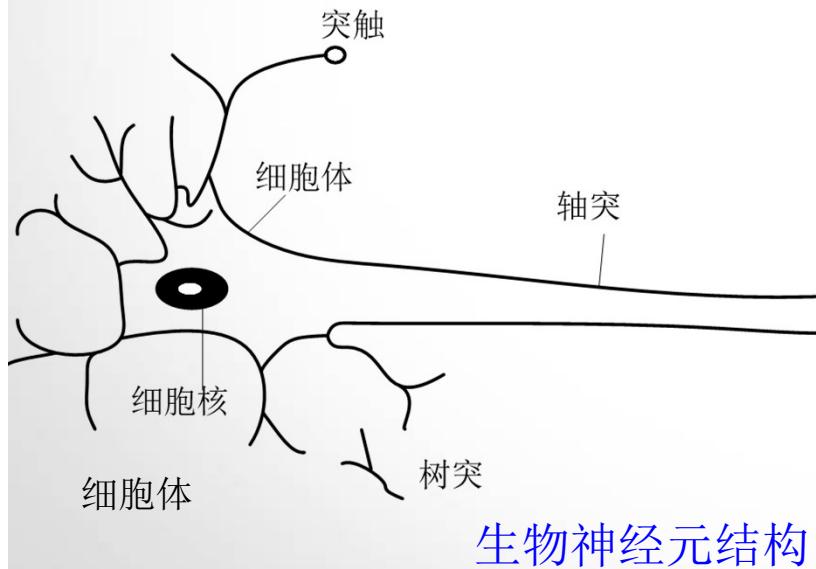


From Wikipedia



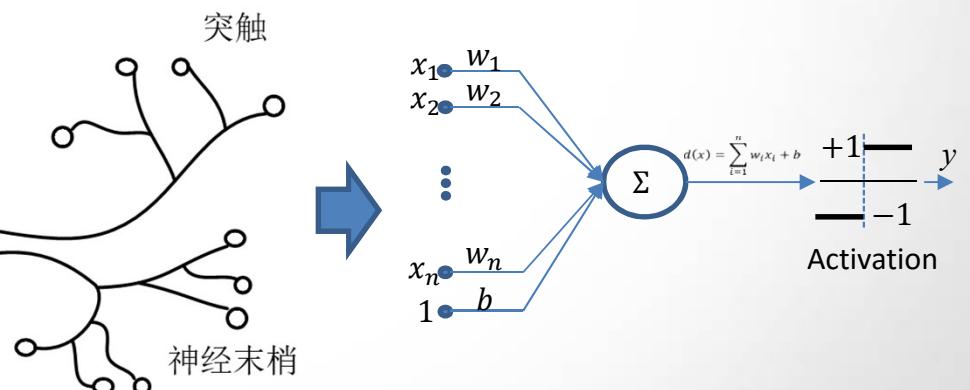
第一章：绪论

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。



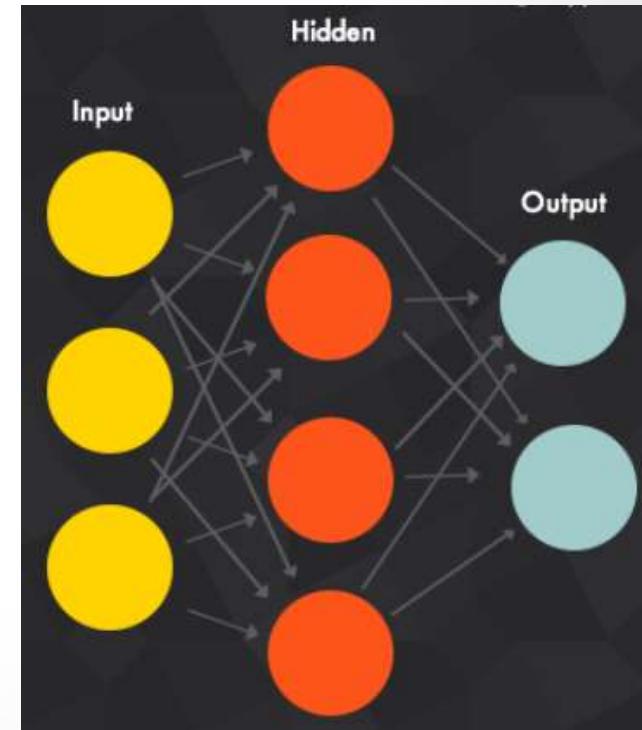
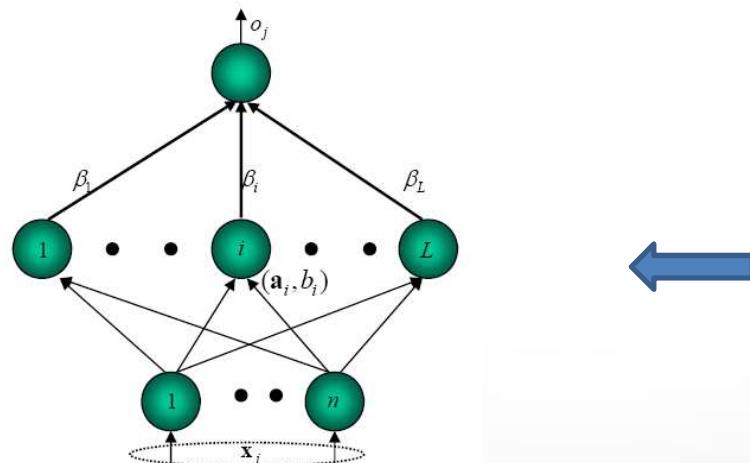
人工神经元结构

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”



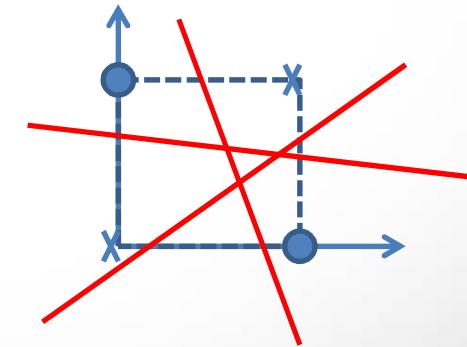


人工智能(AI)冬天(1970s):

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

第一章：绪论

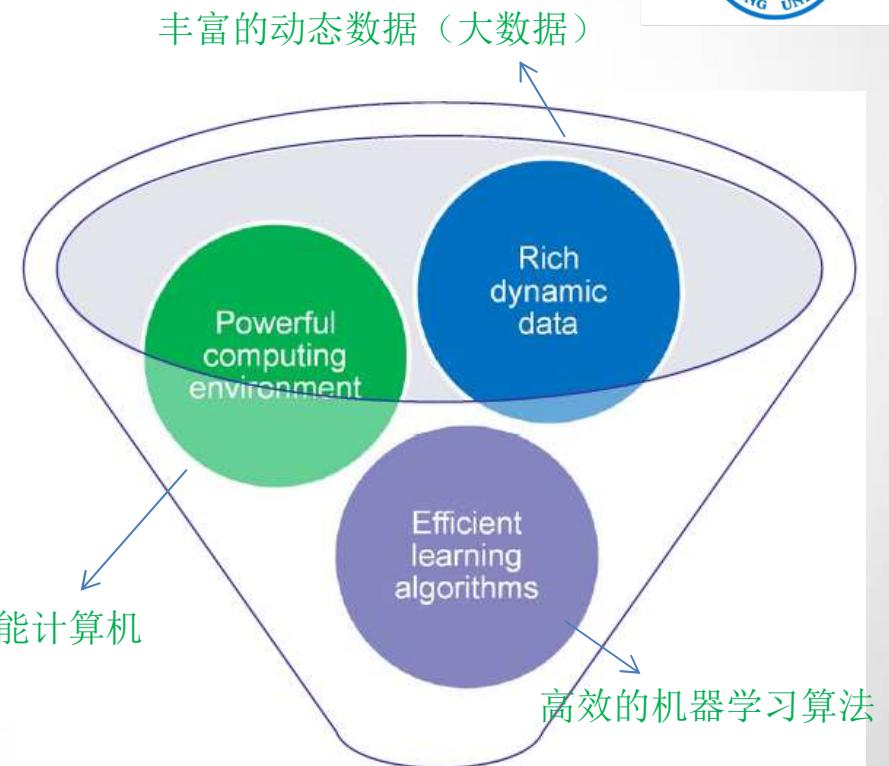


人工智能(AI)基础:

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**：





第一章：绪论

人工智能(AI)最新发展：

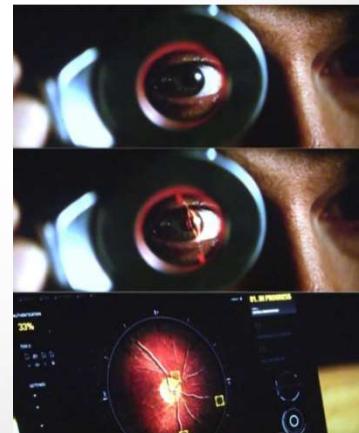
机器视、听、触、嗅、感觉及思维方式的模拟：指纹识别、人脸识别、虹膜识别、掌纹识别、图像/视频识别、语音识别、机器翻译、自然语言处理、搜索/检索、博弈（围棋、象棋）、无人驾驶等



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

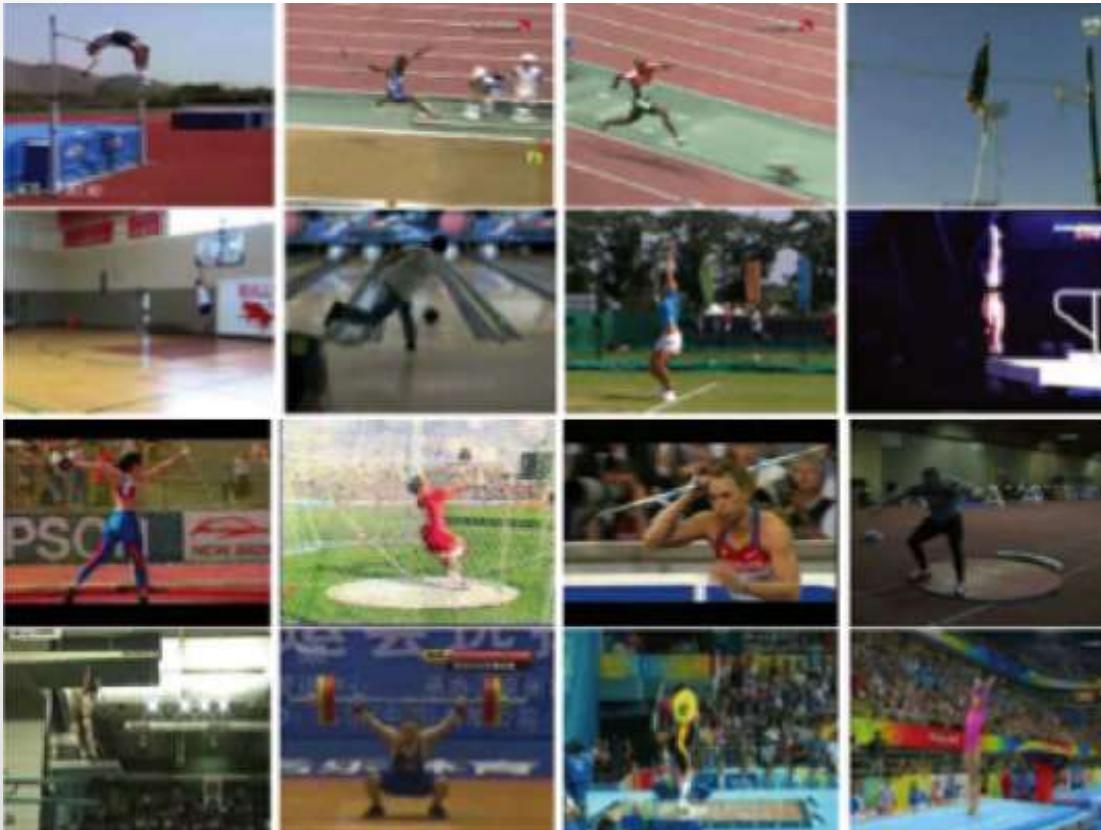


第一章：绪论





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。

第一章：绪论



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



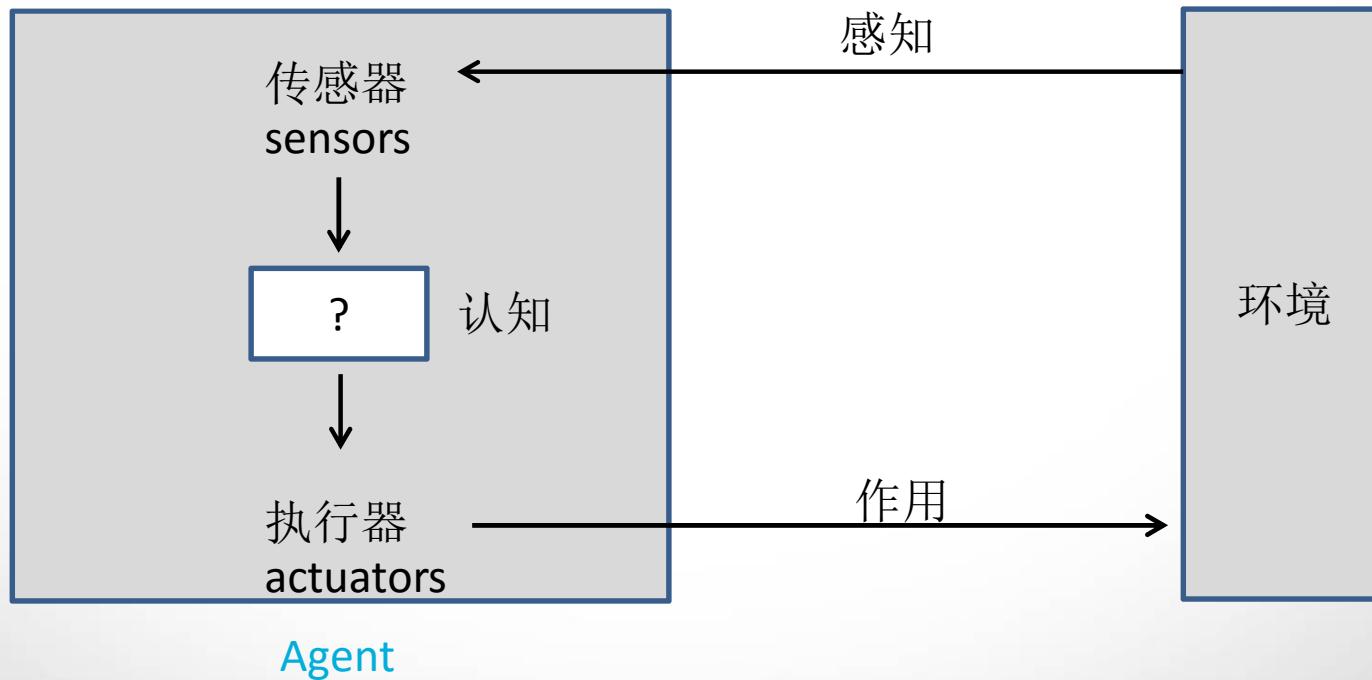
Agent:

- ◆ 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



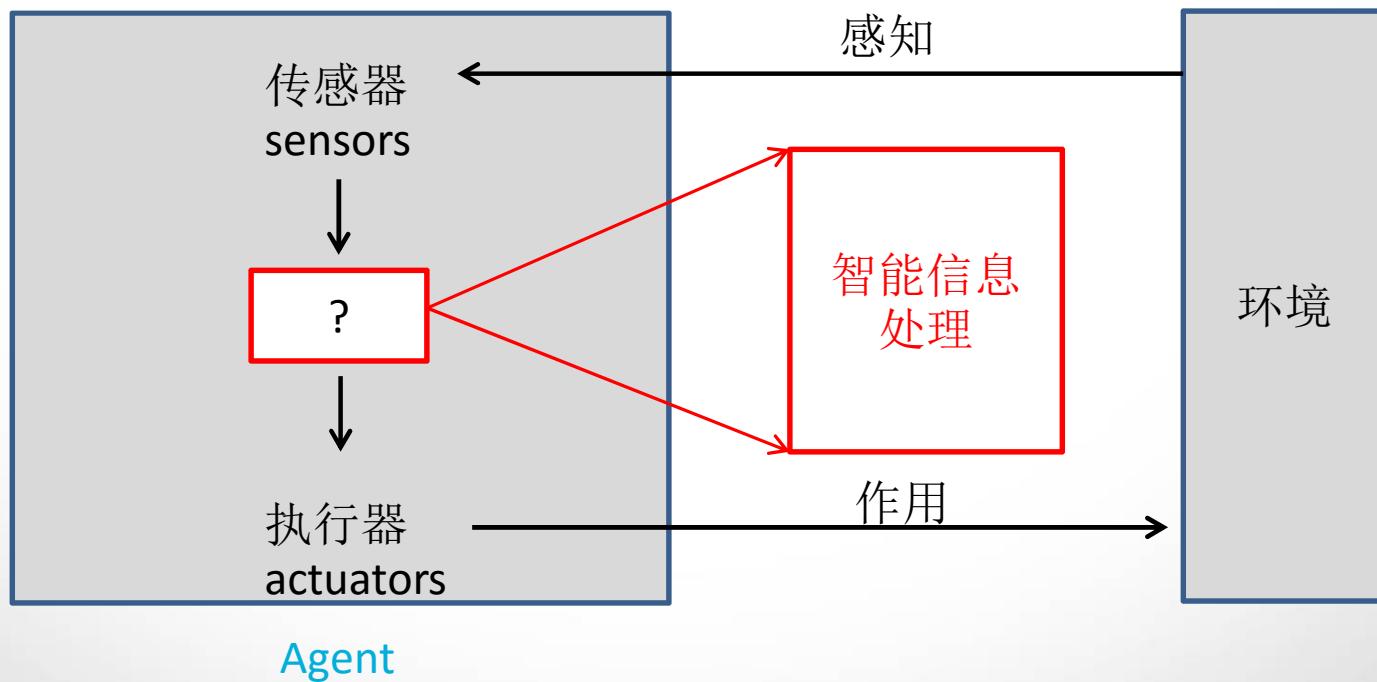
智能体通过传感器和执行器与环境进行交互的过程：



第二章：智能体Agent



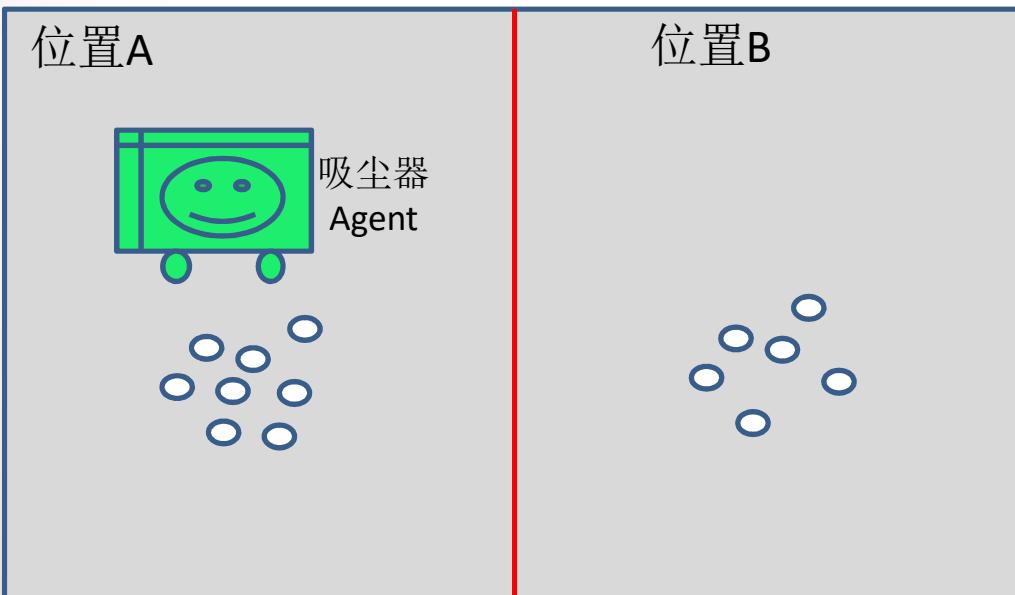
智能体通过传感器和执行器与环境进行交互的过程：



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：



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

吸尘器Agent系统

简单的Agent函数表格



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。



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。



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。



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）。

Agent=体系结构+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。



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 $n \times 1$ 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 $\mathbf{1}_n$ 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



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， vec 是向量化算子。设 $A = [a_{ij}]_{m \times n}$ ，则

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



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 Δx 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

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_p} \end{bmatrix} \in \Re^p \quad \xrightarrow{\text{相当于梯度, 是由 } p \text{ 个偏导数组成的矢量}}$$

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

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



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, $\mathbf{Y} = g(\mathbf{X}) \in \mathbb{R}^{s \times t}$, $\mathbf{z} = f(\mathbf{Y}) \in \mathbb{R}$,

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

(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

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



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 θ 的函数，那么矩阵 \mathbf{M} 对参数 θ 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：

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

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。

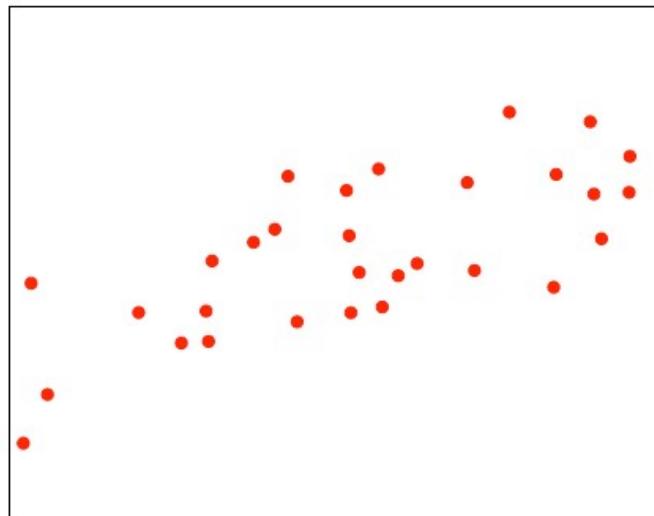
已知条件：已知被点击的电影种类（喜剧、动作、恐怖等），以及该用户对每个种类的点击次数。



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 ω_0 和 ω_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 ω_0, ω_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 ω_0 和 ω_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 ω_0 和 ω_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}

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

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 ω_0, ω_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 ω_0, ω_1 分别求偏导。

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

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

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

例：给出3个样本点组成的集合

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

如何建立线性模型的表达式？

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

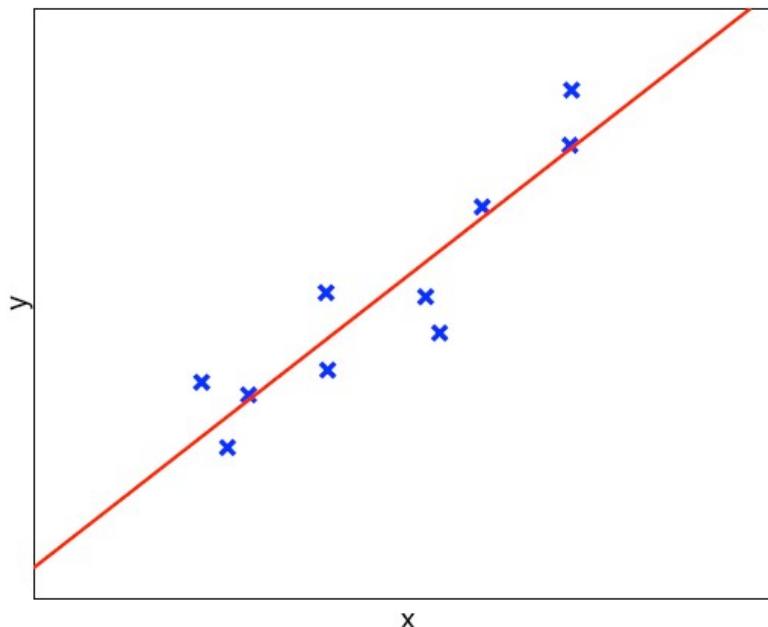
$$t_{new} = \omega_0 + \omega_1 x_{new} = [\omega_0 \ \omega_1] [1 \ x_{new}]^T = \mathbf{w}^T \mathbf{x}_{new}$$



第四章：线性回归算法

Example: Data and best linear hypothesis

$$y = 1.60x + 1.05$$





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， ω_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.



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。

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

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）。



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， ϕ_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

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

其中， ϕ_k 为基函数。

- 线性假设模型还可以重新写成

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

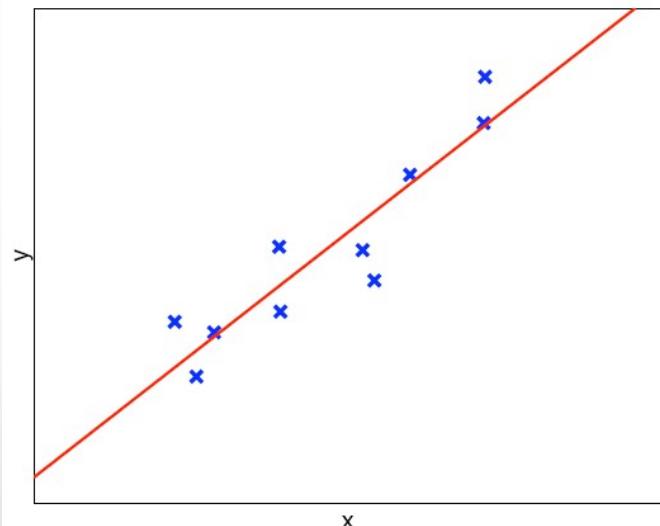
其中， Φ 是由 $\phi(x_j)$ 构成的矩阵。

关于 w 的
线性模型

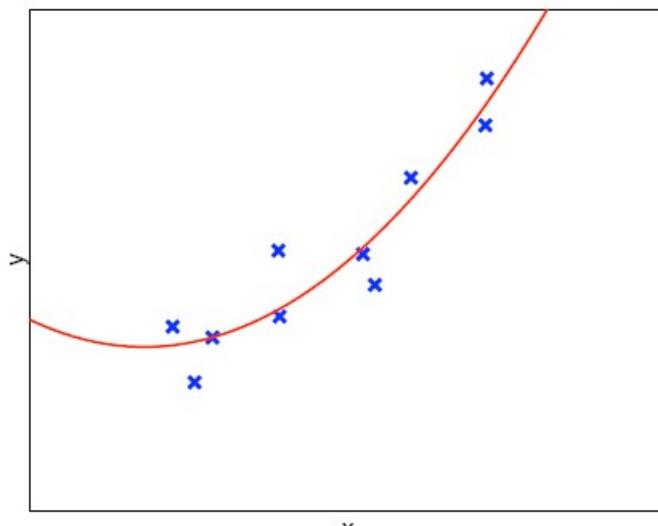


广义线性回归

采用不同基函数（不同阶次）的回归模型效果



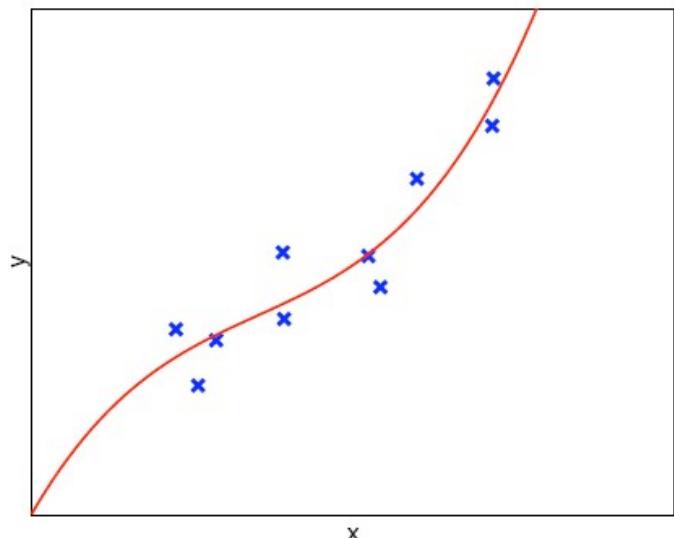
1阶回归 (is it better to fit the data?) 2阶回归



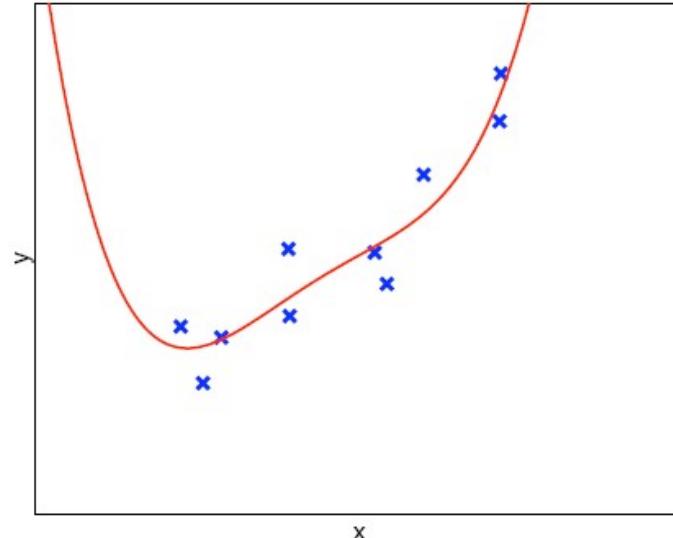


广义线性回归

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?)

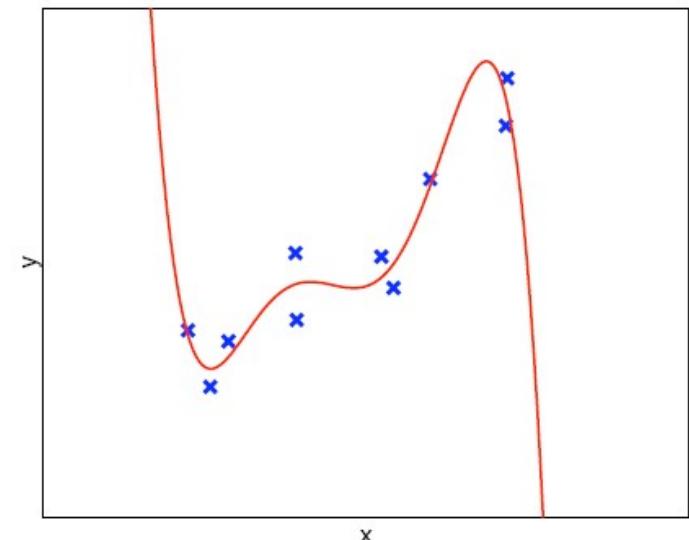


4阶回归

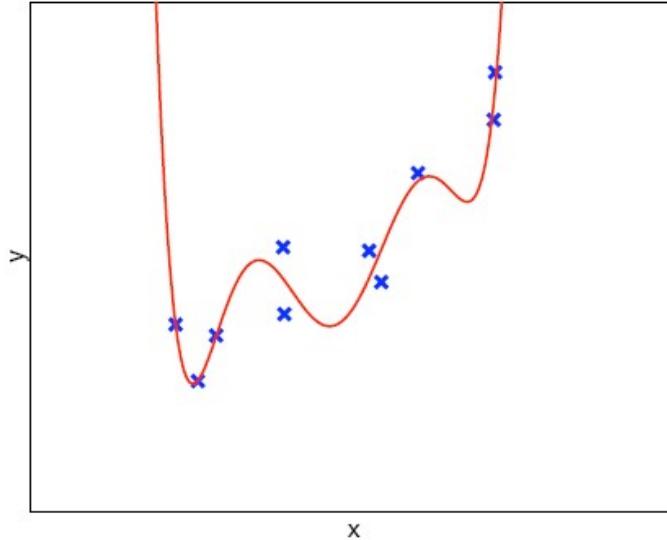


广义线性回归

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?)

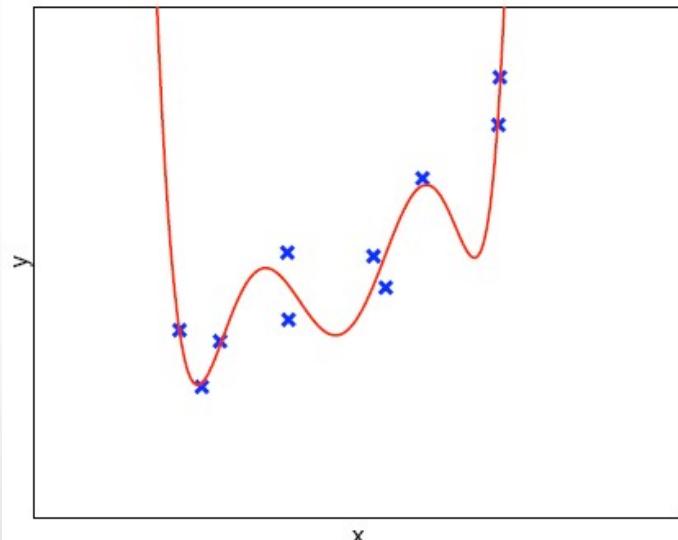


6阶回归

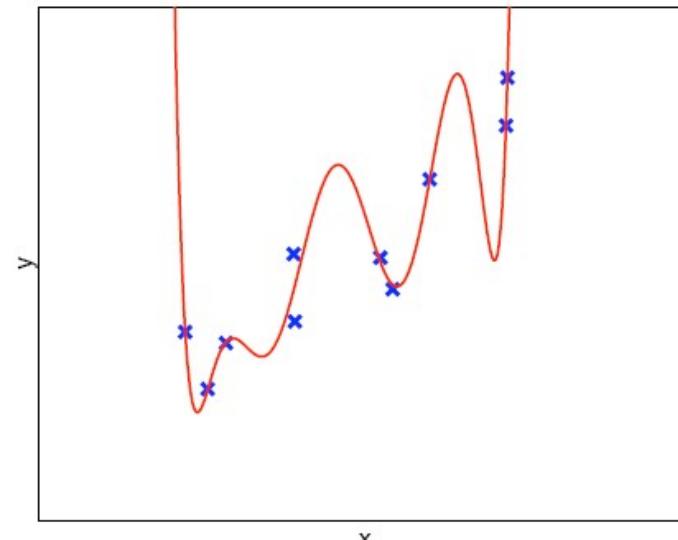


广义线性回归

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?)

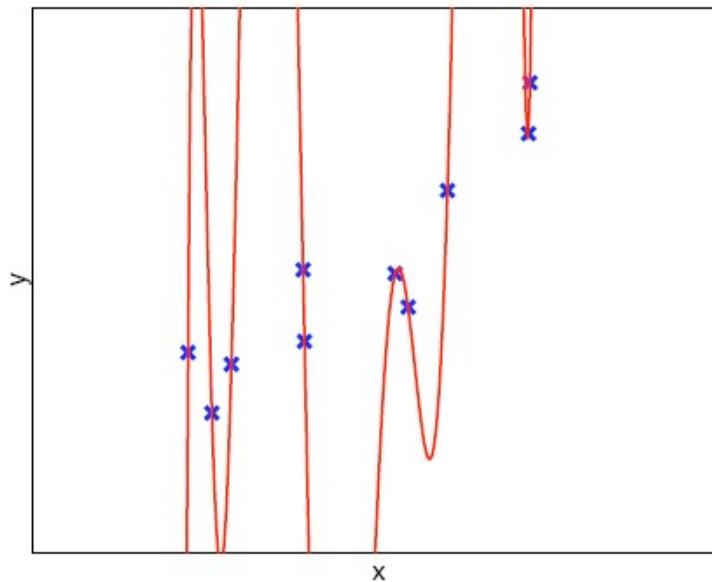


8阶回归



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?)



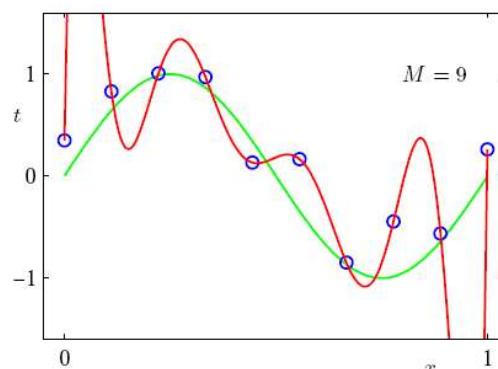
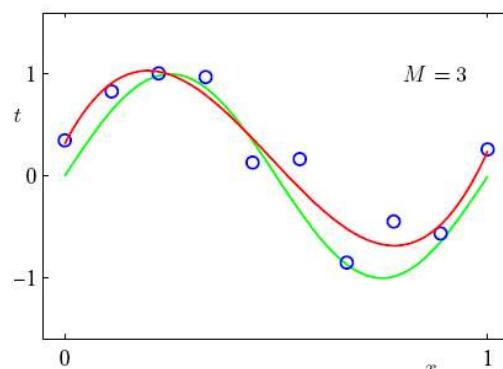
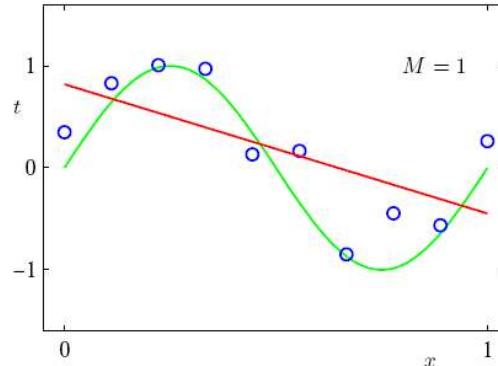
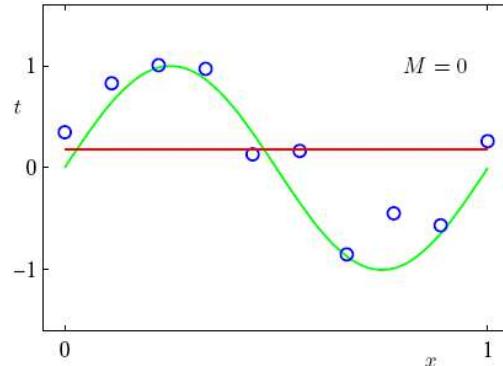
过拟合（overfitting）

- 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）



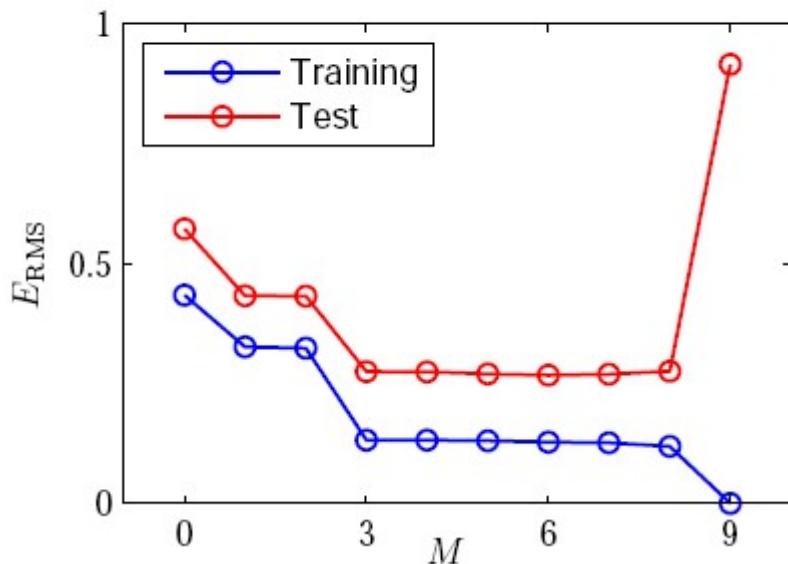
- M 为模型的阶数，反映模型的自由度和复杂度。
- 过拟合是指，训练样本的误差非常低，而新测试样本的预测误差非常大。
- 欠拟合是指，训练样本的预测误差非常高（学习不够）。
- 通常，过拟合是更为常见的问题，因此，在建模过程中，我们需要经验和理论分析，来避免这个问题。



第四章：线性回归算法

过拟合（overfitting）与欠拟合（underfitting）

Typical overfitting plot



- 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：

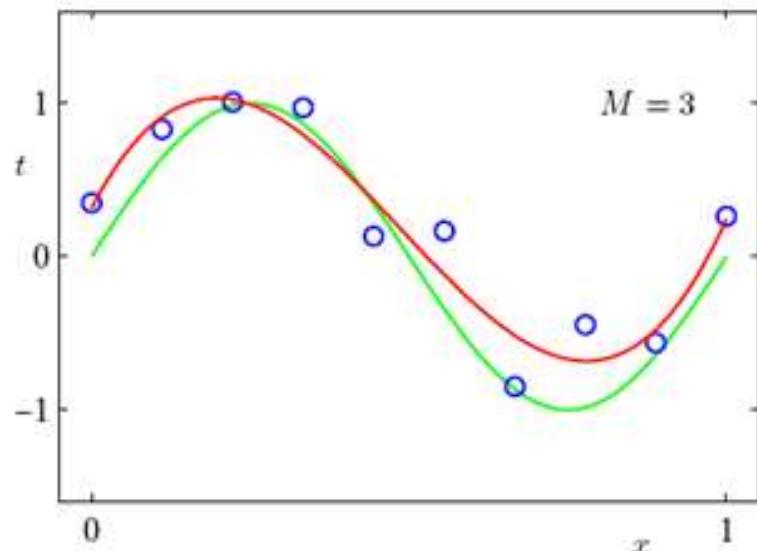
- 将整个数据集分成3个独立的部分：训练集、验证集、测试集
 - ✓ 训练集是用于训练某种假设；
 - ✓ 验证集用于检验假设的可靠性，并进一步修正模型（即 M ）；
 - ✓ 测试集用于检验最终模型假设的性能。

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。

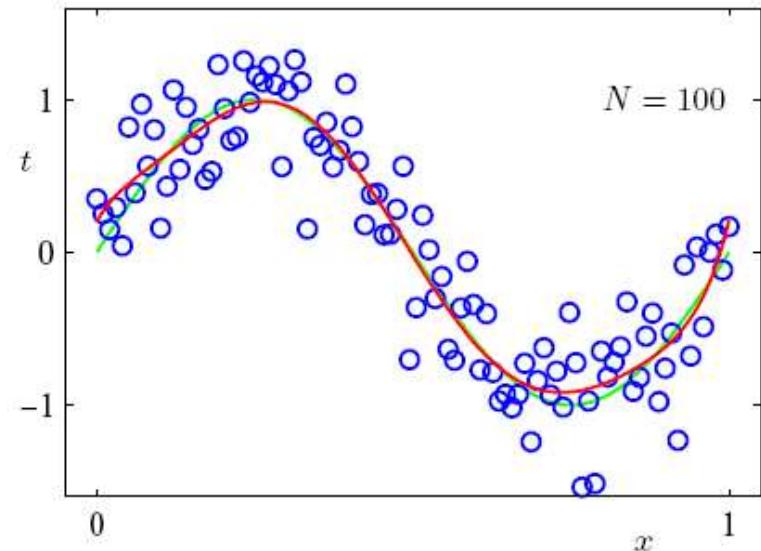
第四章：线性回归算法



最佳的假设 $h(x)$ 依赖于训练样本数量和模型复杂度



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



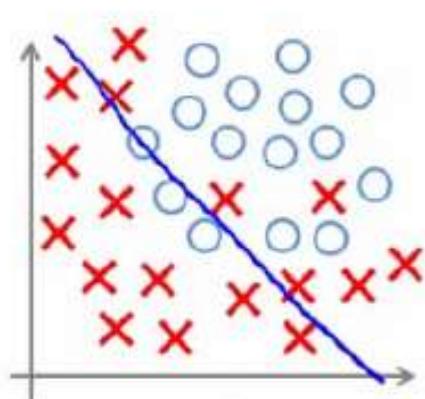
100个训练样本的回归曲线 ($M=3$)
假设与真实更加接近



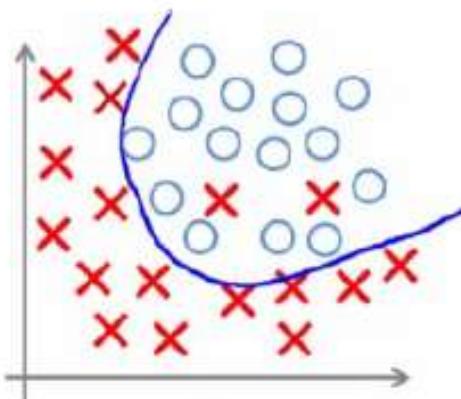
第四章：线性回归算法

最佳的假设 $h(x)$ 依赖于训练样本数量和模型复杂度

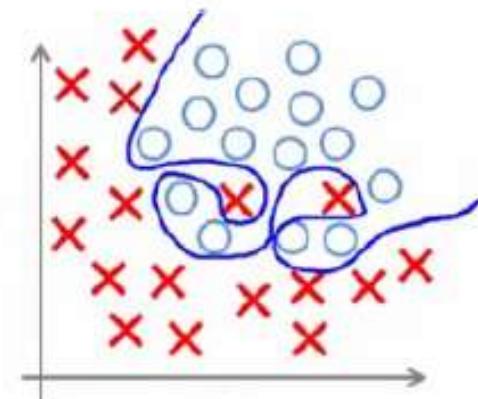
分类问题：



Under-fitting



Appropriate-fitting



Over-fitting



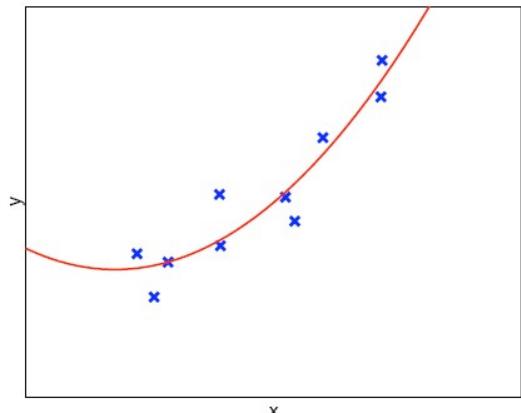
第五章：正则化



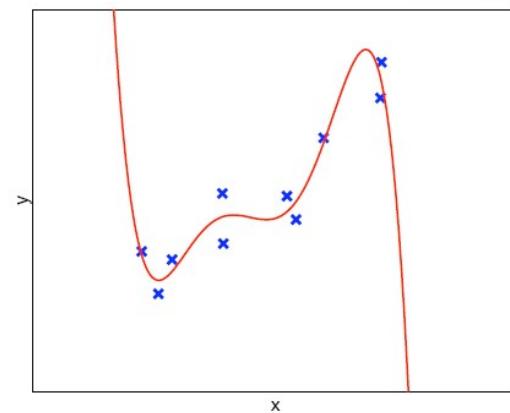
第五章：正则化regularization

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



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



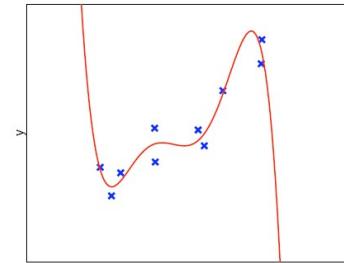
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

第五章：正则化regularization



原线性模型的最小化表达式为

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

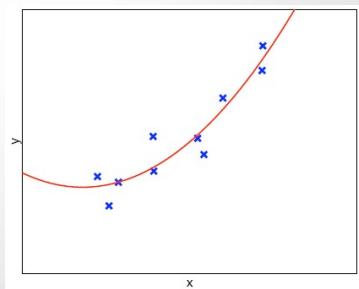


为了获得一个较好的模型，我们希望模型复杂度越小越好（阶数越小），也就是希望得到 $\theta_3, \theta_4, \theta_5 \approx 0$ (参数变少，得到简单的模型假设)
那么模型应当改进如下

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



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





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。

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

正则化项



正则化参数 λ 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。

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

等价于

$$\begin{aligned} & \min_{\theta} \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x_i) - t_i)^2 \\ & s.t. \sum_{j=1}^m \theta_j^2 \leq \xi \end{aligned}$$

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 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$, $\mathbf{Y} = [y_1, y_2, \dots, y_N]$ ，
 \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, Γ 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

□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。

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

L0-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

□多维下的稀疏正则化线性回归模型表达 (Lasso Regression)

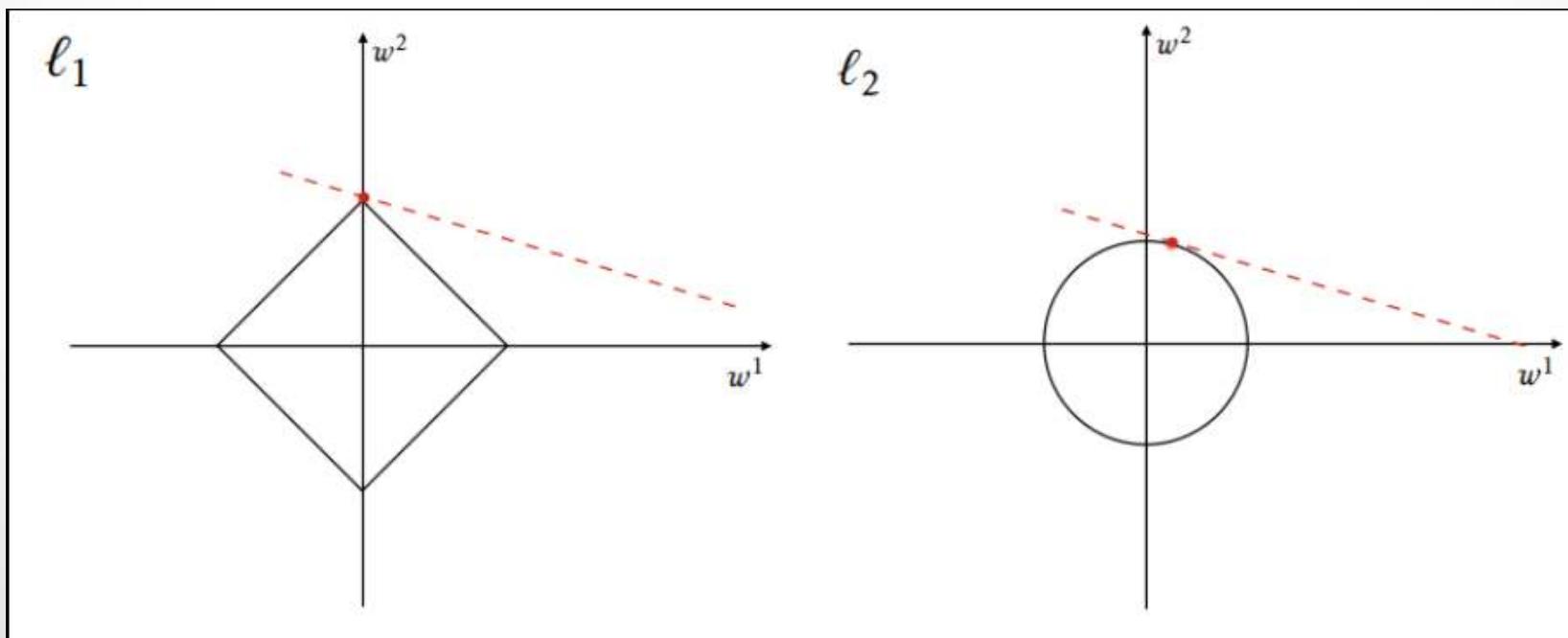
定义一组观测数据集 (\mathbf{X}, \mathbf{Y}) ,其中 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$, \mathbf{X} 为属性, \mathbf{Y} 为响应标签。 \mathbf{x}_i ($i=1, \dots, N$) 是 d -维的向量。

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

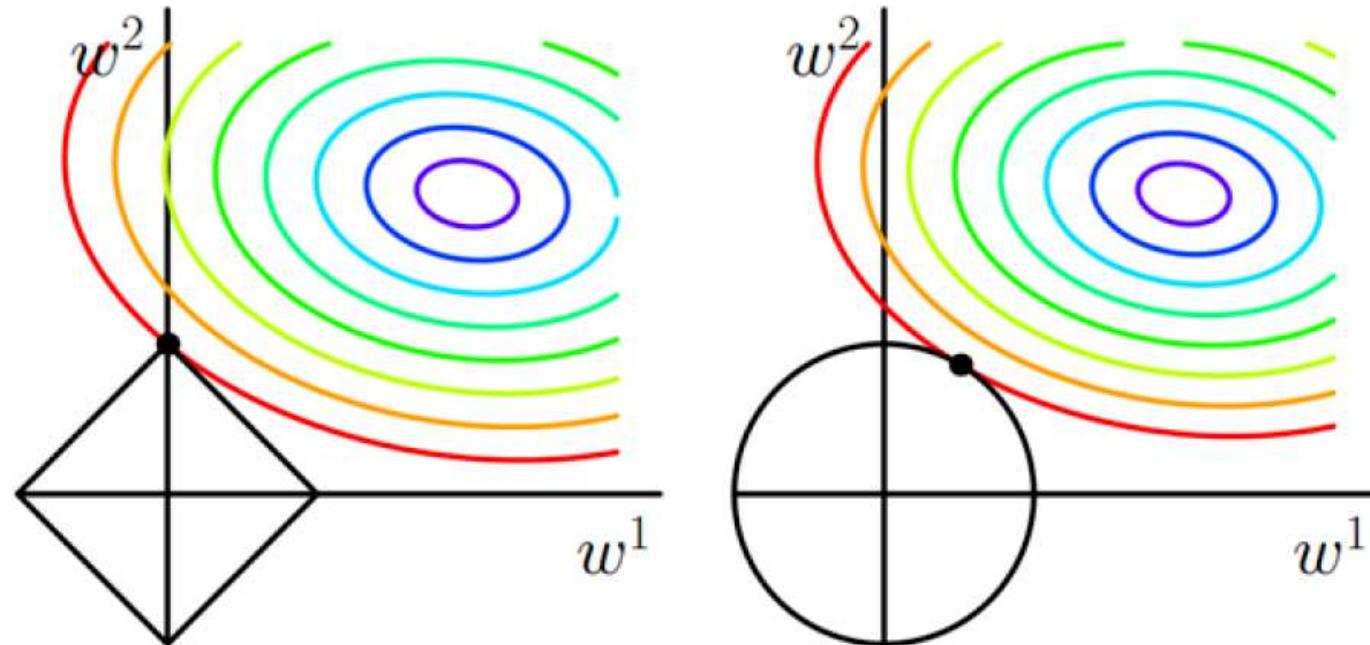
L1-范数的表达式。 $\|\boldsymbol{\theta}\|_1 = \sum_{d=1}^D |\theta_d|$

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 $\mathbf{x}=0$ 处不严格可导/可微，或者说在 $\mathbf{x}=0$ 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

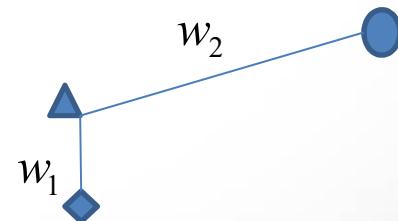
□半监督正则化（流形正则）

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，

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

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

□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

□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， ε_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。

ε_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

□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 ε_n ，可以理解成噪声。

注：这里的随机变量 ε_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 \mathbf{w} 的精确表达式。



第六章：概率建模

□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 ε_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 σ^2 (方差)的值。



第六章：概率建模

□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}, σ^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}, σ^2 已经最优

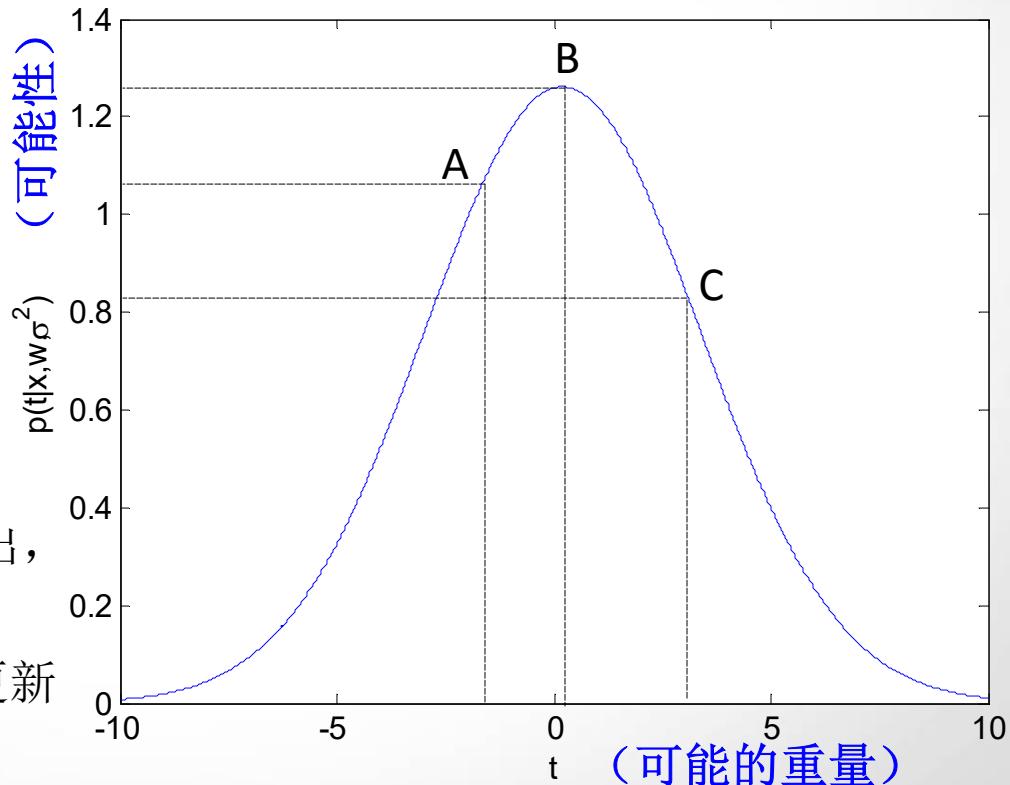
通过右边的概率密度函数可以得出：

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}, σ^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。





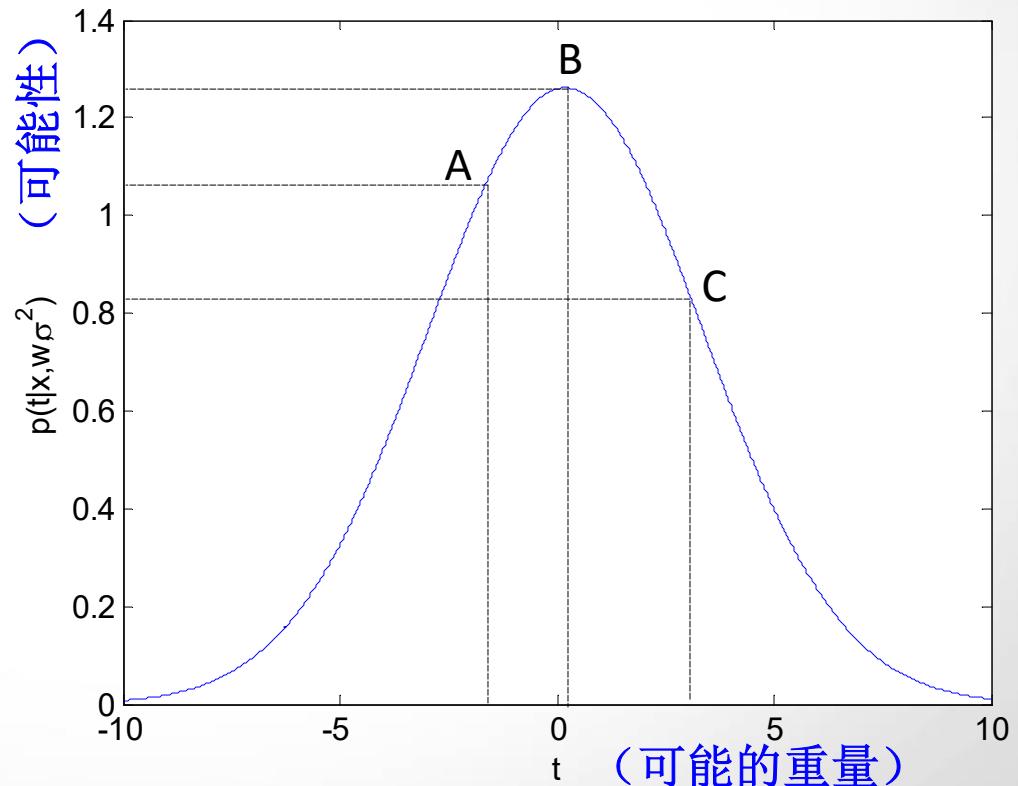
第六章：概率建模

□ 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

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

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

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

第六章：概率建模



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

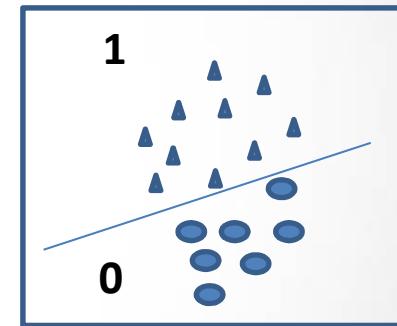
回顾线性回归算法中的二类分类问题

$$h(\mathbf{x}) = \mathbf{w}_0 + \mathbf{w}_1x_1 + \cdots + \mathbf{w}_dx_d$$

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。



第六章：概率建模

□ 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 \mathbf{x} , 其响应 \mathbf{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 $P(Y = 1|X = x) = p(x; \mathbf{w})$, 其中 \mathbf{w} 为参数，那么有

$$P(Y = 0|X = x) = 1 - p(x; \mathbf{w})$$

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：

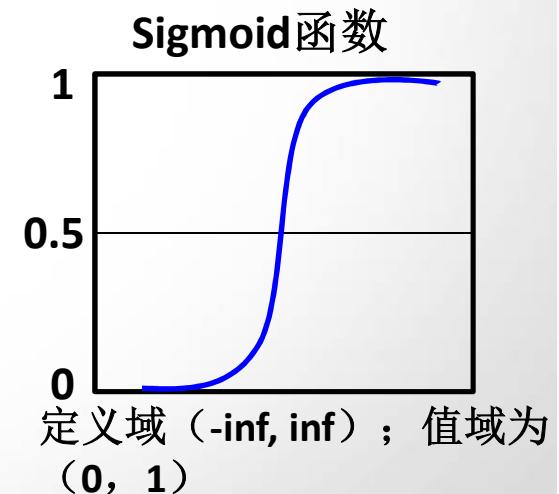
$$h(x) = w_0 + w_1x_1 + \cdots + w_dx_d$$

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

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

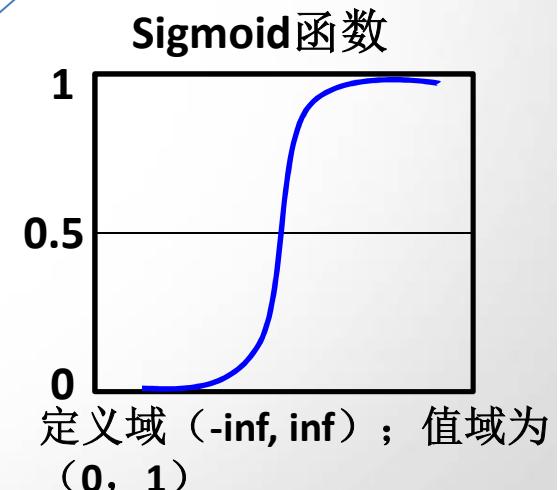
两边同时取对数，

$$\log \frac{p(x)}{1 - p(x)} = h(x) = w_0 + w_1 x_1 + \dots + w_d x_d$$

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变换取可任意值





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 $p(x) > 0.5, Y=1$ ；

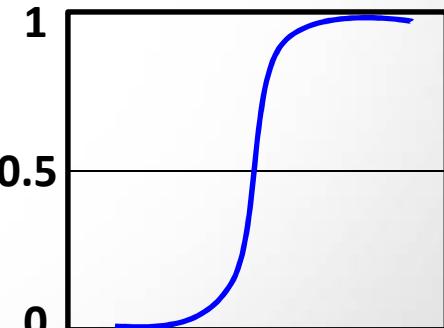
当 $p(x) < 0.5, Y=0$ 。

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



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 $L(\mathbf{w})$,

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

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

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

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 β^* 。我们假设函数 $f(\cdot)$ 是平滑的，那么意味着该函数在 β^* 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 β^* 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 β 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 $L(\mathbf{w})$,

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

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

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

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

□ 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。

