



# 人工智能导论

Artificial Intelligence



# 目标

## TARGET

- ◆ 机器学习基本流程感知
- ◆ 进阶学习AI 分支领域
- ◆ 使用Microsoft Azure 云平台

# 目录

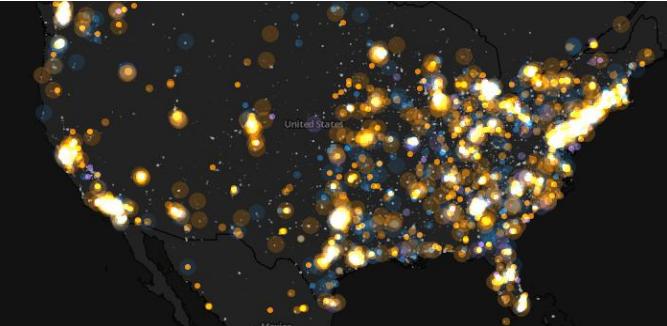
## Contents

- ◆ 应用场景与发展历程
- ◆ 人工智能主要分支
- ◆ 机器学习工作流程
- ◆ 机器学习算法分类
- ◆ 模型评估

# ■ 课题导入



Barack Obama's Political Election



Social Networking



Bruegel



Dead Sea Scrolls



# 1-1 行业应用

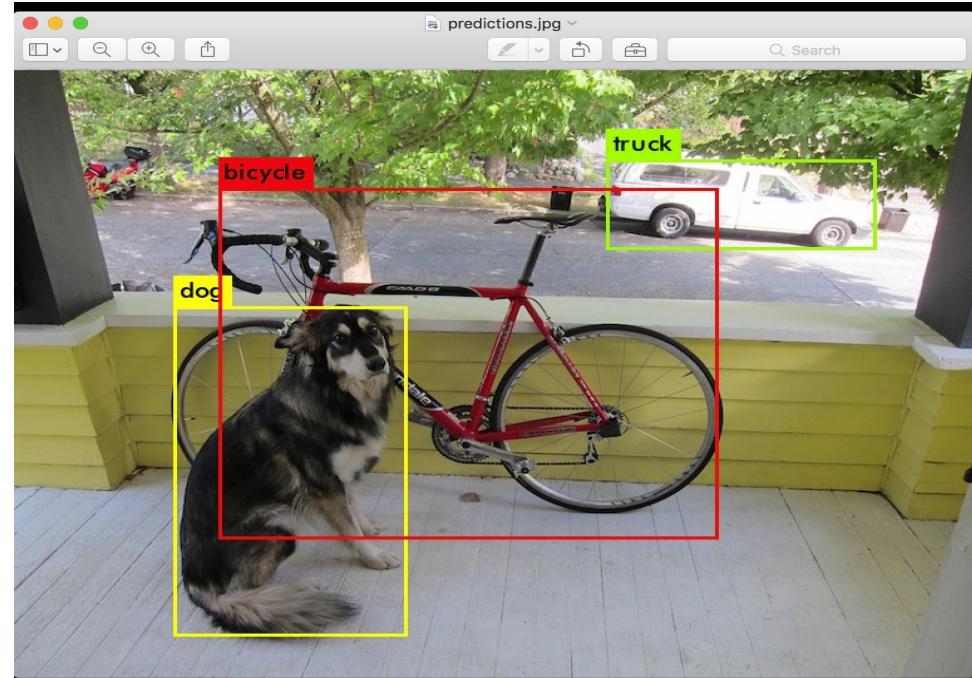
## DeepDream

<https://deepprojects.org/deeplearning/deepdream/>



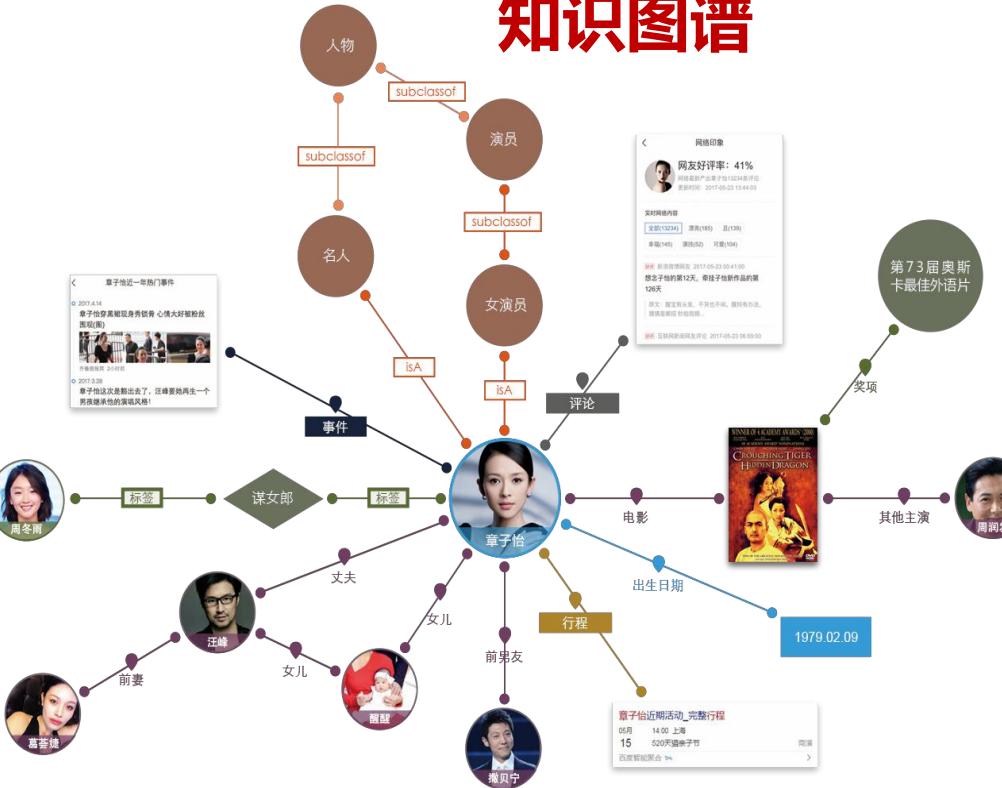
## YOLO

<https://pjreddie.com/darknet/yolo/>



# 1-1 行业应用

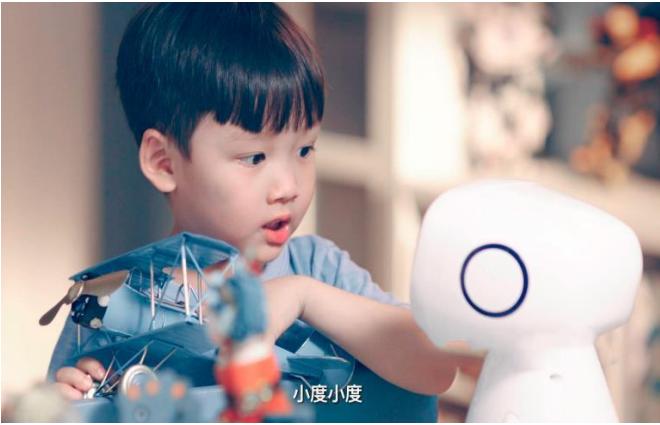
## 知识图谱



## 画风迁移



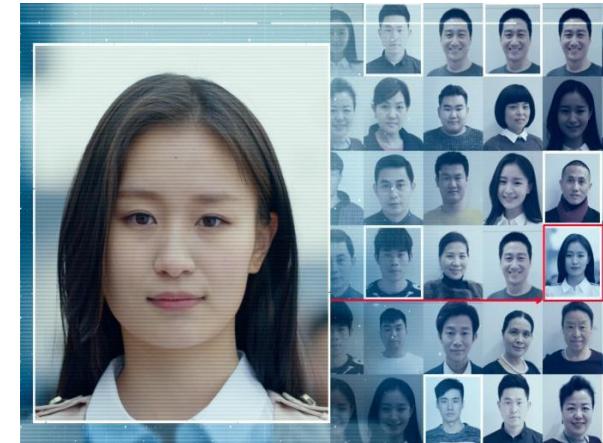
# 1-1 行业应用



## 语音识别

- 距离小于1米，中文字准率97%+
- 支持耳语、长语音、中英文混合及方言

## 计算机视觉

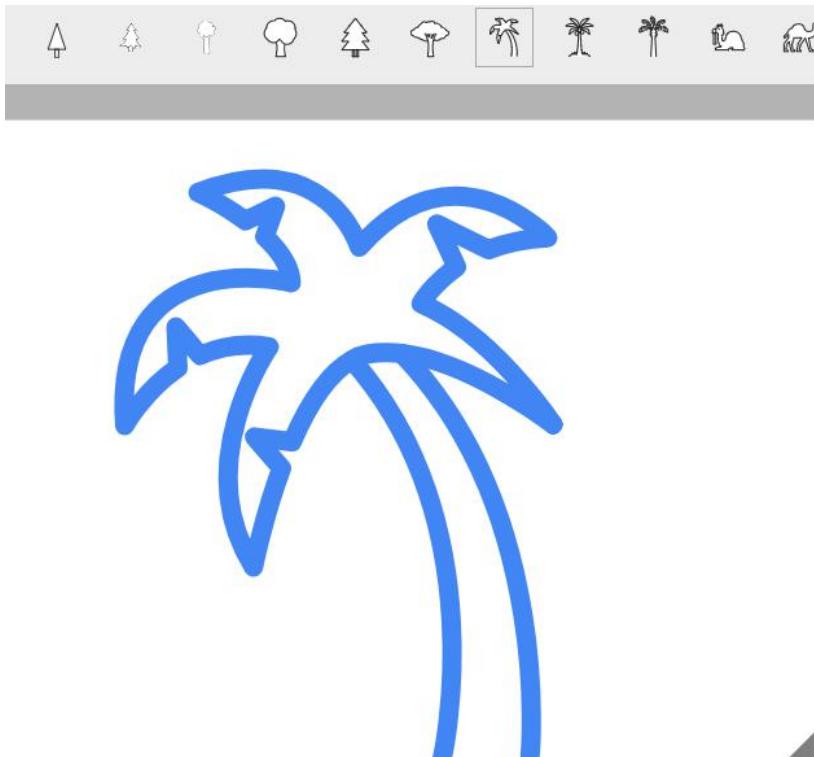


## 用户画像



# 1-1 行业应用

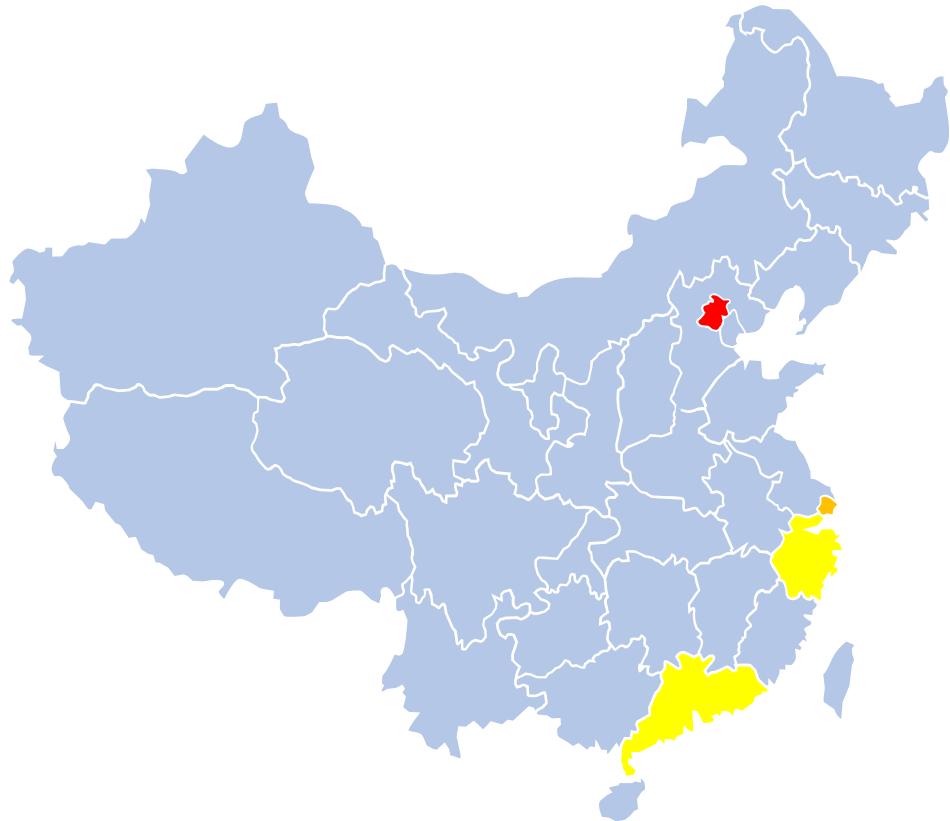
<https://www.autodraw.com/>



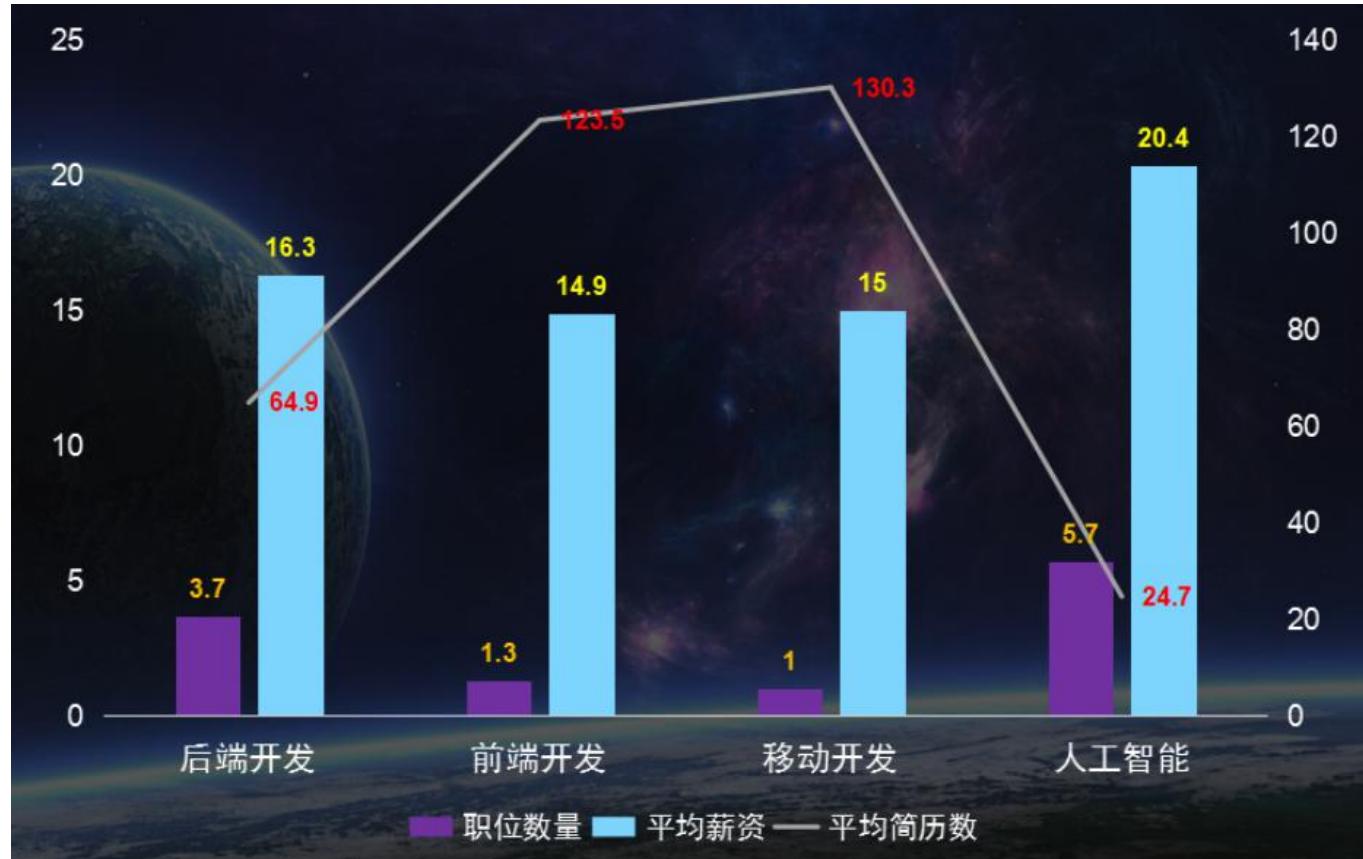
<https://quickdraw.withgoogle.com/#>



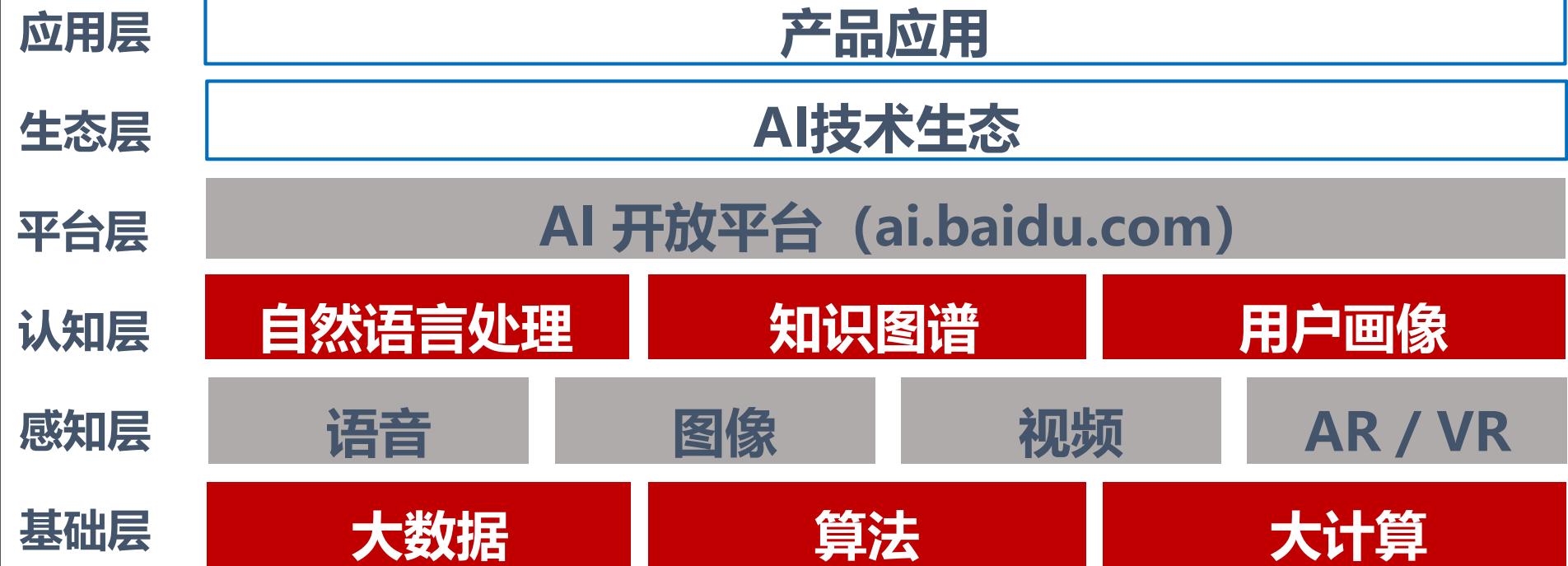
## 1-1 行业应用



# 1-1 行业应用



## 百度人工智能技术布局



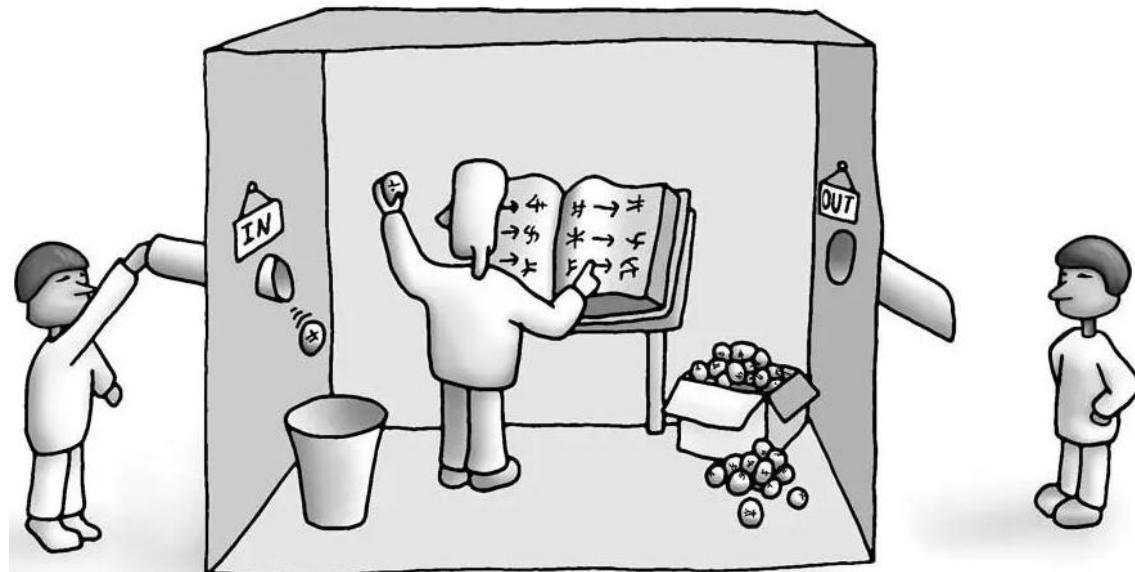
# 1-2 发展历程



## 图灵测试 机器能思考吗?

测试者与被测试者（一个人和一台机器）隔开的情况下，通过一些装置（如键盘）向被测试者随意提问。

多次测试（一般为**5min**之内），如果有超过**30%**的测试者不能确定被测试者是人还是机器，那么这台机器就通过了测试，并被认为具有**人类智能**。



艾伦·麦席森·图灵  
Alan Mathison  
Turing

“计算机科学之父”  
提出“图灵测试”概念  
破解德国的著名密码系统Enigma

# 1-2 发展历程

## Artificial Intelligence (AI), 1956 -



1956年夏 美国达特茅斯学院



J. McCarthy

“人工智能之父”  
图灵奖(1971)



M. Minsky

图灵奖(1969)



C. Shannon

“信息论之父”



H. A. Simon

图灵奖(1975)  
诺贝尔经济学奖(1978)



A. Newell

图灵奖(1975)

达特茅斯会议标志着人工智能这一学科的诞生

# 1-2 发展历程



2006年，会议五十年后，当事人重聚达特茅斯。

左起：摩尔，麦卡锡，明斯基，赛弗里奇，所罗门诺夫



1996.02：卡vs深蓝：4-2

1997.05：卡vs更深的蓝：2.5-3.5

# 1-2 发展历程

20世纪90年至今

1997 “深蓝”

击败国际象棋世界冠军  
卡斯帕罗夫



2012 ImageNet

图像分类连创新高  
超过人类识别能力

2017  
GoogleDeepMind

AlphaGo 围棋战胜世界冠军

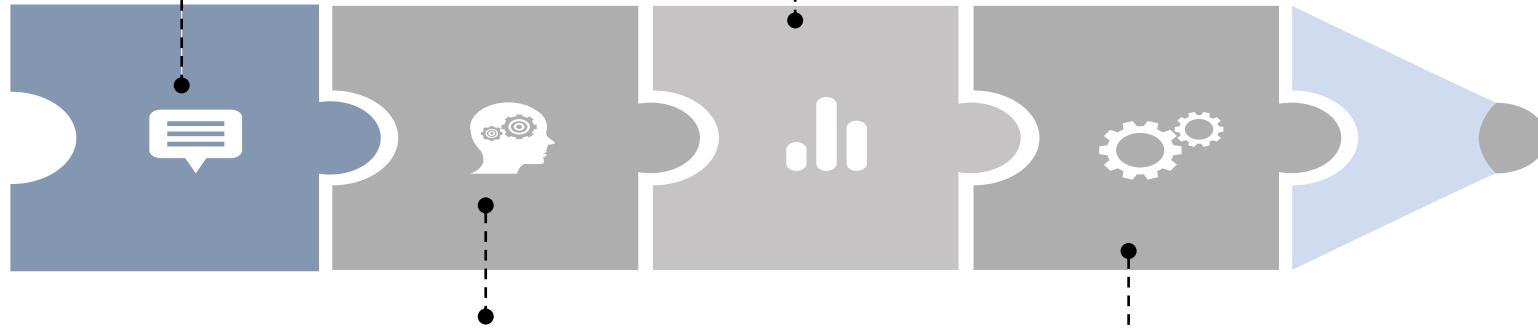
• 2016 发改委

《“互联网+”人工智能实施方案》

# 1-2 发展历程

## 第一个黄金时期 (1956—20世纪70年代)

美国国防部赞助MIT222万美金，之后每年300万美金；  
1963年斯坦福第一个AI实验室成立



## 第二个黄金时期 (1986—20世纪90年代)

日本的第五代计算机项目拨款8.5亿美金；  
1986, 误差反向传播 (BP) 算法诞生；

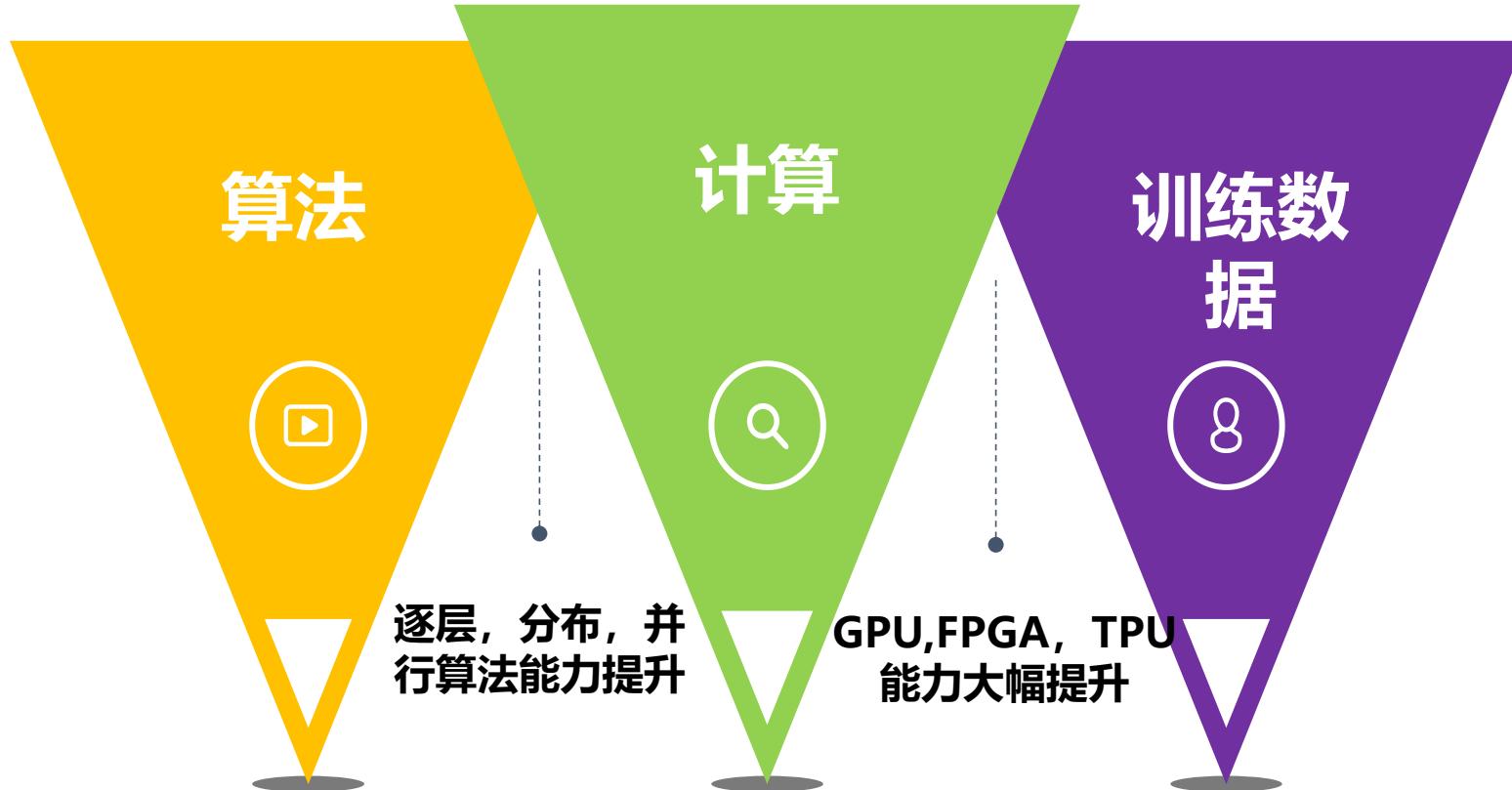
## 第一个低谷时期 (20世纪70年代至90年代初)

James Lighthill:  
《Artificial Intelligence: A general survey》  
专家系统：“知识期”

## 第二个低谷时期 (20世纪70年代至90年代初)

IBM与Apple 普通台式机性能超越“智能计算机”  
1991, 日本的第五代计算机宣

# 1-2 发展历程



# ■ 1-2 发展历程

人工智能的发展经历了3个阶段：

1980年代是**正式成形期**，尚不具备影响力。

1990-2010年代是**蓬勃发展期**，诞生了众多的理论和算法，真正走向了实用。

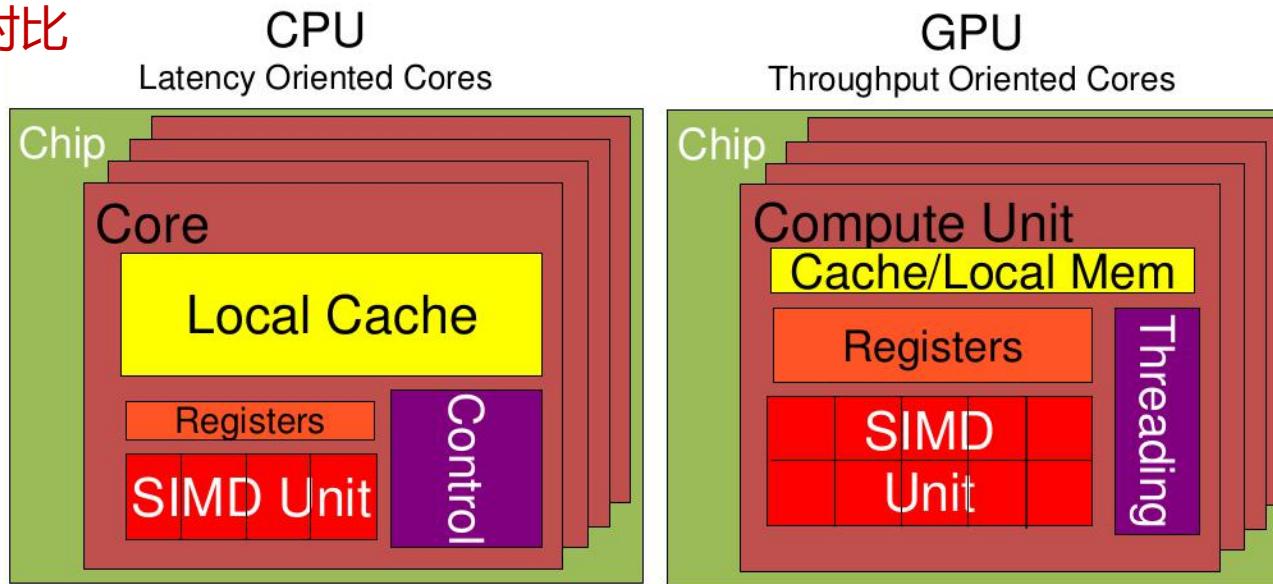
2012年之后是**深度学习期**，深度学习技术诞生并急速发展，较好的解决了现阶段AI的一些重点问题，并带来了产业界的快速发展。

# 总结

总结

# 1-3 CPU与GPU

## GPU与CPU对比



**Cache, local memory:** CPU > GPU

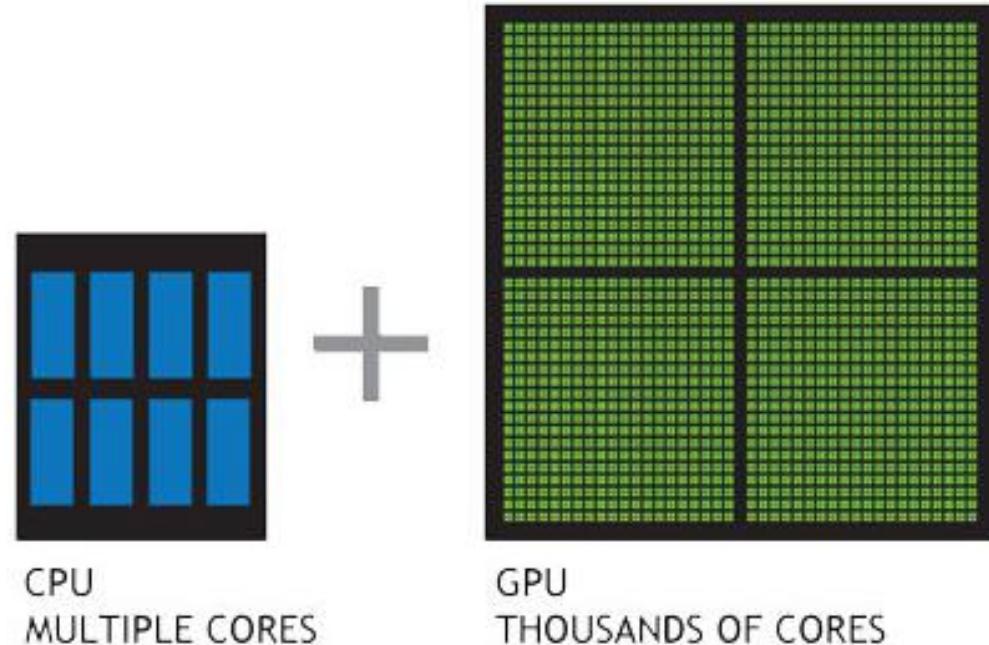
**Threads(线程数):** GPU > CPU

**SIMD Unit(单指令多数据流,以同步方式,在同一时间内执行同一条指令):** GPU > CPU

## GPU与CPU对比

1, GPU 加速计算可以将应用程序**计算密集部分的工作负载转移到 GPU**, 同时仍由**CPU 运行其余程序代码**。从用户的角度来看, **应用程序的运行速度明显加快**.

2,CPU 由专为顺序串行处理而优化的几个核心组成, 而 GPU 则拥有一个由**数以千计的更小、更高效的核心** (专为同时处理多重任务而设计) 组成的大规模并行计算架构.



# 1-3 CPU与GPU

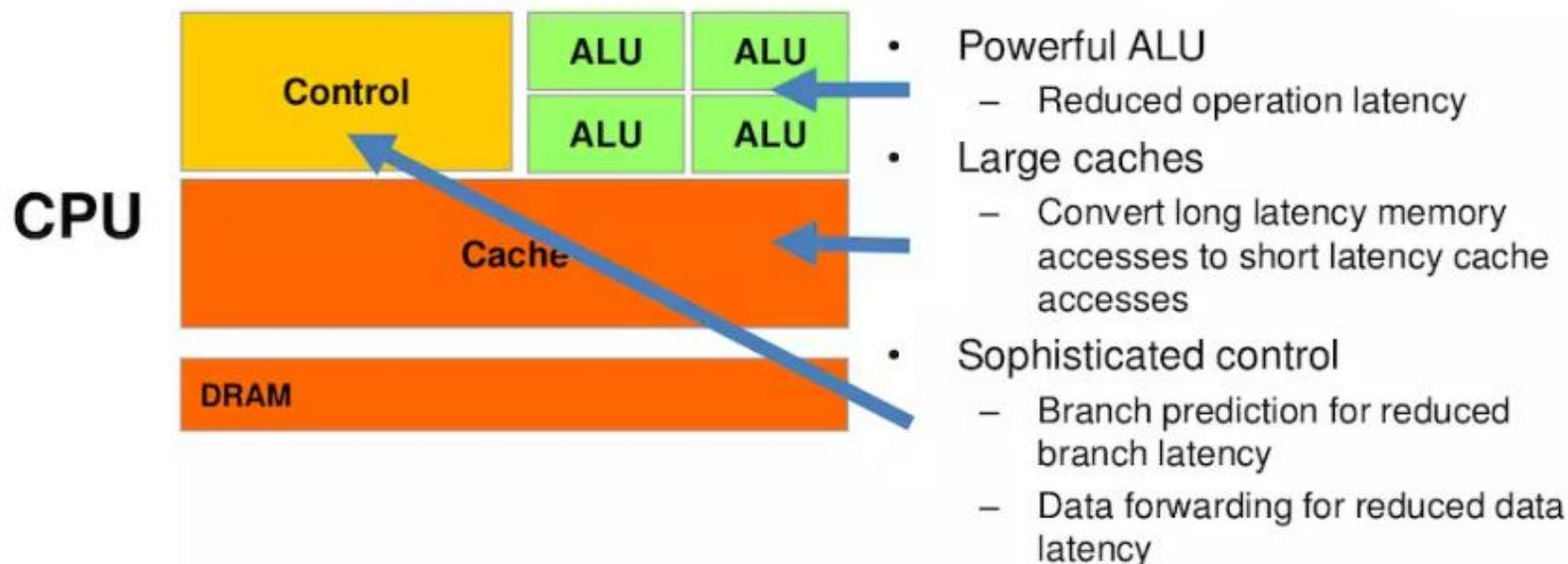
## GPU与CPU对比



3,CPU需要很强的通用性来处理各种不同的数据类型，同时又要逻辑判断又会引入大量的分支跳转和中断的处理。这些都使得CPU的内部结构异常复杂。而GPU面对的则是类型高度统一的、相互无依赖的大规模数据和不需要被打断的纯净的计算环境。GPU采用了数量众多的计算单元和超长的流水线，但只有非常简单的控制逻辑并省去了Cache。而CPU不仅被Cache占据了大量空间，而且还有有复杂的控制逻辑和诸多优化电路，相比之下计算能力只是CPU很小的一部分。

## GPU与CPU对比

### CPUs: Latency Oriented Design



## GPU与CPU对比

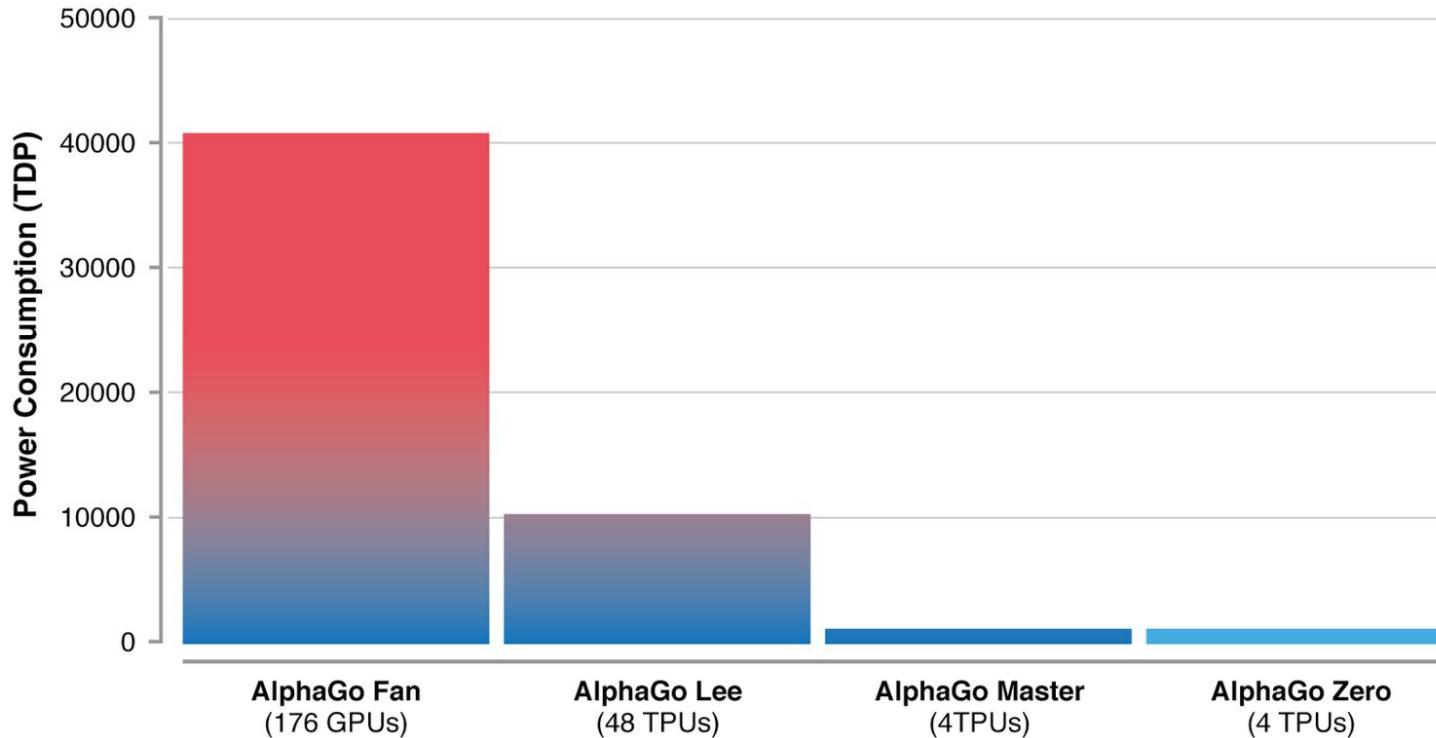
与CPU擅长逻辑控制，串行的运算对比，通用类型数据运算不同，**GPU擅长的是大规模并发计算**，这也正是密码破解等所需要的。所以GPU除了图像处理，也越来越多的参与到计算当中来。

GPU

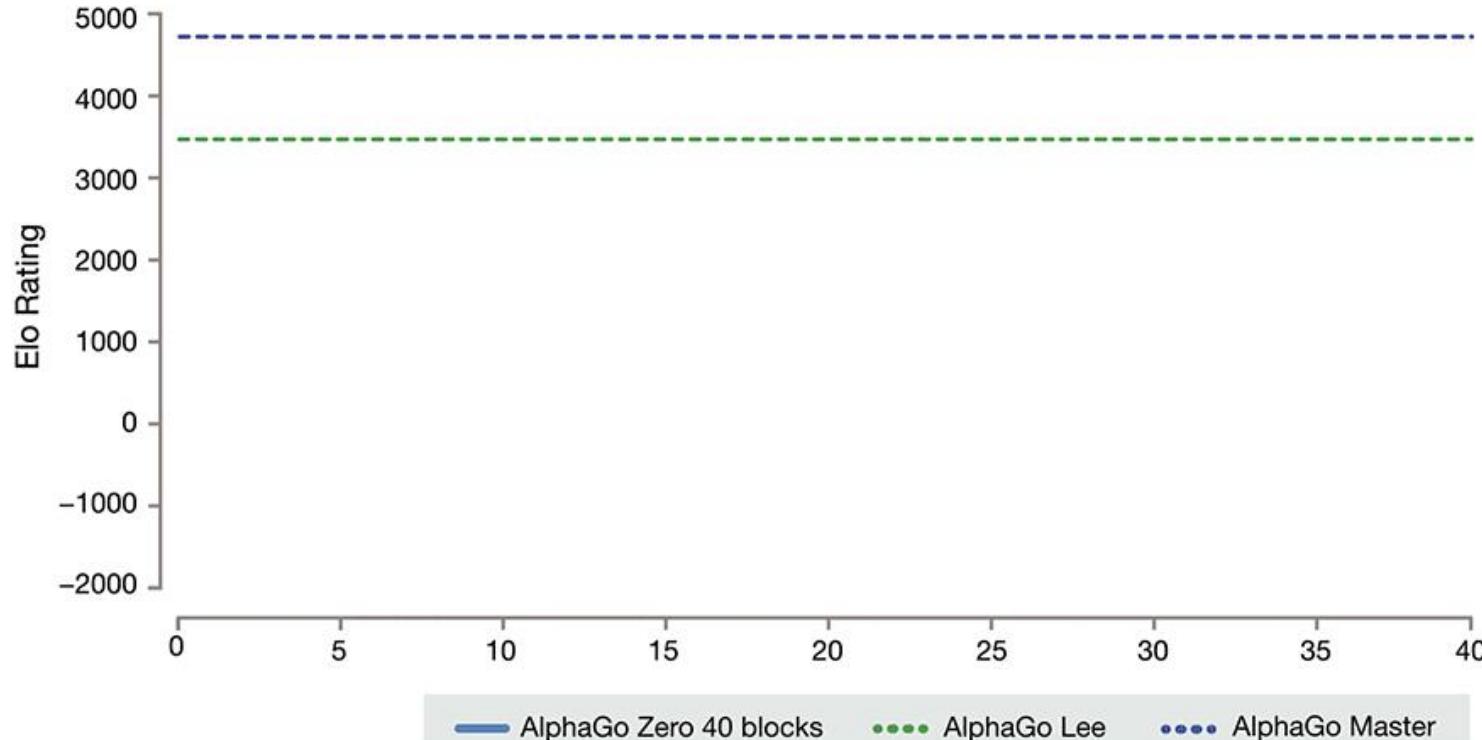
### GPUs: Throughput Oriented Design

- 
- Small caches
    - To boost memory throughput
  - Simple control
    - No branch prediction
    - No data forwarding
  - Energy efficient ALUs
    - Many, long latency but heavily pipelined for high throughput
  - Require massive number of threads to tolerate latencies

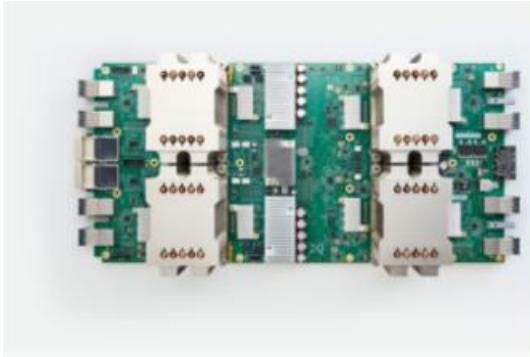
# 1-3 CPU与GPU



# 1-3 CPU与GPU



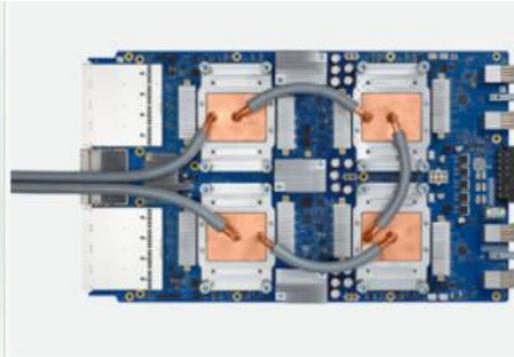
# 1-3 CPU与GPU



Cloud TPU v2

每秒 180 万亿次浮点运算

64 GB 高带宽内存 (HBM)



Cloud TPU v3 测试版

每秒 420 万亿次浮点运算

128 GB HBM



Cloud TPU v2 Pod Alpha 版

每秒 11.5 千万亿次浮点运算

4 TB HBM

二维环形网状网络

<https://cloud.google.com/tpu/?hl=zh-cn>

<https://buzzorange.com/techorange/2017/09/27/what-intel-google-nvidia-microsoft-do-for-ai-chips/>

## 什么类型的程序适合在GPU上运行？

(1) **计算密集型的程序**。所谓计算密集型(Compute-intensive)的程序，就是其大部分运行时间花在了寄存器运算上，寄存器的速度和处理器的速度相当，从寄存器读写数据几乎没有延时。可以做一下对比，读内存的延迟大概是几百个时钟周期；读硬盘的速度就不说了，即便是SSD, 也实在是太慢了。

(2) **易于并行的程序**。他有成百上千个核，每一个核在同一时间最好能做同样的事情。



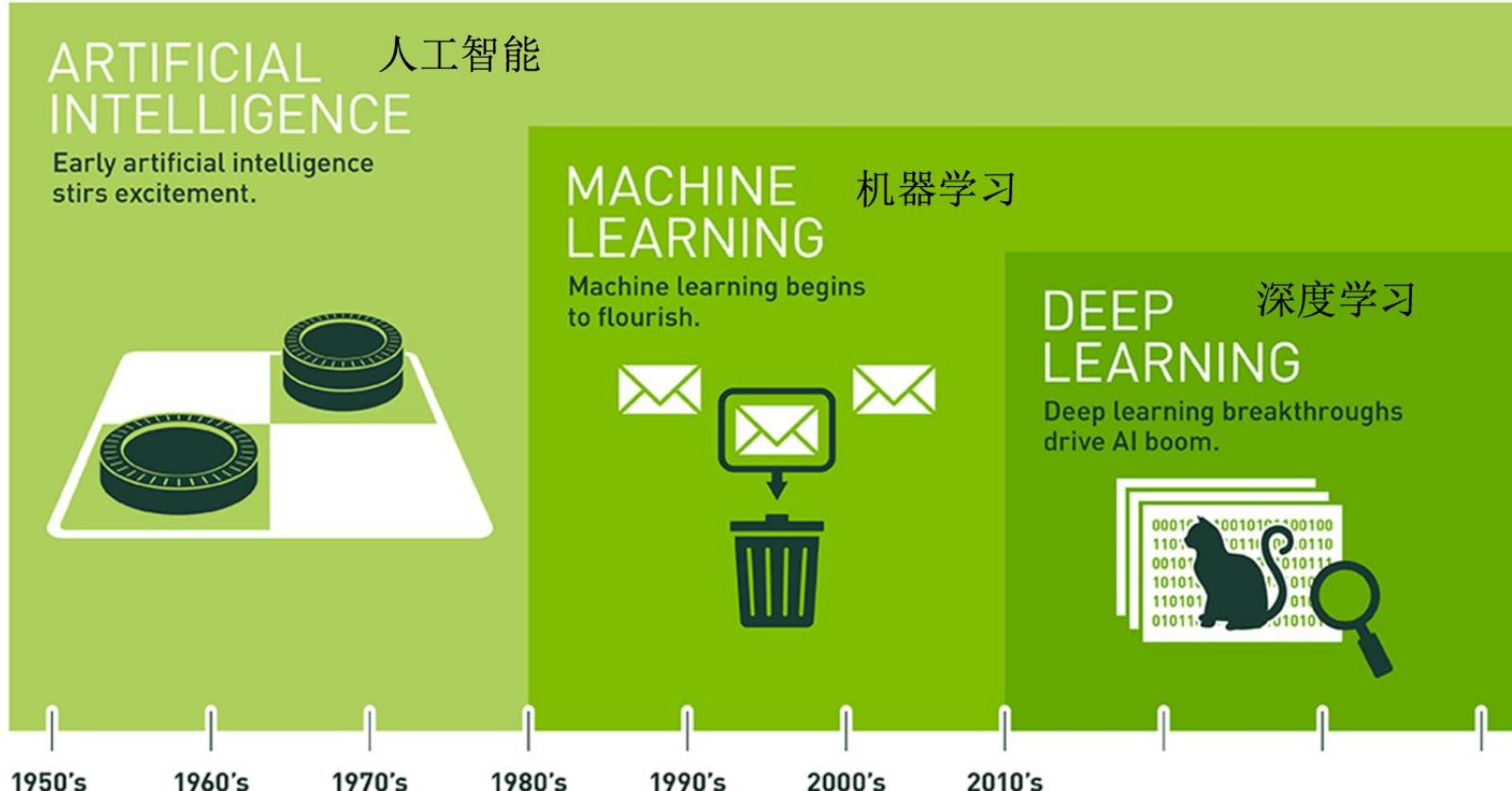
总结

# 目录

## Contents

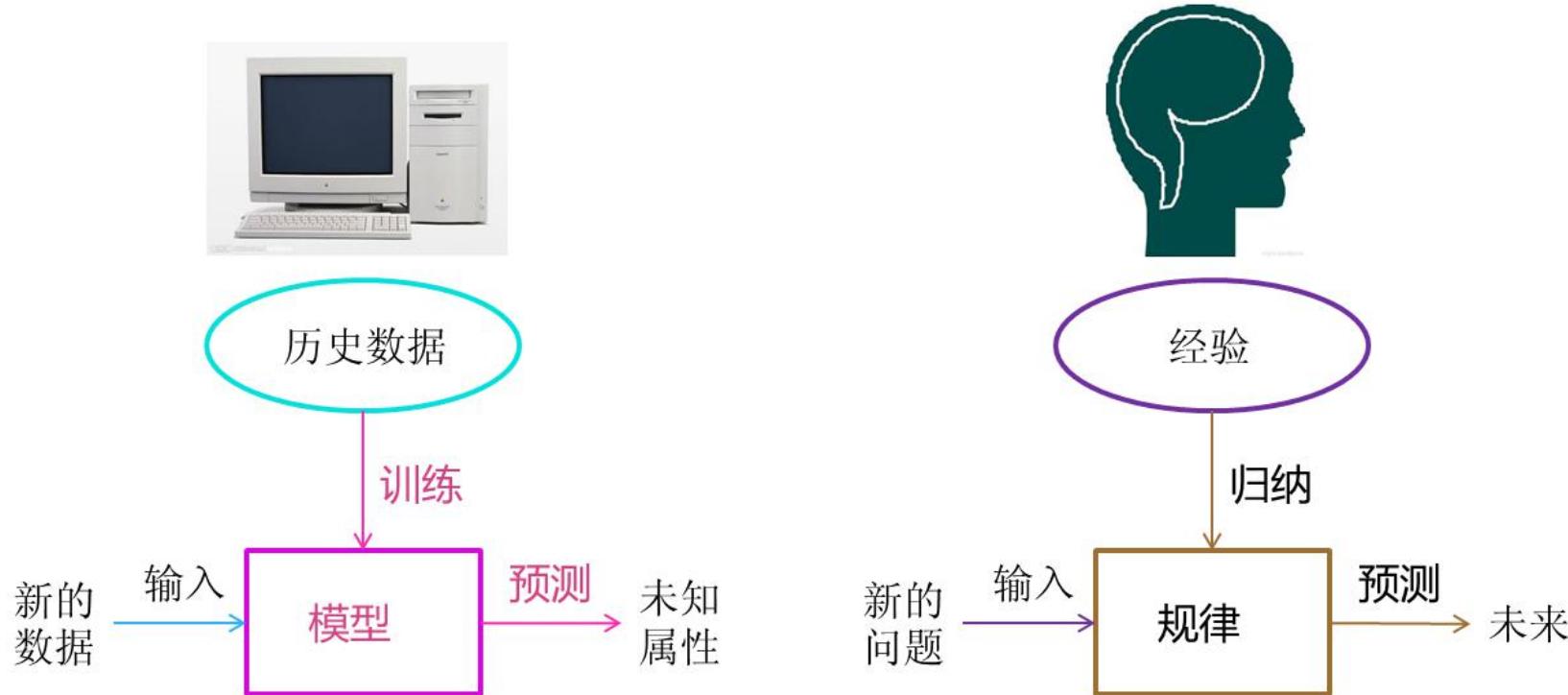
- ◆ 应用场景与发展历程
- ◆ 人工智能主要分支
- ◆ 机器学习工作流程
- ◆ 机器学习算法分类
- ◆ 模型评估

# 2-1 人工智能主要分支

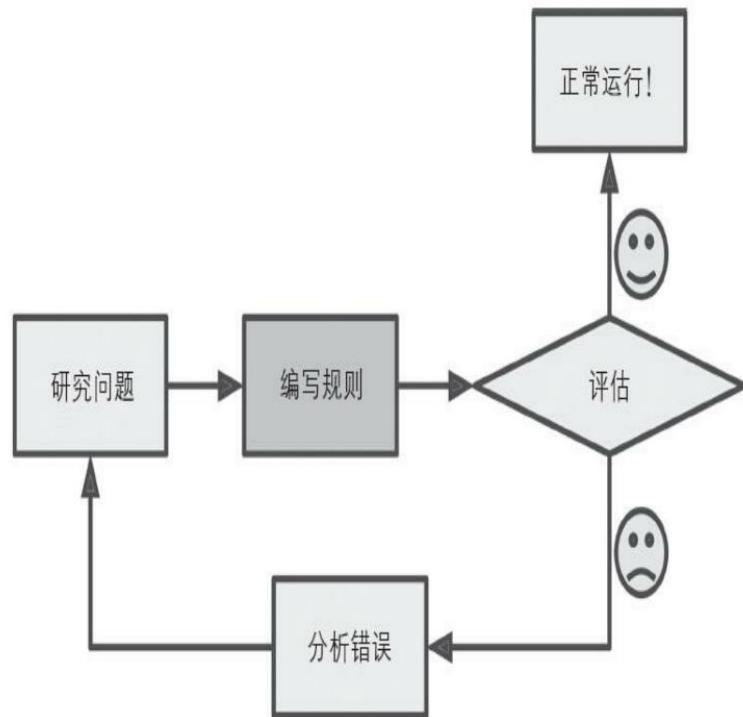


## 2-2 机器学习

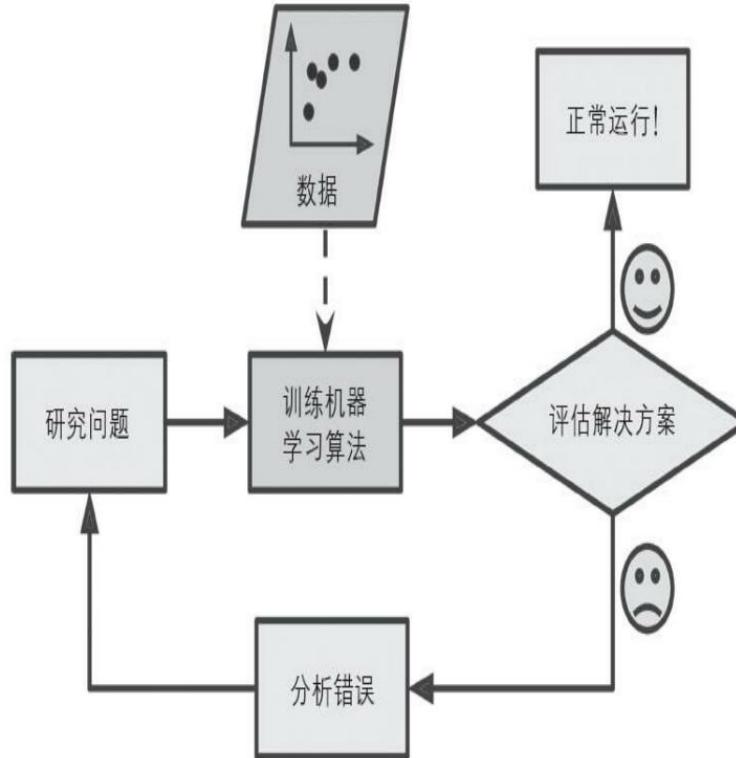
**机器学习**是从数据中自动分析获得模型，并利用模型对未知数据进行预测



## 2-2 机器学习



传统编程方法

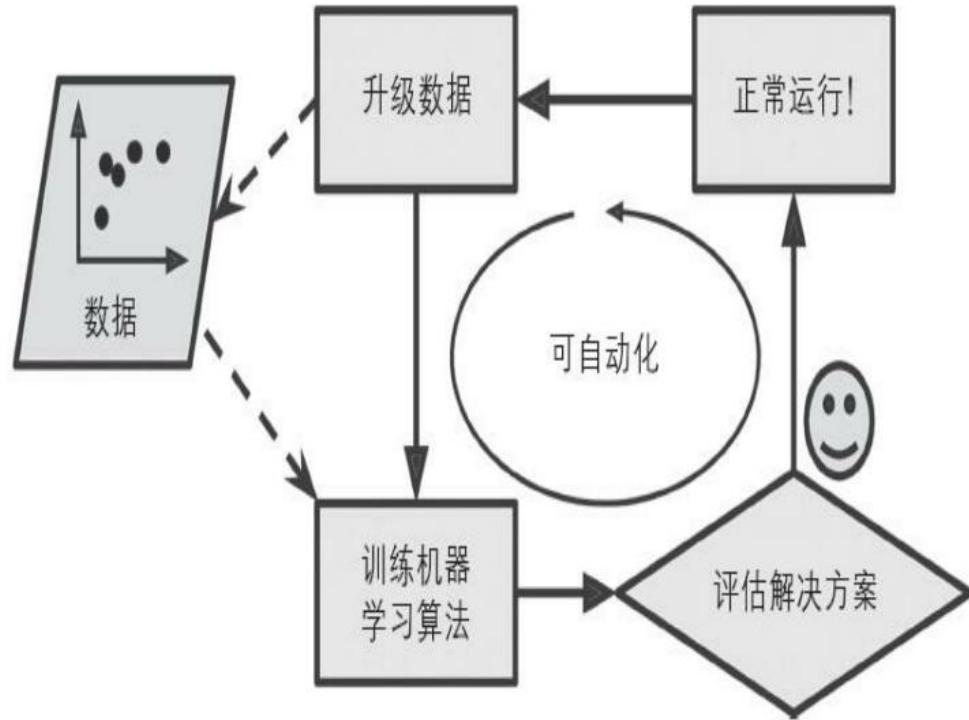


机器学习解决方法

## 2-2 机器学习

如果垃圾邮件发送者注意到所有包含“4U”字眼的电子邮件都被阻止，他们很可能会开始写成“For U”。使用**传统编程技术的垃圾邮件过滤器需要人为更新标记“For U”的邮件**。而如果垃圾邮件发送者持续围绕着你的过滤器来变更策略，那你就将需要永无止境地续写新的过滤规则。

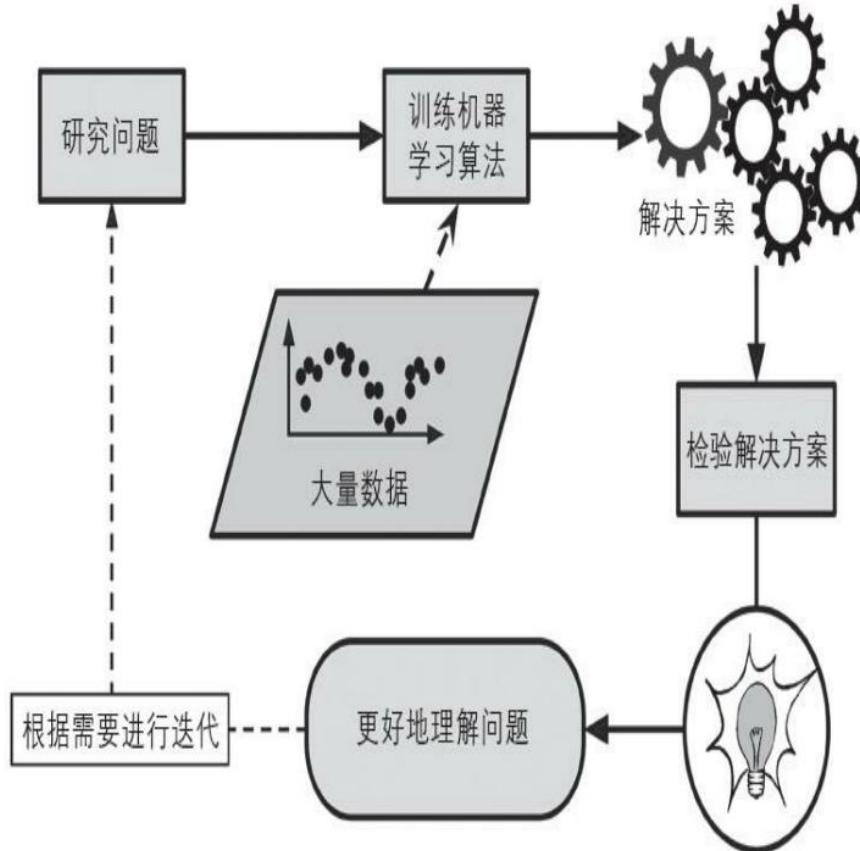
相比之下，基于**机器学习技术**的垃圾邮件过滤器可以**自动**注意到“For U”开始在用户标记的垃圾邮件中出现得异常频繁，**人为干预很少，就会开始自动标记它们**。



## 2-2 机器学习

机器学习还可以帮助人类学习：

一旦垃圾邮件过滤器经过了足够的训练，人们可以很轻松地检视它，查看它认为的可以作为垃圾邮件最佳预判因子的单词及单词组合的列表。有时候，这可能会揭示出一些人类未曾意识到的关联性或是新趋势，从而帮助我们更好地理解问题。



机器学习是从人工智能中产生的一个重要学科分支，是实现智能化的关键  
经典定义：利用经验改善系统自身的性能



经验 → 数据



随着该领域的发展，目前主要研究**智能数据分析**的理论和算法，并已成为智能数据分析技术的源泉之一

图灵奖连续授予在该方面取得突出成就的学者



**Leslie Valiant**  
**(1949 - )**  
**(Harvard Univ.)**

“计算学习理论”奠基人

2010  
年度



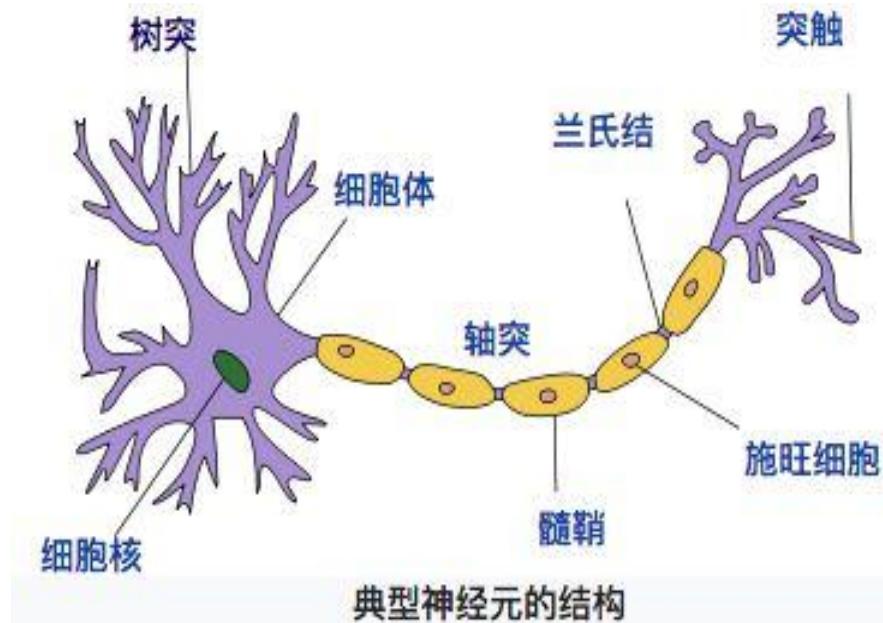
**Judea Pearl**  
**(1936 - )**  
**(UCLA)**

“图模型学习方法”先驱

2011  
年度

## 2-3 深度学习

深度学习通过组合低层特征形成更加抽象的高层表示属性类别，以发现数据的分布式特征表示。



# 2-3 深度学习



**Yann LeCun**

卷积网络之父

[PDF] Backpropagation Applied to Handwritten Zip Code Recognition

[yann.lecun.com/exdb/pubs/pdf/lecun-89e.pdf](http://yann.lecun.com/exdb/pubs/pdf/lecun-89e.pdf) • 翻译此页

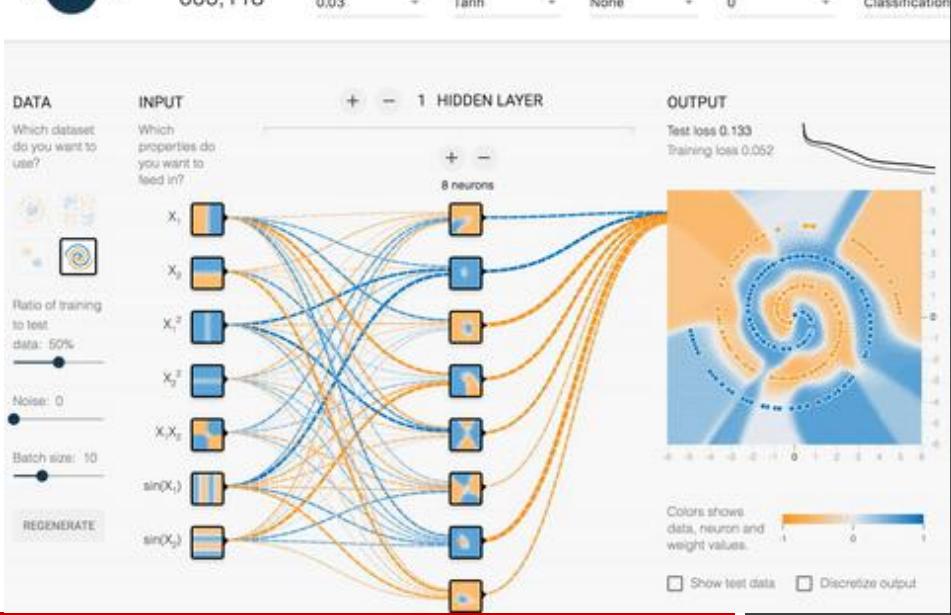
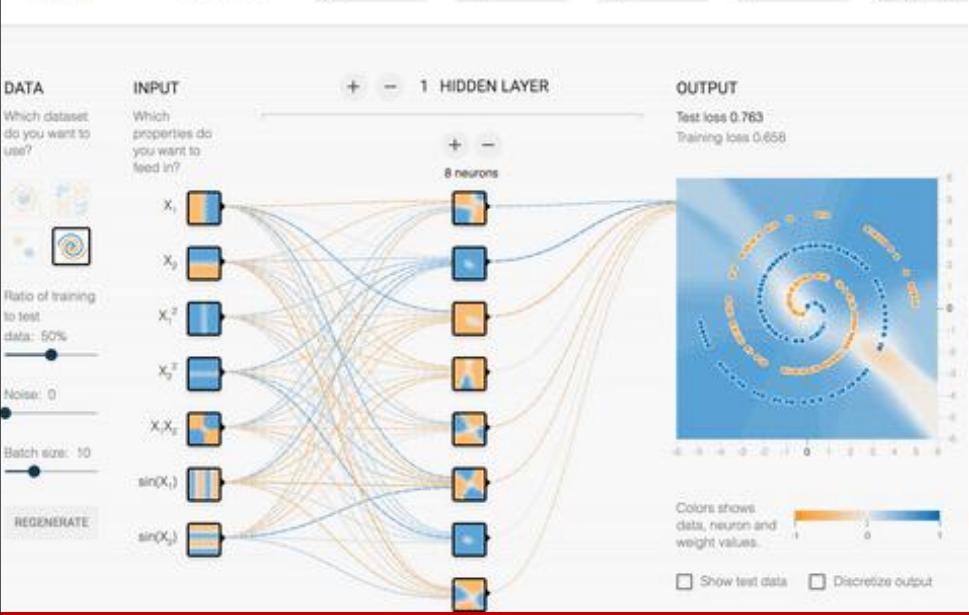
作者: Y LeCun - 1989 - 被引用次数: 3168 - 相关文章

Backpropagation Applied to Handwritten Zip Code Recognition, Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, AT&T Bell Laboratories Holmdel, NJ 07733 USA.  
The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain.

# 2-3 深度学习

## 深度学习平台Tensorflow游乐场

<http://playground.tensorflow.org>



## 第1层：负责识别颜色及简单纹理



## 2-3 深度学习

第2层：识别更加细化的纹理，布纹，刻纹，叶纹等



第3层：负责感受黑夜里的黄色烛光，高光，萤火，鸡蛋黄色等。



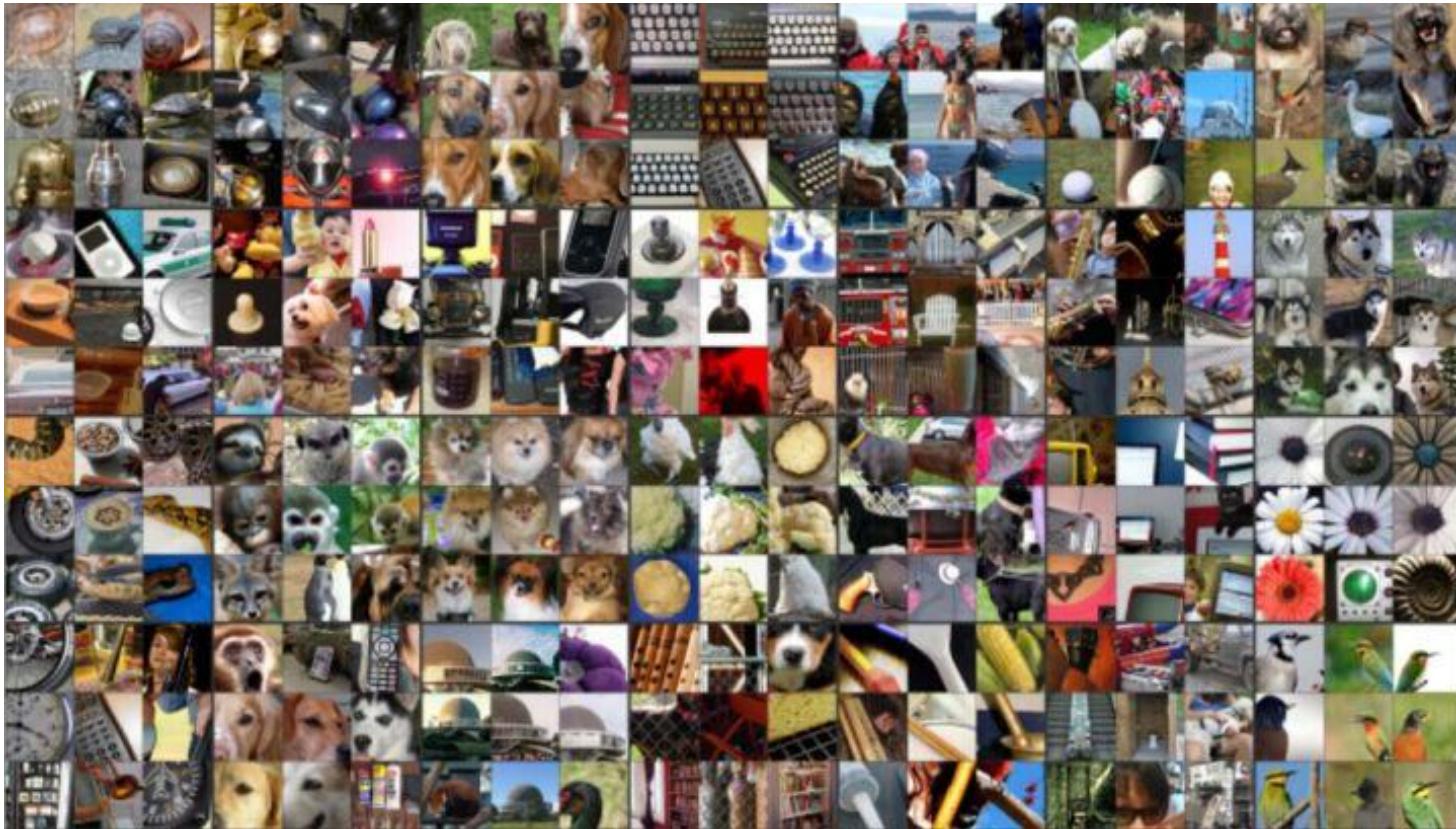
# 2-3 深度学习

第4层：识别萌狗的脸，宠物形貌，圆柱体事物，七星瓢虫等的存在



# 2-3 深度学习

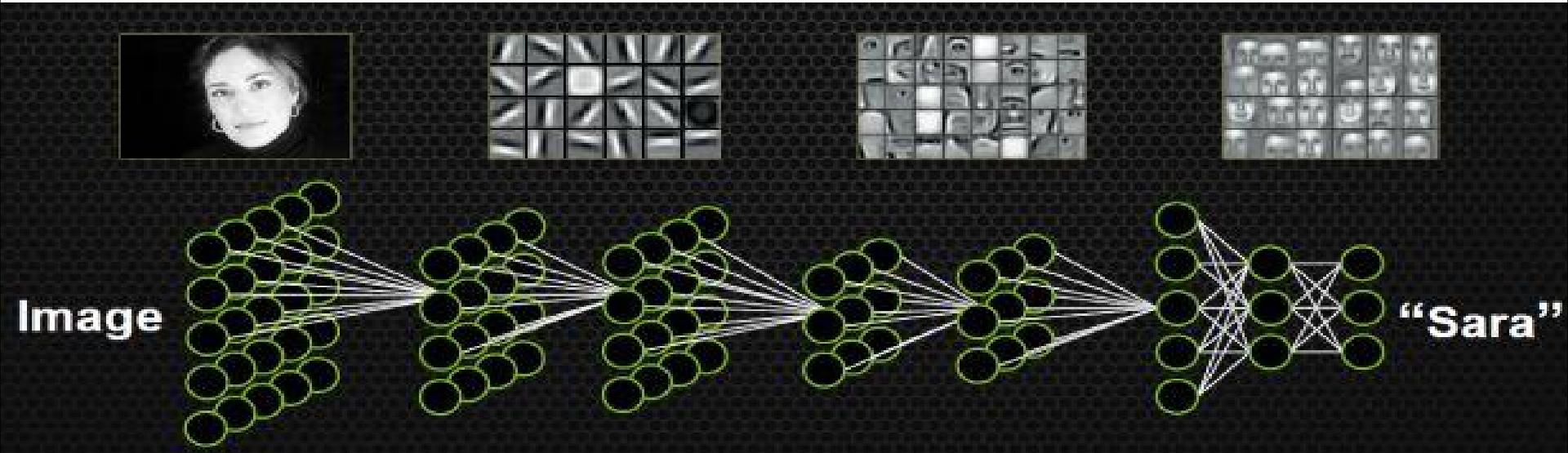
**第5层：负责识别花，黑眼圈动物，鸟，键盘，原型屋顶等**



## 2-3 深度学习

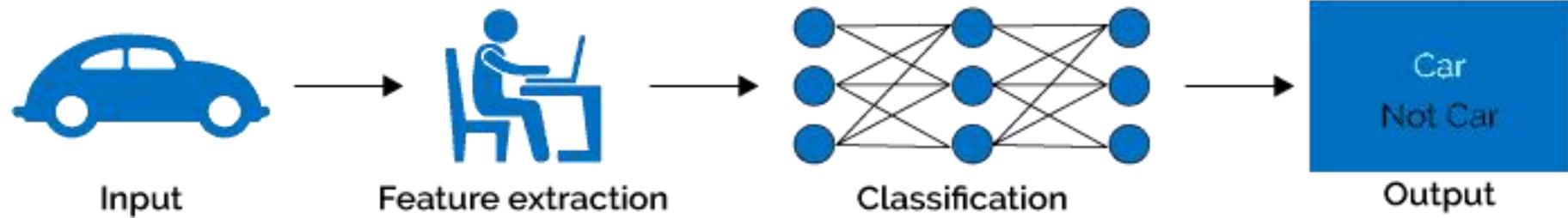
增加层数：通过更抽象的概念识别物体，器官层，分子层，原子层

增加结点数：增加同一层物质的种类

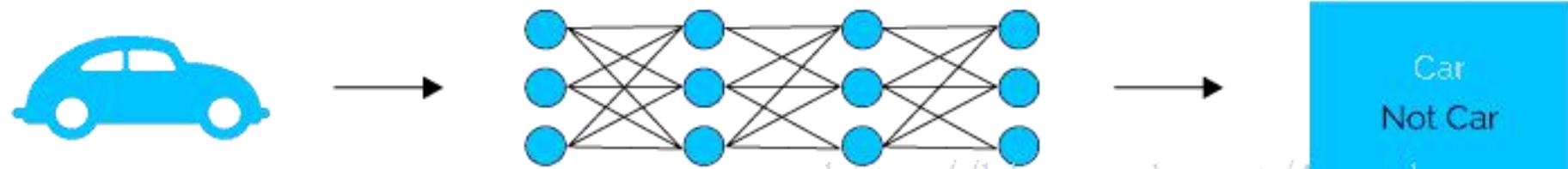


## 机器学习 VS 深度学习

### Machine Learning



### Deep Learning



# 思考

思考

请尝试着总结出人工智能的发展历史及主要应用场景：

## 思考

请尝试着总结出人工智能的发展历史及主要应用场景：

1980年代是**正式成形期**，尚不具备影响力。

1990-2010年代是**蓬勃发展期**，诞生了众多的理论和算法，真正走向了实用。

2012年之后是**深度学习期**，深度学习技术诞生并急速发展，较好的解决了现阶段AI的一些重点问题，并带来了产业界的快速发展。

# 目录

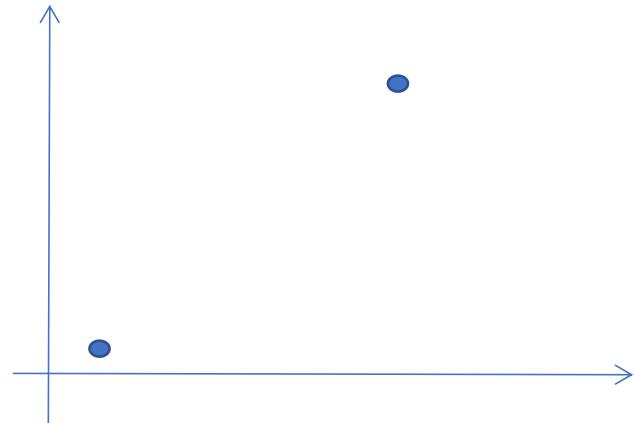
## Contents

- ◆ 应用场景与发展历程
- ◆ 人工智能主要分支
- ◆ 机器学习工作流程
- ◆ 机器学习算法分类
- ◆ 模型评估

# 3-1 机器学习工作流程



一辆红色的奥迪汽车进行车速测试，第一次测试，1小时行驶了60km，第二次测试，1分钟行驶了1200m，那么，这个汽车正常行驶，3小时你估计能开多远？



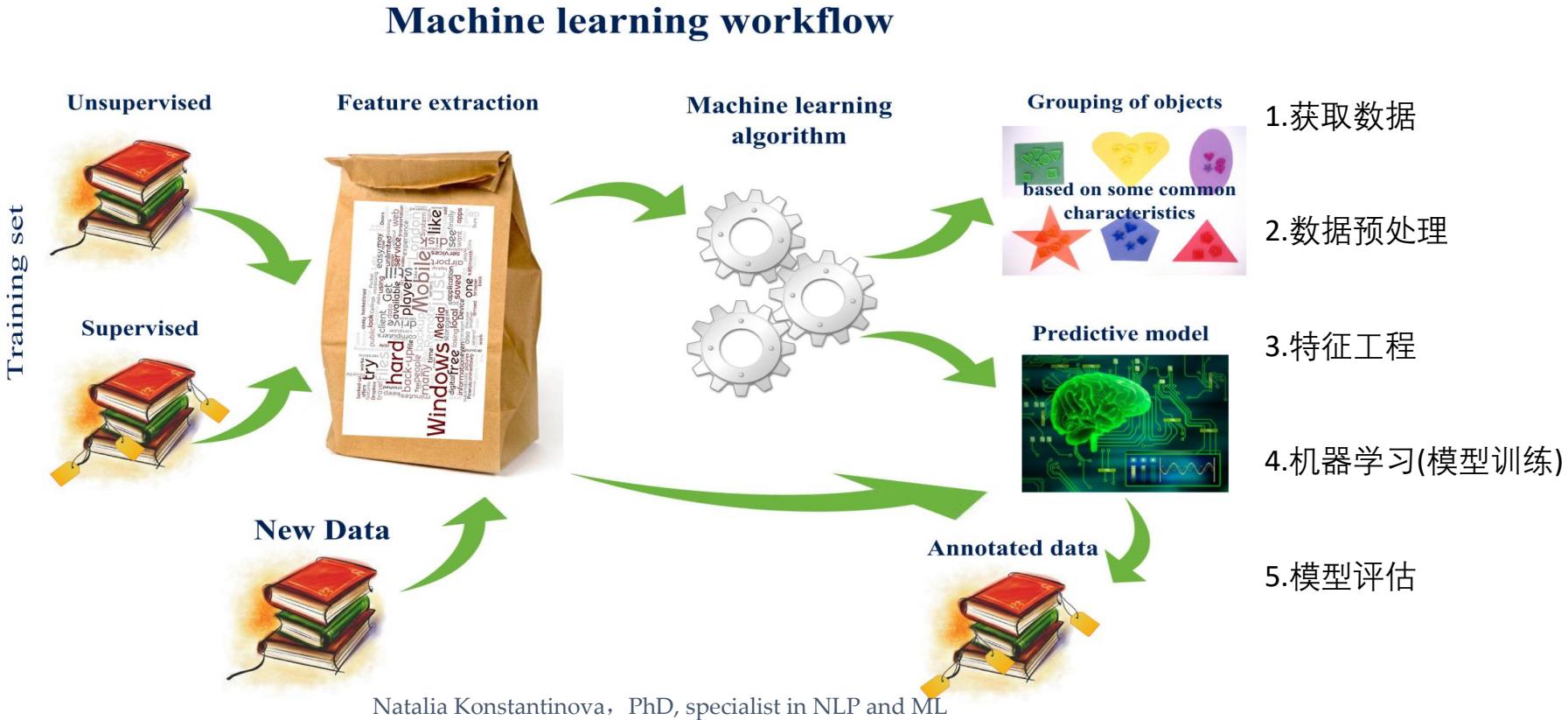
# 3-1 机器学习工作流程



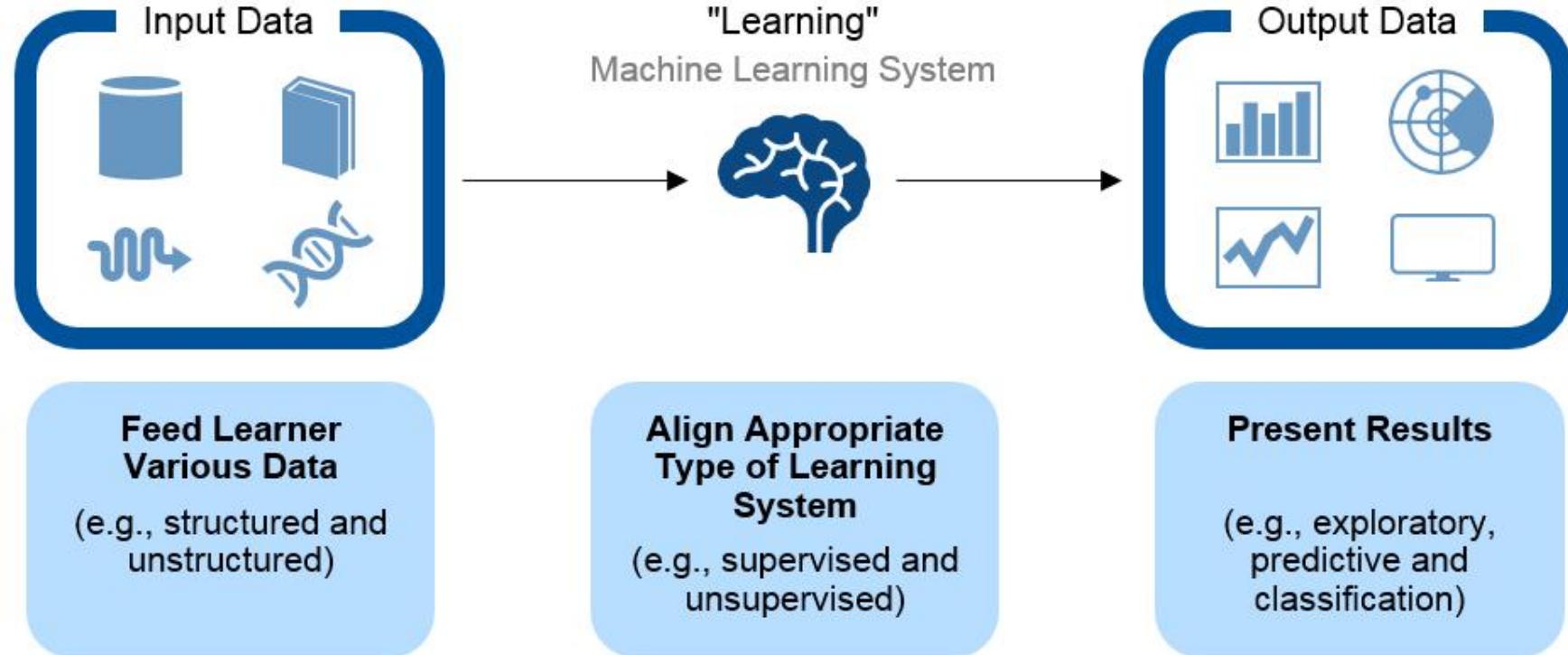
一辆红色的奥迪汽车进行车速测试，第一次测试，1小时行驶了60km 第二次测试，1分钟行驶了1200m 那么，这个汽车正常行驶，3小时你估计能开多远？

参考答案：198km

# 3-1 机器学习工作流程



# 3-2 Azure机器学习实验



# 3-2 Azure机器学习实验

Microsoft Azure Machine Learning Studio

experiments

MY EXPERIMENTS SAMPLES

NAME	AUTHOR	STATUS	LAST EDITED	PROJECT
Experiment created on 2019/2... 772678260	772678260	Finished	2/27/2019 11:47:55 PM	None
Experiment created on 2019/2... 772678260	772678260	Finished	2/25/2019 7:54:28 PM	None
Experiment created on 2019/2... 772678260	772678260	Finished	2/14/2019 2:06:33 PM	None
Experiment created on 2019/2... 772678260	772678260	Draft	2/14/2019 12:42:14 PM	None
Experiment created on 2019/2... 772678260	772678260	Finished	2/14/2019 12:37:00 PM	None
CAROL	772678260	Finished	2/13/2019 6:43:17 PM	None

adult.data.csv

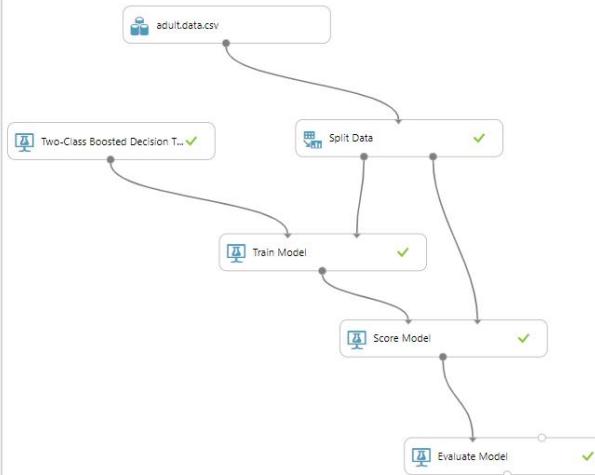
Two-Class Boosted Decision Tree

Split Data

Train Model

Score Model

Evaluate Model



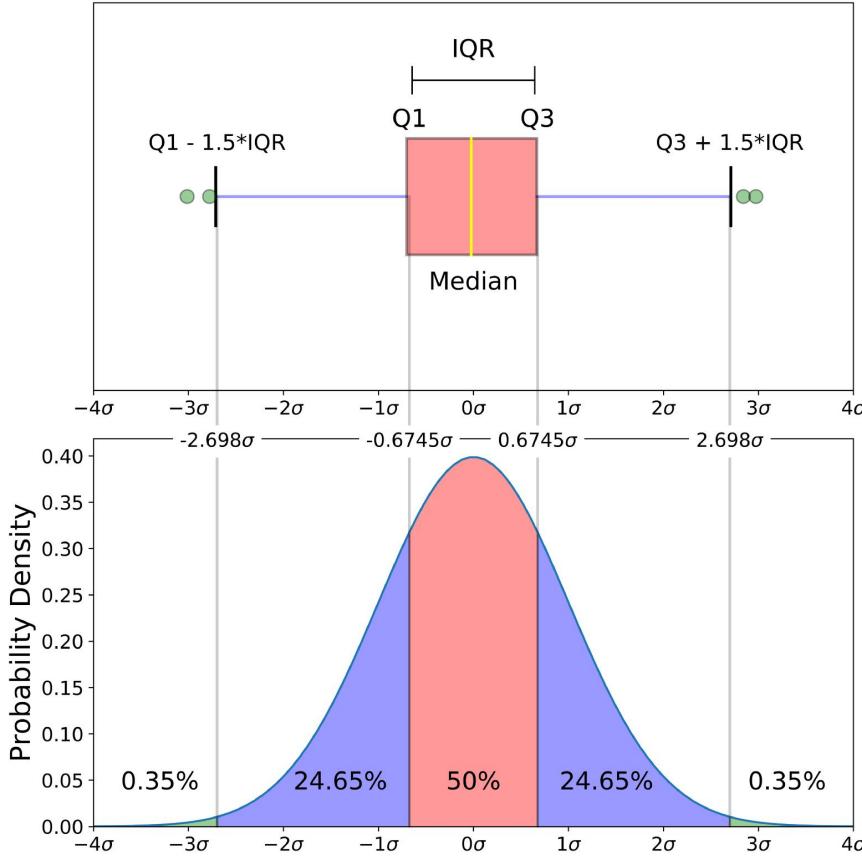
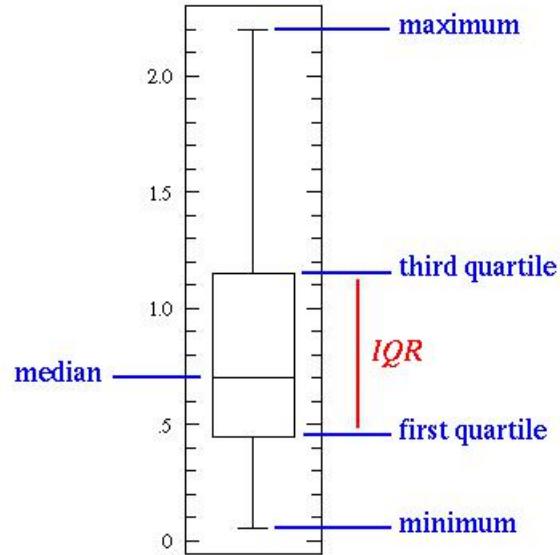
```
graph TD; A[adult.data.csv] --> B[Two-Class Boosted Decision Tree]; B --> C[Split Data]; C --> D[Train Model]; D --> E[Score Model]; E --> F[Evaluate Model];
```

<https://studio.azureml.net/>

## 3-2 Azure机器学习实验

### 箱形图 (备注)

箱形图 (Box-plot) 又称为盒须图、盒式图或箱线图，是一种用作显示一组数据分散情况资料的统计图。



## 3-2 Azure机器学习实验

### 箱形图 (备注)

12位商学院毕业生月起薪的样本在这里按升序重复如下：

2710 2755 2850 | 2880 2880 2890 | 2920 2940 2950 | 3050 3130 3325

$Q_1 = 2865$     $Q_2 = 2905$ (中位数)    $Q_3 = 3000$

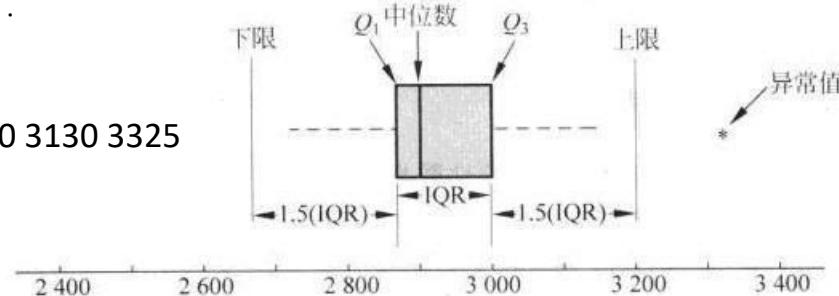


图 1 起薪数据带有上下限制线的箱线图

中位数是2905，第一个四分位数 $Q_1 = 2865$ ，第三个四分位数 $Q_3 = 3000$ 。检查这些数据，最小值为2710，最大值为3325。因此，薪水数据的五数概括数据为2710、2865、2905、3000、3325。大约1 / 4或25%的观察值在五数概括的相邻两个数字之间。箱线图是在五数概括的基础上对数据进行描述的图形方法。绘制箱线图的关键是计算中位数、四分位数 $Q_1$ 和 $Q_3$ ，四分位数全距 $IQR = Q_3 - Q_1$ 。

## 3-2 Azure机器学习实验

### 箱形图 (备注)

绘制箱线图的步骤如下：

1. 画一只箱子，箱子两端分别位于第一个和第三个四分位数上。

对于薪水数据来说， $Q_1 = 2865$  以及  $Q_3 = 3000$ 。

2. 在箱子中位数(薪水数据是2905)的位置画一条垂直线。

3. 用四分位数全距  $IQR = Q_3 - Q_1$ ，确定限制线的位置。箱线图的上、下限制线分别在比  $Q_1$  低  $1.5(IQR)$  和比  $Q_3$  高  $1.5(IQR)$  的位置上。对于薪水数据来说， $IQR = Q_3 - Q_1 = 3000 - 2865 = 135$ 。因此，限制线的位置在  $2865 - 1.5(135) = 2662.5$  和  $3000 + 1.5(135) = 3202.5$  处。两条限制线以外的数据可以认为是异常值。

4. 图1中的虚线称为触须线。触须线从箱子两端开始绘制，直至第3步中计算的限制线内的最小值和最大值。因此，薪水数据的触须线分别在 2710 和 3130 处结束。

5. 最后，每个异常值的位置用星号“\*”表示，看到一个异常值，即 3325。

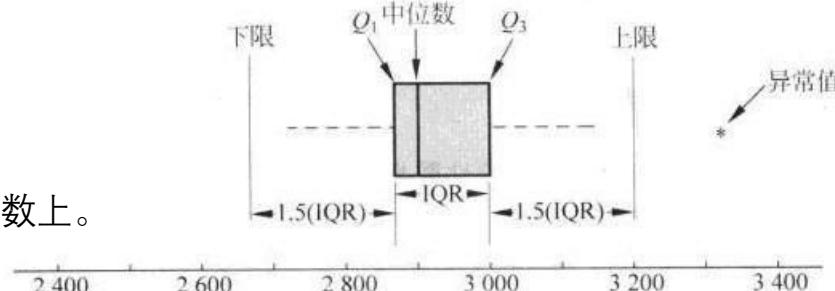


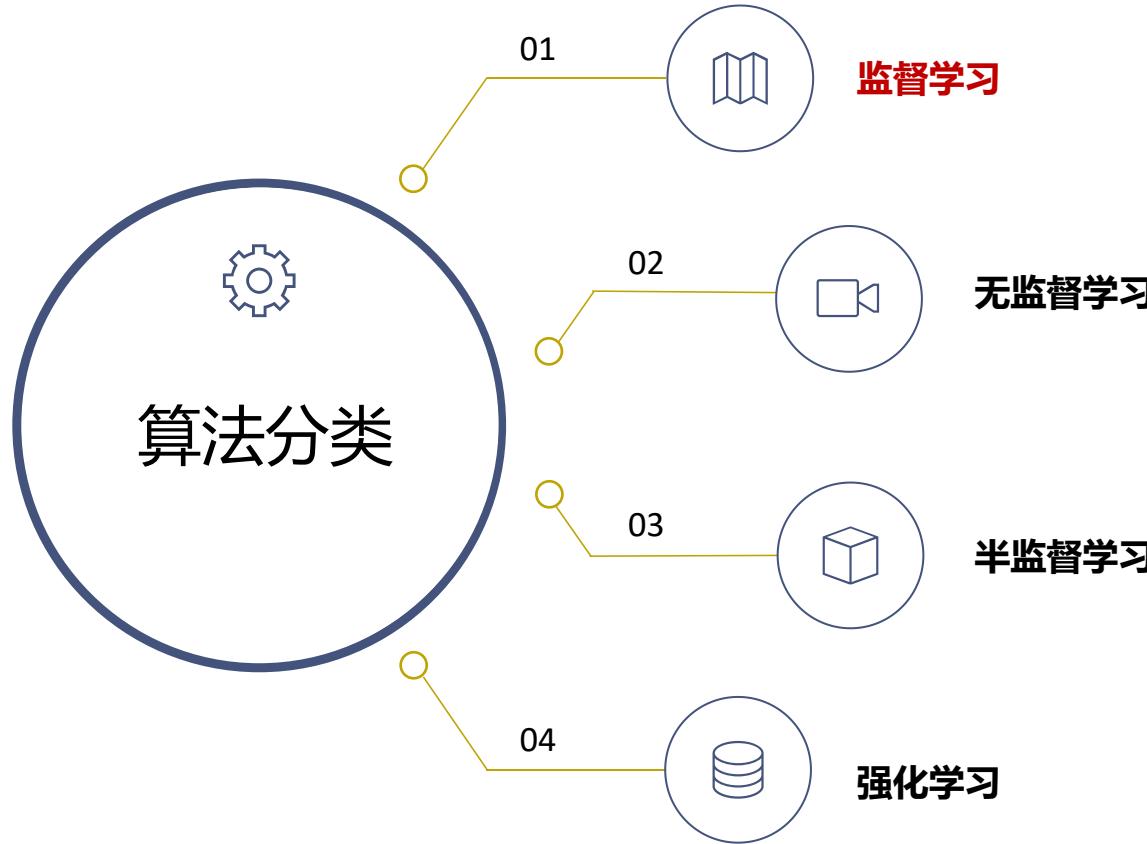
图 1. 起薪数据带有上下限制线的箱线图

# 目录

## Contents

- ◆ 应用场景与发展历程
- ◆ 人工智能主要分支
- ◆ 机器学习工作流程
- ◆ 机器学习算法分类
- ◆ 模型评估

# 4 机器学习算法分类



# 4 机器学习算法分类

样本

标签

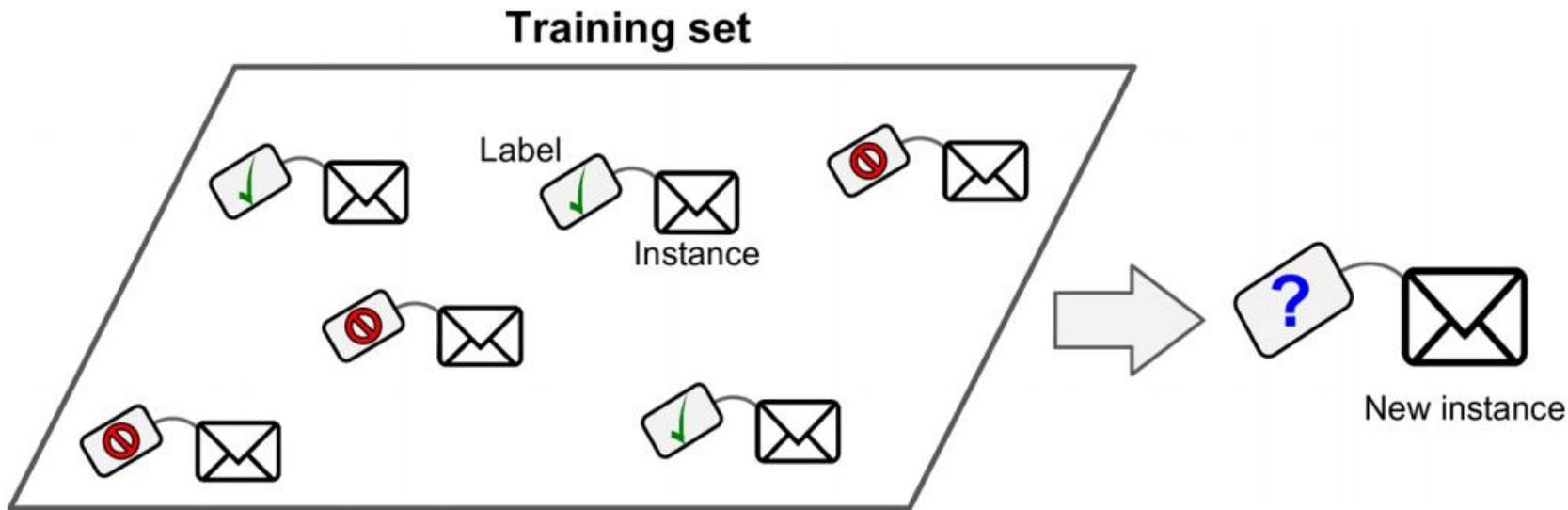
训练集

测试集

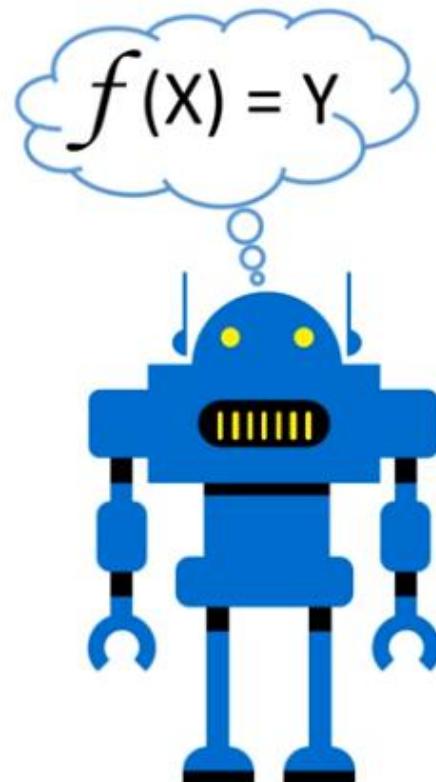
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavano	Proanth	Color.int	Hue	OD	Proline
2	1	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065
3	1	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050
4	1	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185
5	1	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480
6	2	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735
7	2	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
8	2	14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290
9	3	14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295
10	3	14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045
11	3	13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045

## ■ 4-1 监督学习

**有监督**定义：输入数据是由输入特征值和目标值所组成，即输入的训练数据为**有标签的**。



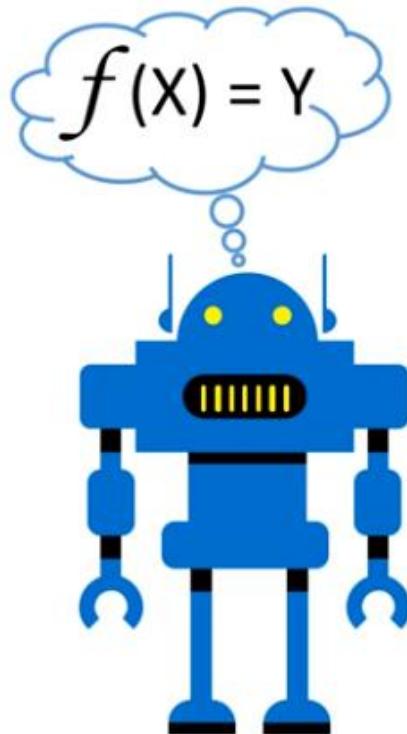
## ■ 4-1 监督学习



[ $X_1, X_2, X_3, Y$ ]  
[ $X_1, X_2, X_3, Y$ ]  
[ $X_1, X_2, X_3, Y$ ]

## ■ 4-1 监督学习

[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]



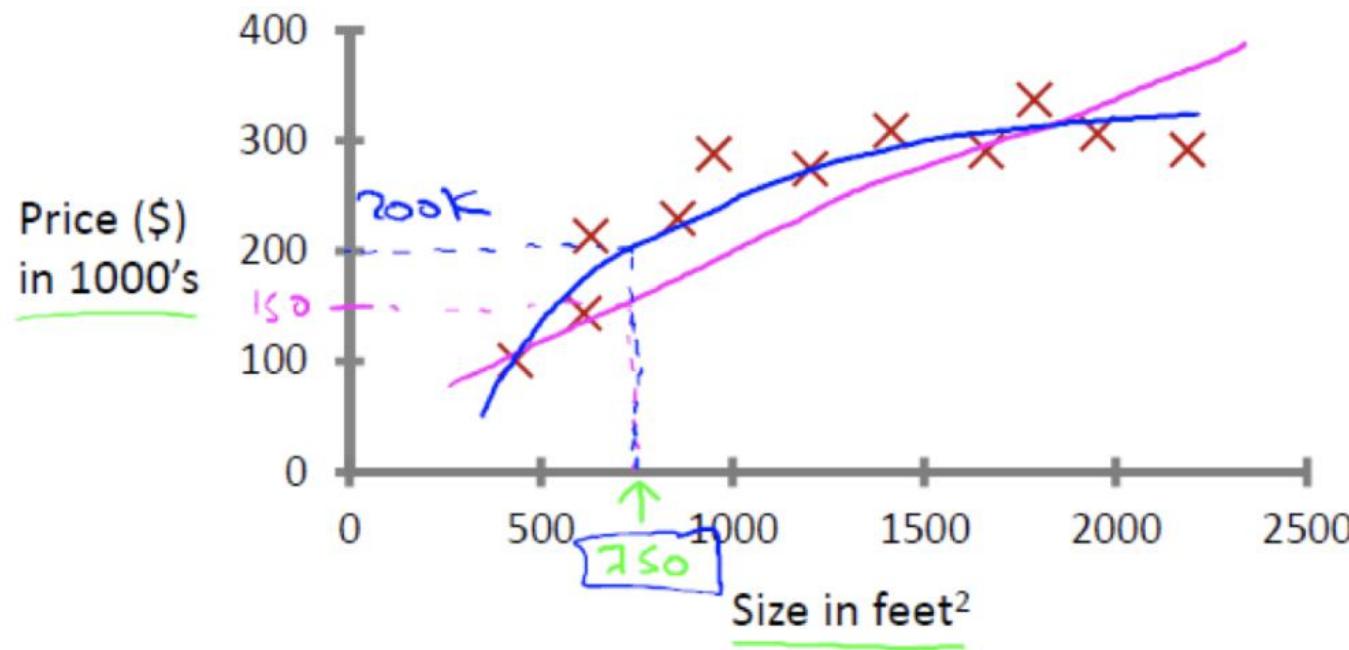
[ $f(X), Y$ ] ✓  
[ $f(X), Y$ ] ✓  
[ $f(X), Y$ ] ✓

# 4-3 半监督学习

## 有监督学习1：回归问题

回归问题：给定D维输入变量x，并且每一个输入矢量x都有对应的值y，要求对于新来的数据预测它对应的连续的目标值t。

例如：预测房价，根据样本集拟合出一条连续曲线



## 4-3 半监督学习

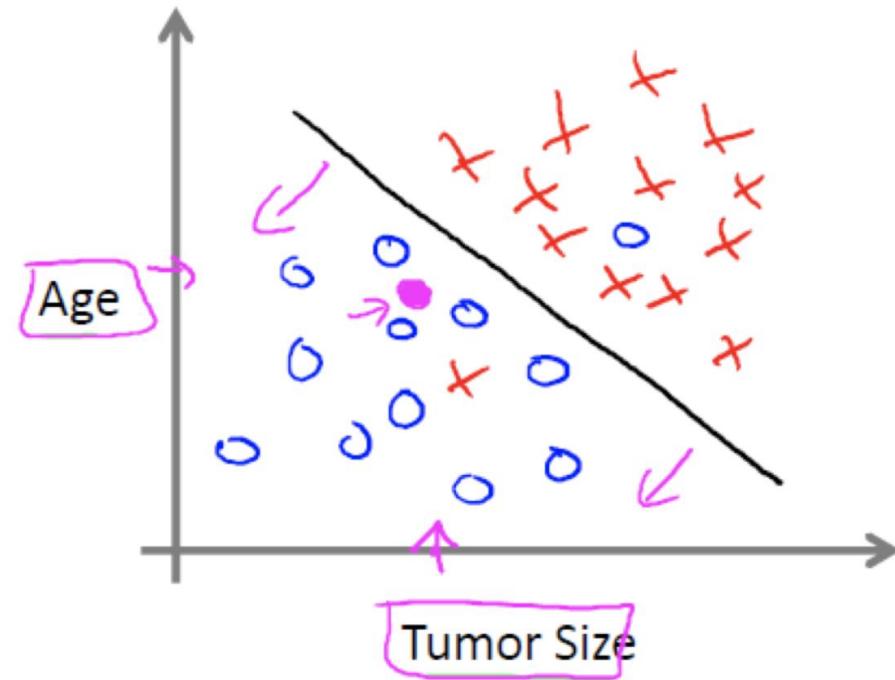
### 有监督学习2：分类问题

回归问题和分类问题的**本质一样**，都是针对输入做出输出预测，其**区别在于输出预测的类型**。

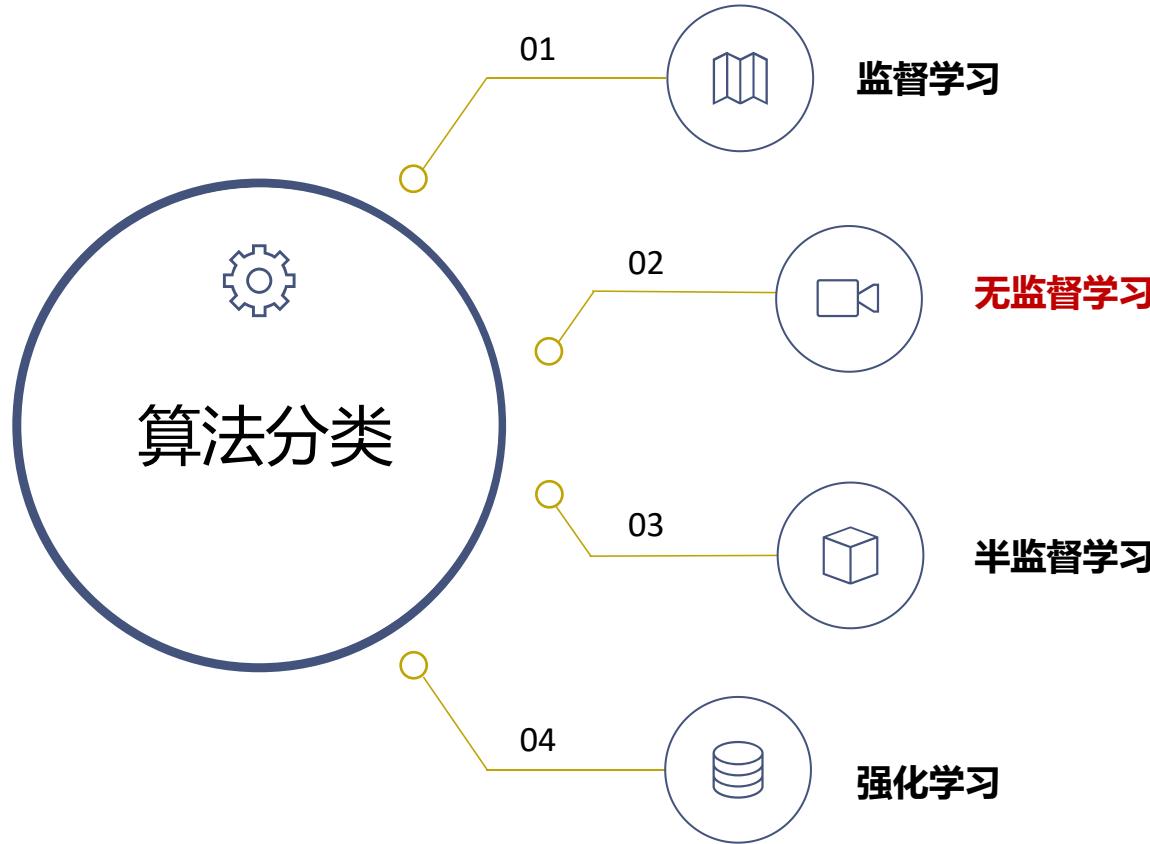
**分类问题**：给定一个新的模式，根据训练集推断它所对应的类别（如： $+1, -1$ ），是一种定性输出，也叫离散变量预测，而**回归问题**，给定一个新的模式，根据训练集推断它所对应的输出值（实数）是多少，是一种定量输出，也叫连续变量预测。

如果预测的结果为连续的值，是回归问题；如为离散的值，是分类问题

例如：根据肿瘤特征判断良性还是恶性，得到的是结果是“良性”或者“恶性”，是离散的；温度是连续的值，所以是回归。

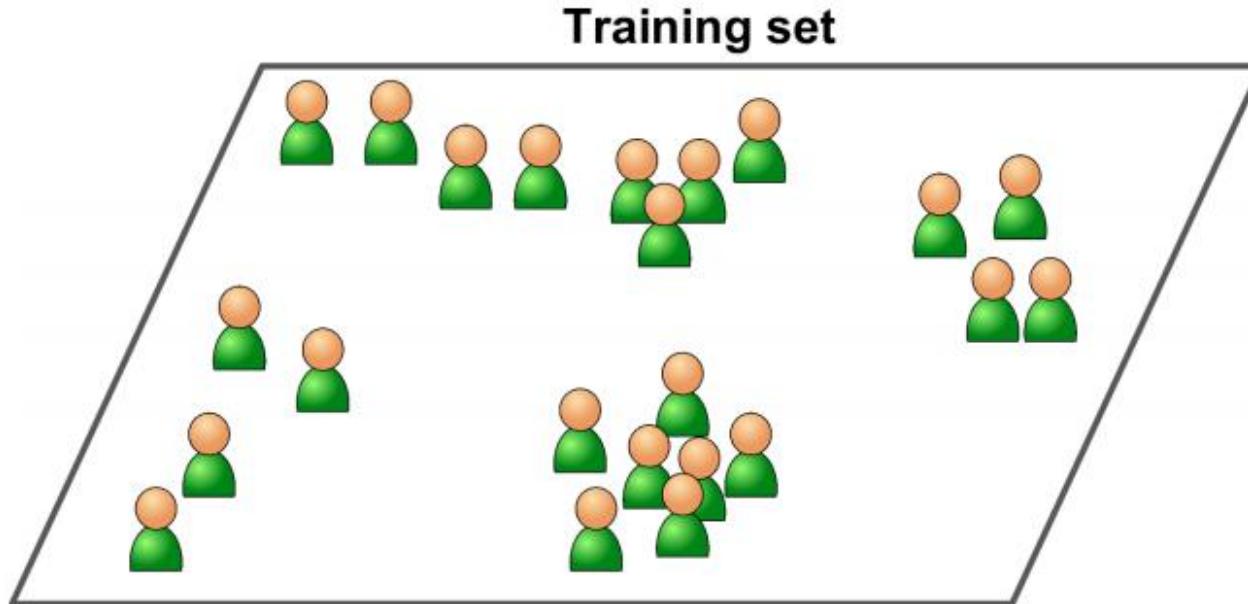


# 4 机器学习算法分类



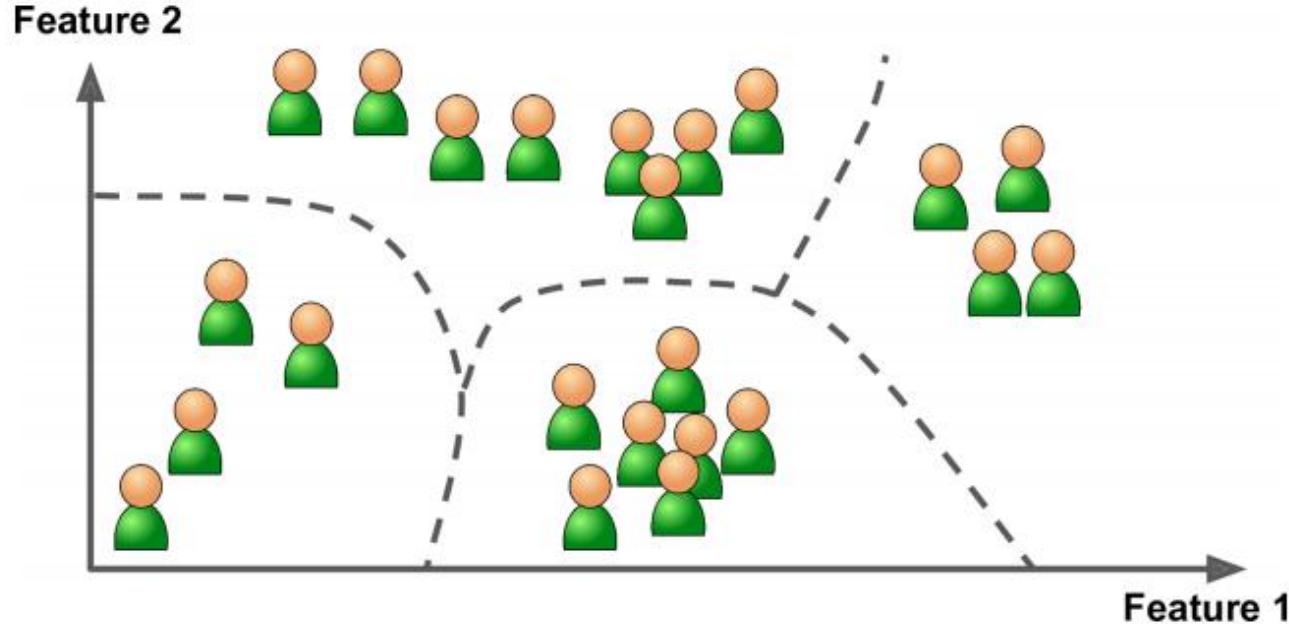
## 4-2 无监督学习

**无监督**定义：输入数据没有被标记，也没有确定的结果，即样本数据类别未知，**没有标签**，需要根据样本间的相似性对样本集进行聚类，以发现事物内部结构及相互关系。



## ■ 4-2 无监督学习

**无监督**定义：输入数据没有被标记，也没有确定的结果，即样本数据类别未知，**没有标签**，需要根据样本间的相似性对样本集进行聚类，以发现事物内部结构及相互关系。



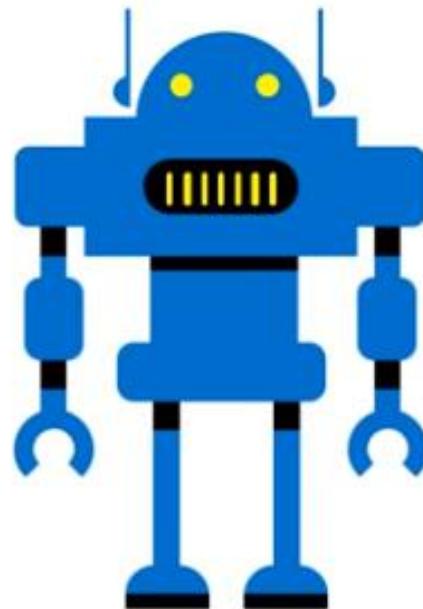
## 4-2 无监督学习

**无监督**定义：输入数据没有被标记，也没有确定的结果，即样本数据类别未知，**没有标签**，需要根据样本间的相似性对样本集进行聚类，以发现事物内部结构及相互关系。



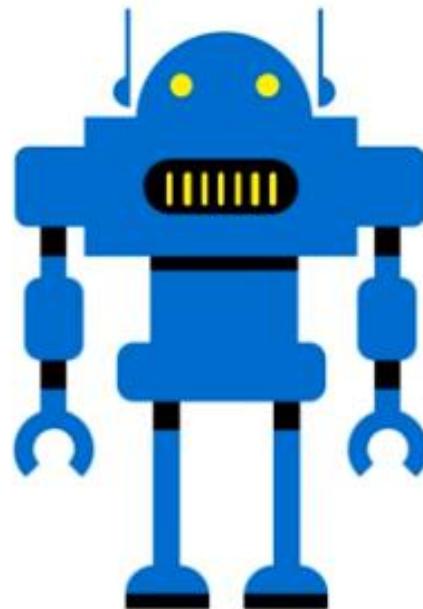
## ■ 4-2 无监督学习

[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]



## ■ 4-2 无监督学习

[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]  
[ $X_1, X_2, X_3$ ]



[Y]  
[Y]  
[Y]  
[Y]

## 4-2 无监督学习

q1

1类

**基于样本间相似性度量**的聚类方法：设法定出不同类别的核心或初始内核，然后依据样本与核心之间的相似性度量将样本聚集成不同的类别。

q2

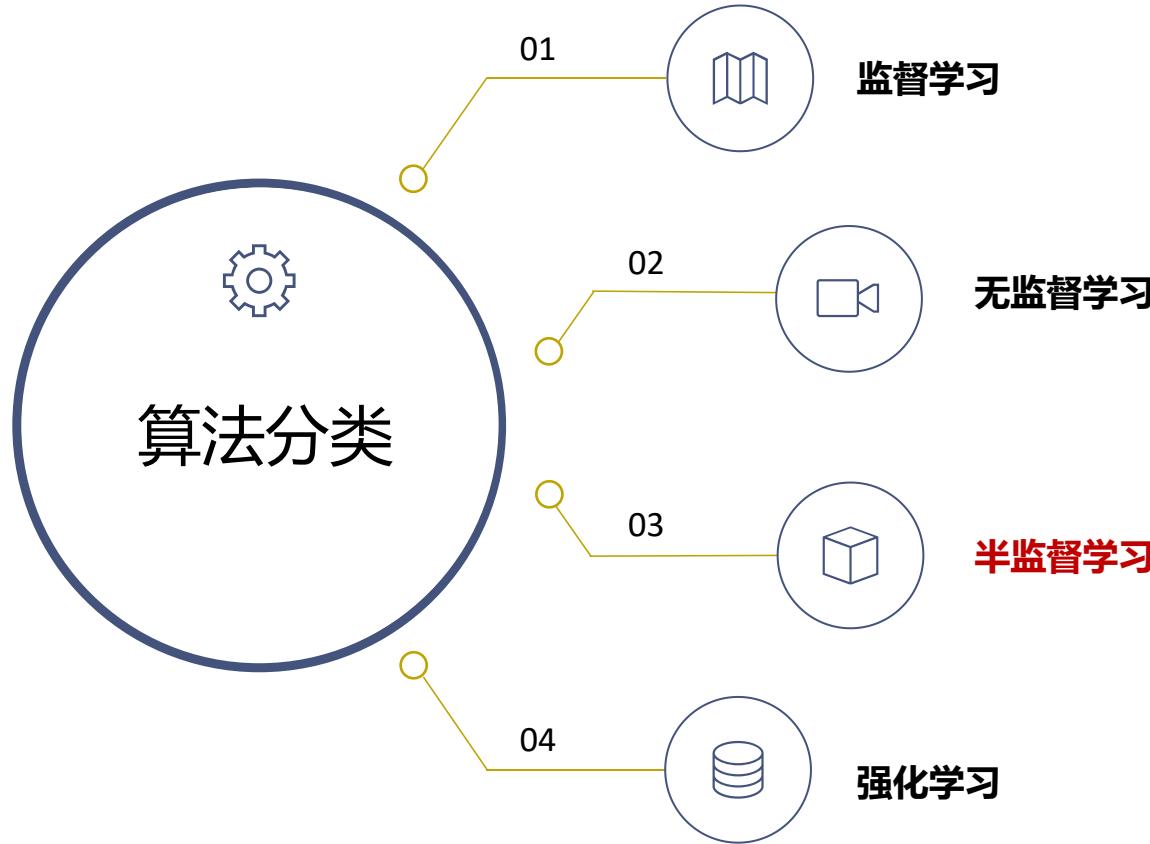
2类

**基于概率密度函数**估计的直接方法：指设法找到各类别在特征空间的分布参数，再进行分类。

**无监督学习**是在寻找数据集中的规律性，这种规律性并不一定要达到划分数据集的目的，也就是说不一定要“分类”。

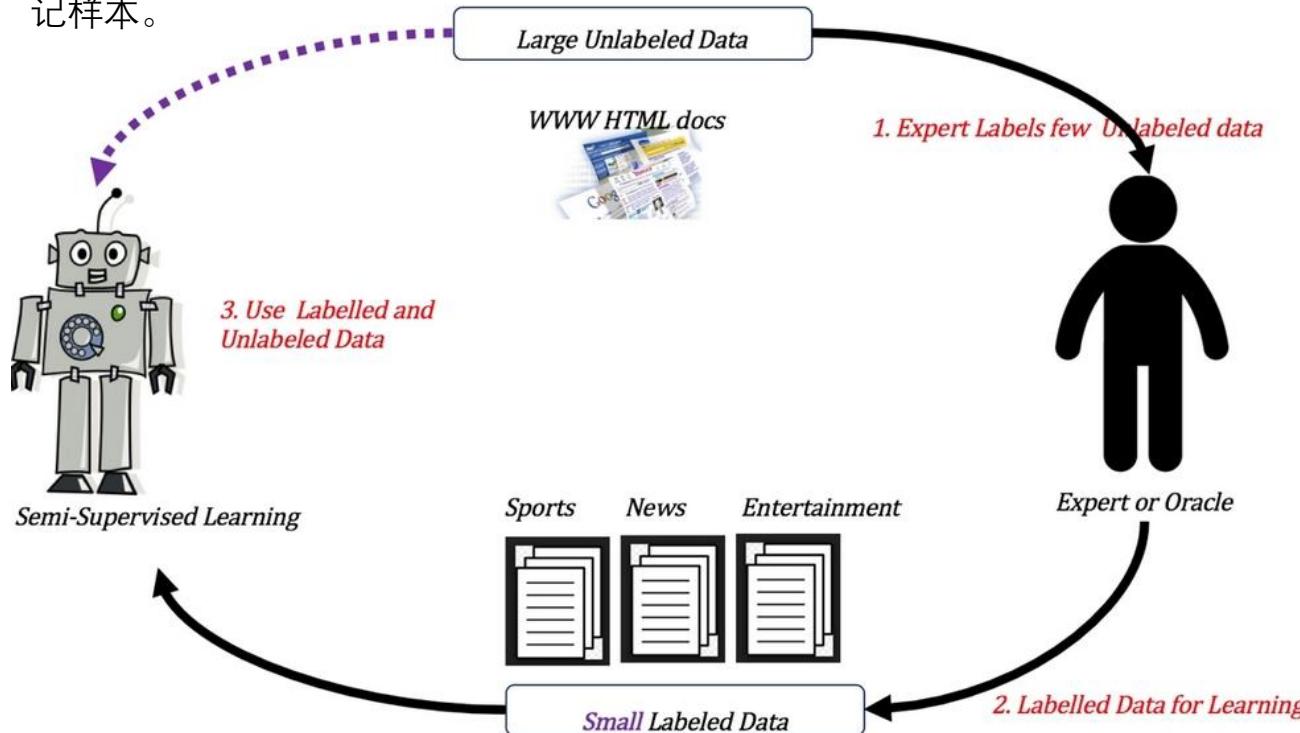
这一点是比有监督学习方法的用途要广。譬如分析一堆数据的主分量，或分析数据集有什么特点都可以归于非监督学习方法的范畴。

# 4 机器学习算法分类



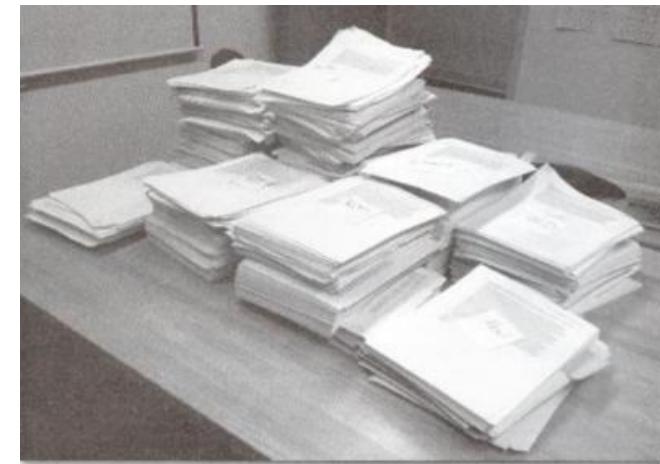
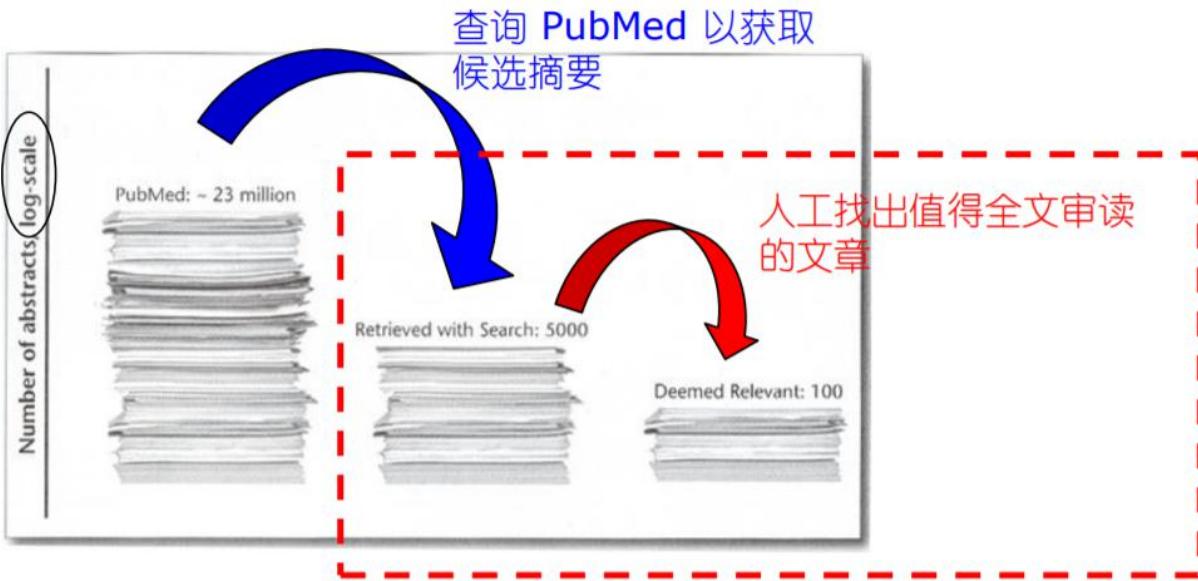
## 4-3 半监督学习

**半监督学习**: 训练集数据**一部分有**标签而其余部分无标签, 即训练集同时包含有标记样本和无标记样本。



## 4-3 半监督学习

### 半监督学习案例：循证医学

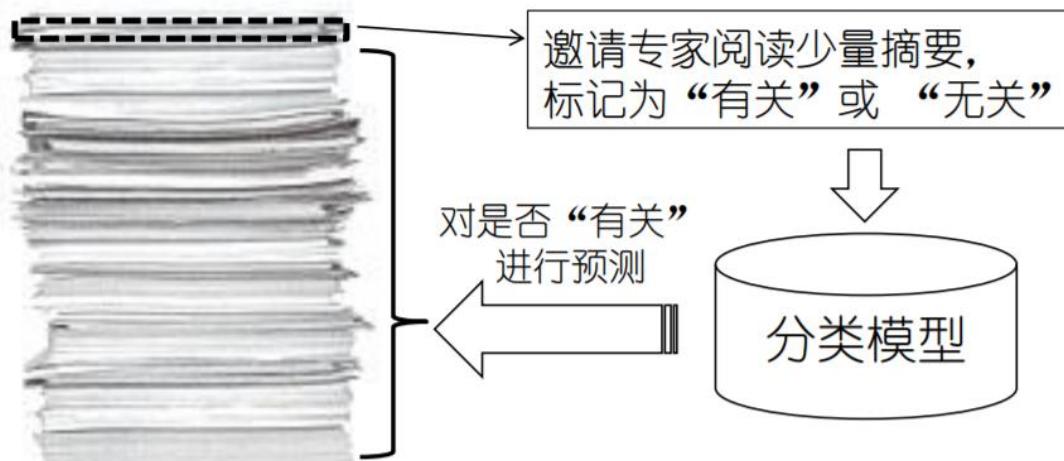


a portion of the 33,000 abstracts

## 4-3 半监督学习

### 半监督学习案例：循证医学

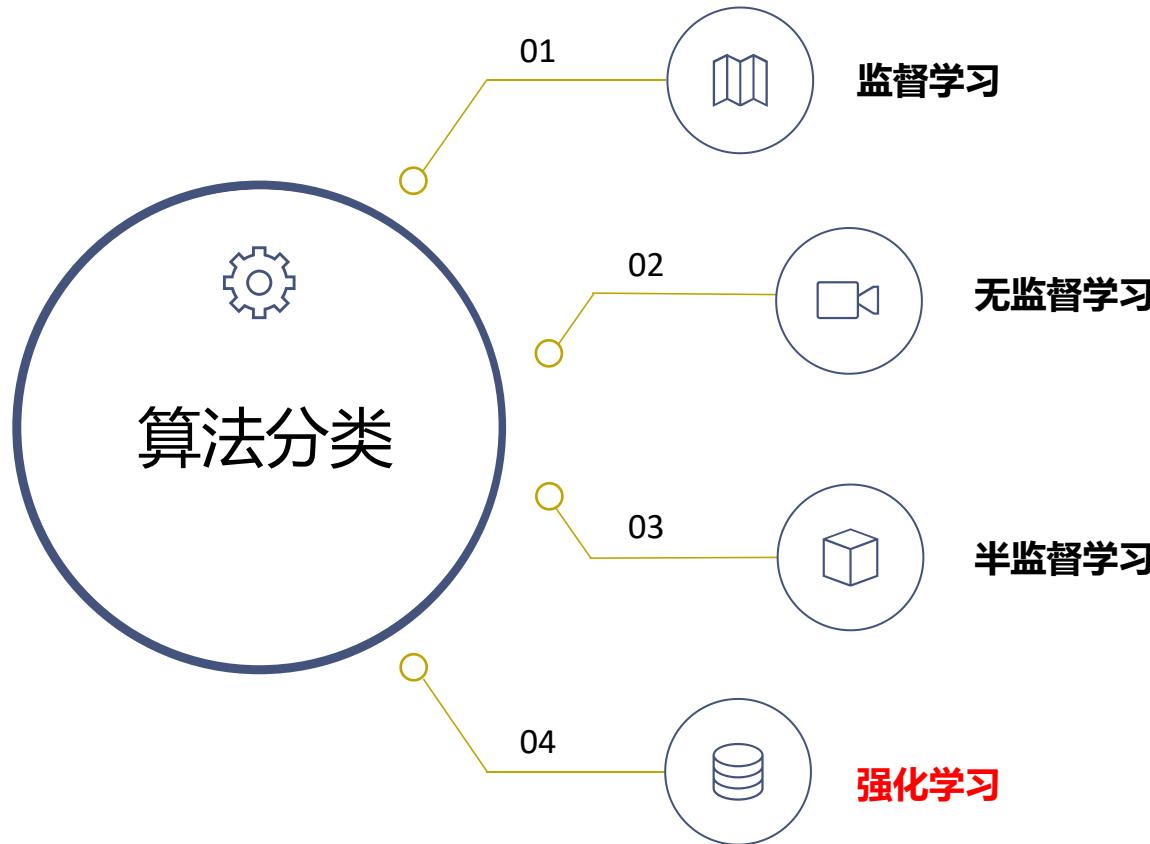
为了降低昂贵的成本，Tufts医学中心引入了机器学习技术



人类专家只需阅读 **50** 篇摘要，系统的自动筛选精度就达到 **93%**

人类专家阅读 **1,000** 篇摘要，则系统的自动筛选敏感度达到 **95%**  
(人类专家以前需阅读 **33,000** 篇摘要才能获得此效果)

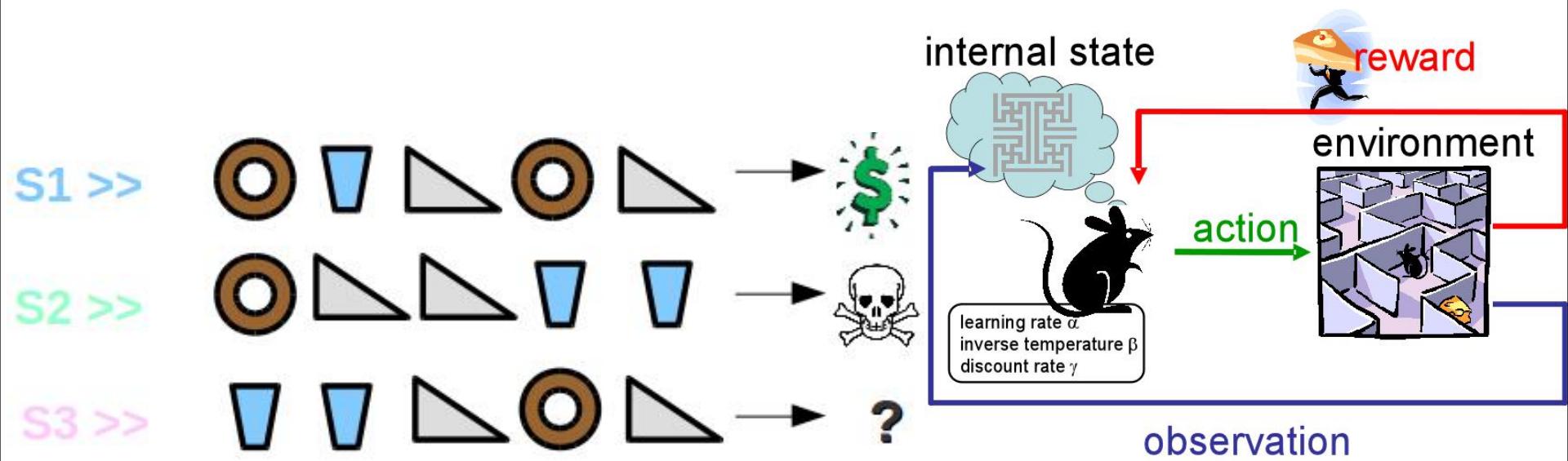
# 4 机器学习算法分类



## 4-4 强化学习

强化学习定义：实质是**make decisions** 问题，即自动进行决策，并且可以做连续决策**希望一段时间后获得最多的累计奖励**。

主要包含**四个要素**：agent，环境状态，行动，奖励；

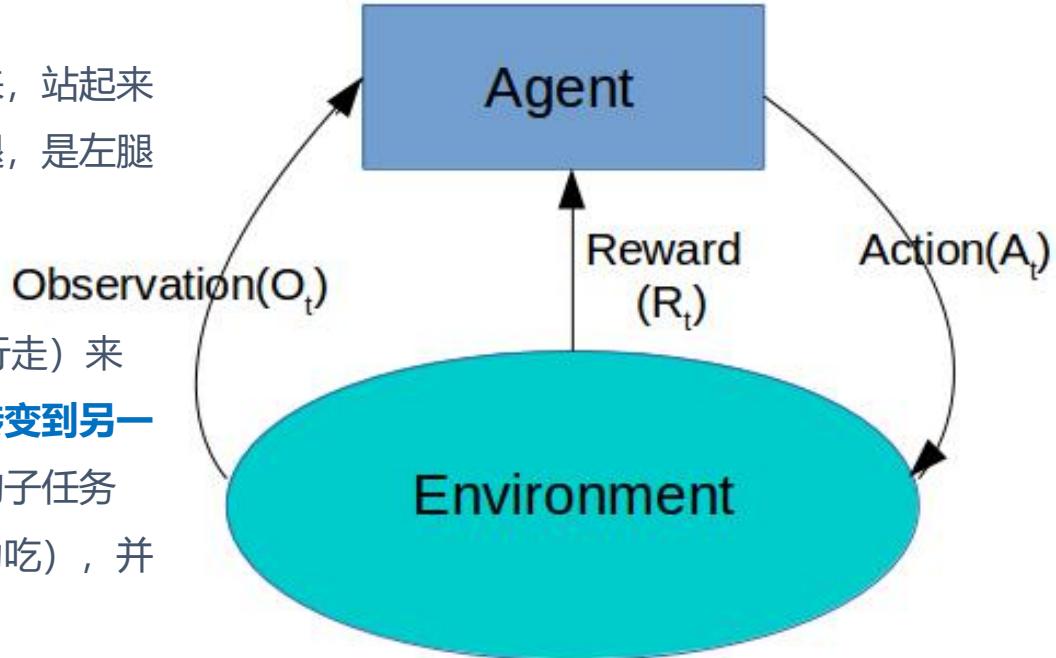


## 4-4 强化学习

### 强化学习案例1：

小孩想要走路，但在这之前，他需要先站起来，站起来之后还要保持平衡，接下来还要先迈出一条腿，是左腿还是右腿，迈出一步后还要迈出下一步。

小孩就是 **agent**，他试图通过采取**行动**（即行走）来操纵**环境**（行走的表面），并且从一个**状态转变到另一个状态**（即他走的每一步），当他完成任务的子任务（即走了几步）时，孩子得到**奖励**（给巧克力吃），并且当他不能走路时，就不会给巧克力。



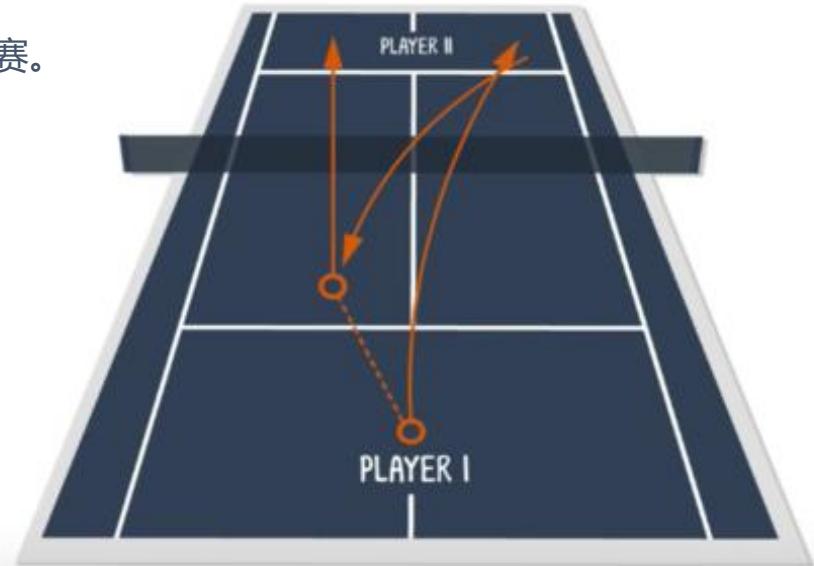
# ■ 4-4 强化学习

## 强化学习案例2：

一个 autonomous agent 要学习如何打 tennis 比赛，它需要考虑这些动作：serves, returns, and volleys，这些行为会影响谁赢谁输。

执行每一个动作都是在一个激励下进行的，就是要赢得比赛。  
为了实现比分最大化，它需要遵循一个策略。

I. PLAYER I	4	0	7		1
2. PLAYER II	0	6		1	

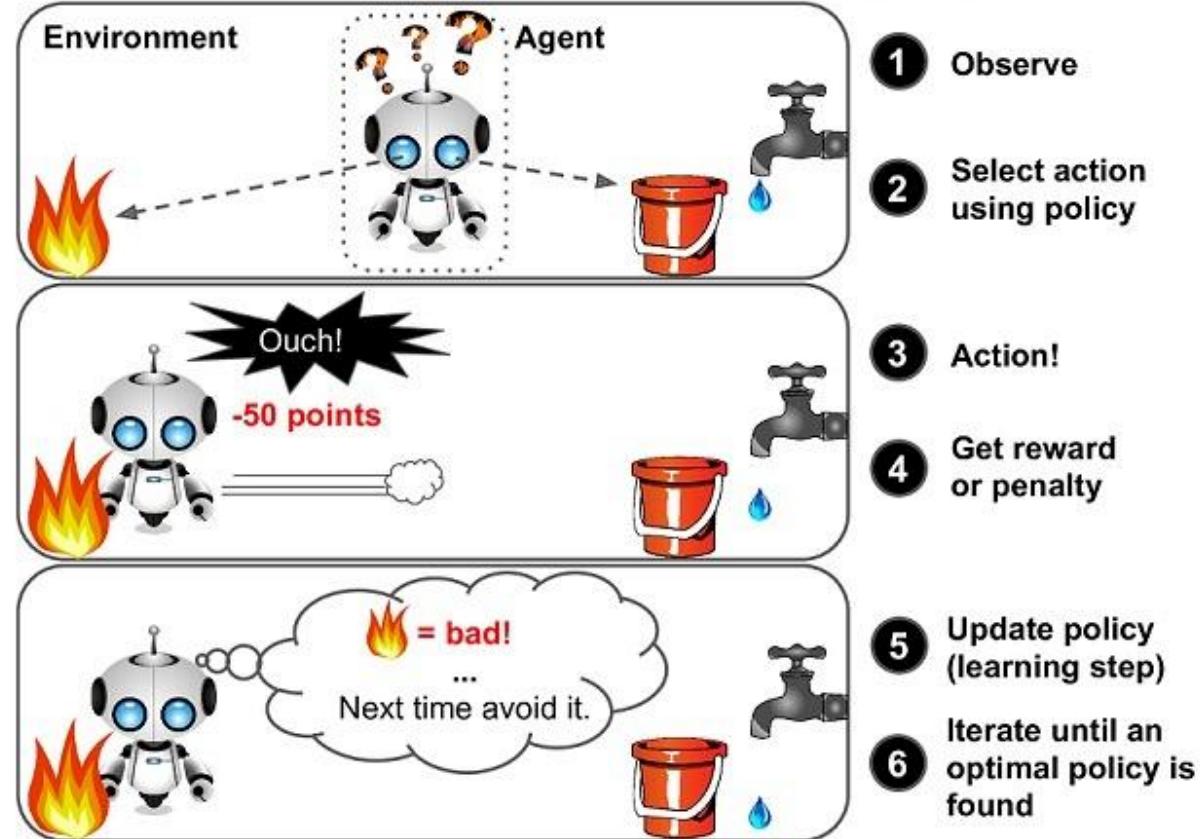


# 4-4 强化学习

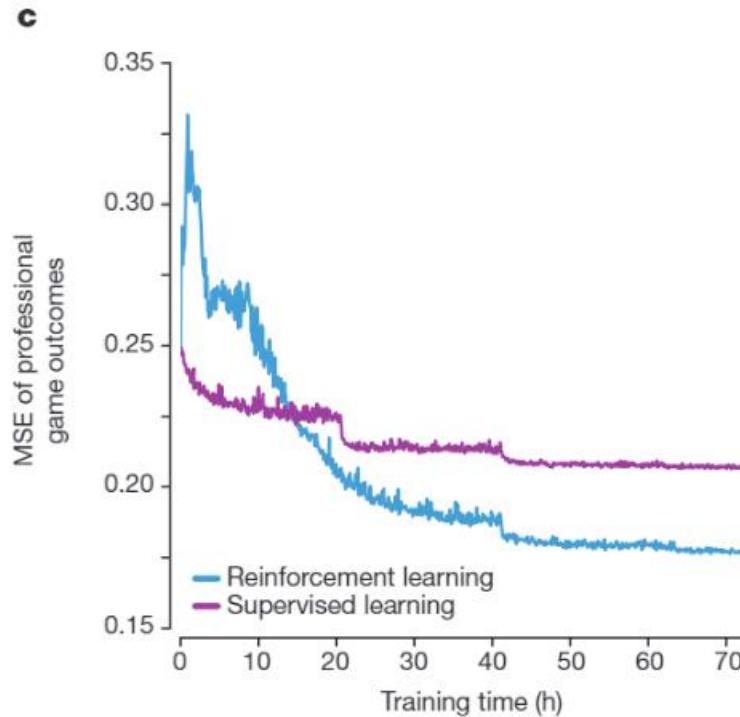
## 强化学习案例3：

### Manufacturing

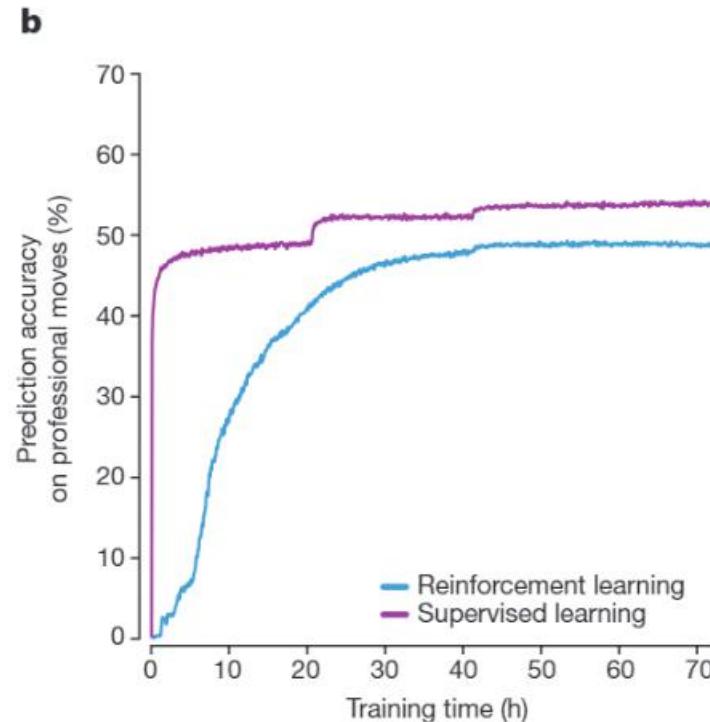
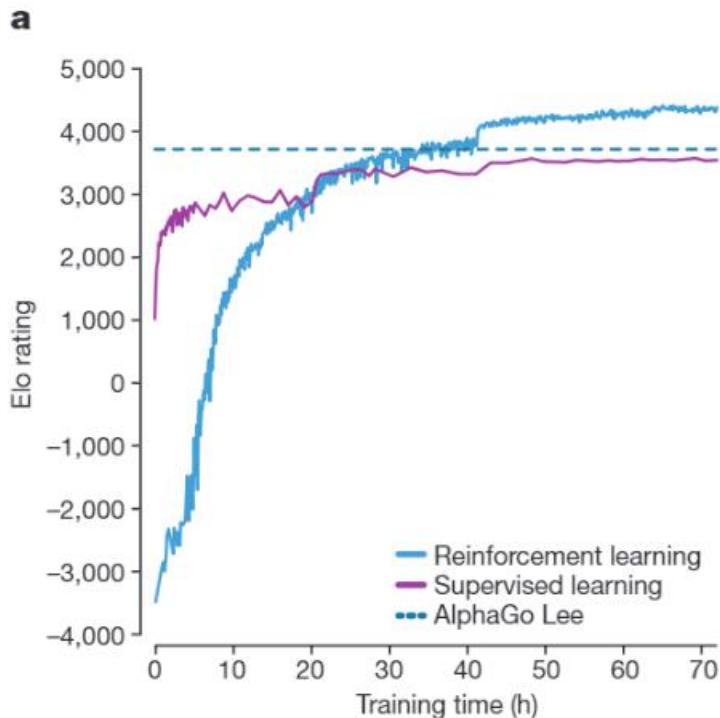
一家日本公司 Fanuc，工厂机器人在拿起一个物体时，会捕捉这个过程的视频，记住它每次操作的行动，操作成功还是失败了，积累经验，下一次可以更快更准确地采取行动。



请尝试总结监督学习和强化学习的区别：



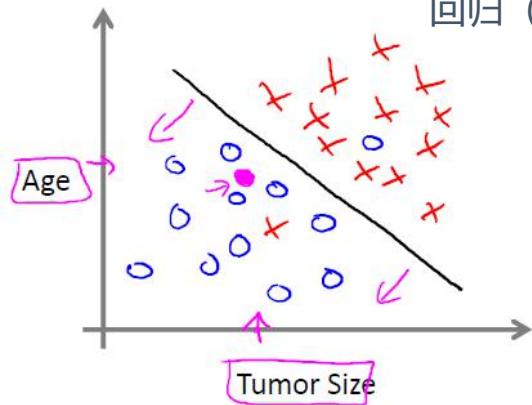
请尝试总结监督学习和强化学习的区别：



# 总结

	监督学习	强化学习
反馈映射	都会学习出输入到输出的一个映射，监督式学习出的是之间的关系，可以告诉算法什么样的输入对应着什么样的输出。	强化学习出的是给机器的反馈 <b>reward function</b> ，即用来判断这个行为是好是坏。
反馈时间	做了比较坏的选择会立刻反馈给算法。	结果反馈有延时，有时候可能需要走了很多步以后才知道以前的某一步的选择是好还是坏。
输入特征	输入是独立同分布的。	面对的输入总是在变化，每当算法做出一个行为，它影响下一次决策的输入。

# 总结



**监督学习**

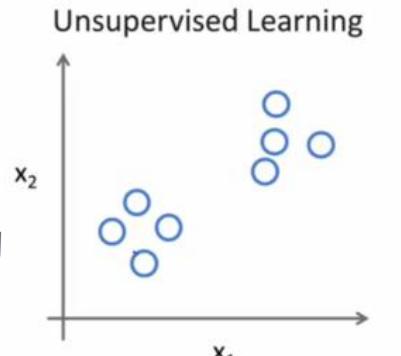
In: 有标签  
Out: 有反馈  
目的: 预测结果  
案例: 学认字  
算法: 分类 (类别)  
回归 (数字)



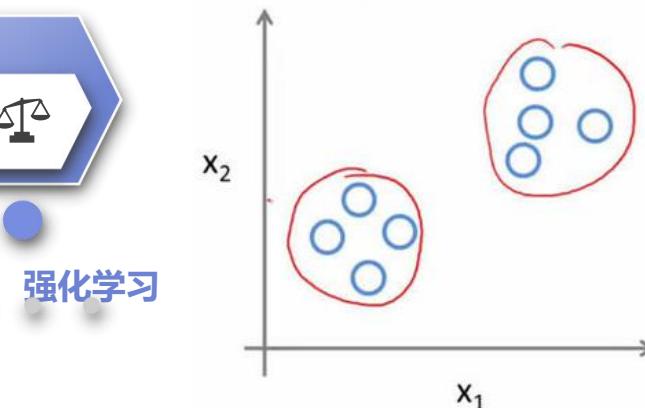
半监督学习

**无监督学习**

In: 无标签  
Out: 无反馈  
目的: 发现潜在结构  
案例: 自动聚类  
算法: 聚类, 降维



Unsupervised Learning

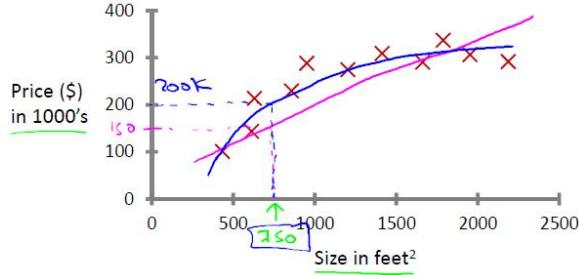


强化学习

# 总结

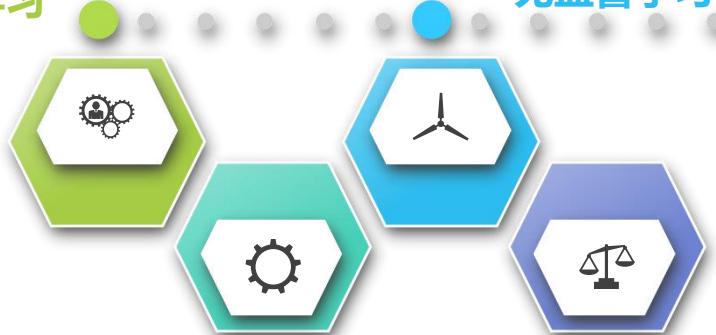
## 监督学习

Housing price prediction.



已知：训练样本Data和待分类的类别  
未知：训练样本有无标签均可  
应用：训练数据量过时，  
监督学习效果不能满足需求，  
因此用来增强效果。

## 无监督学习



## 半监督学习

## 强化学习

In: 决策流程及激励系统  
Out: 一系列行动  
目的: 长期利益最大化, 回报函数 (只会提示你是否在朝着目标方向前进的延迟反映)  
案例: 学下棋  
算法: 马尔科夫决策, 动态规划

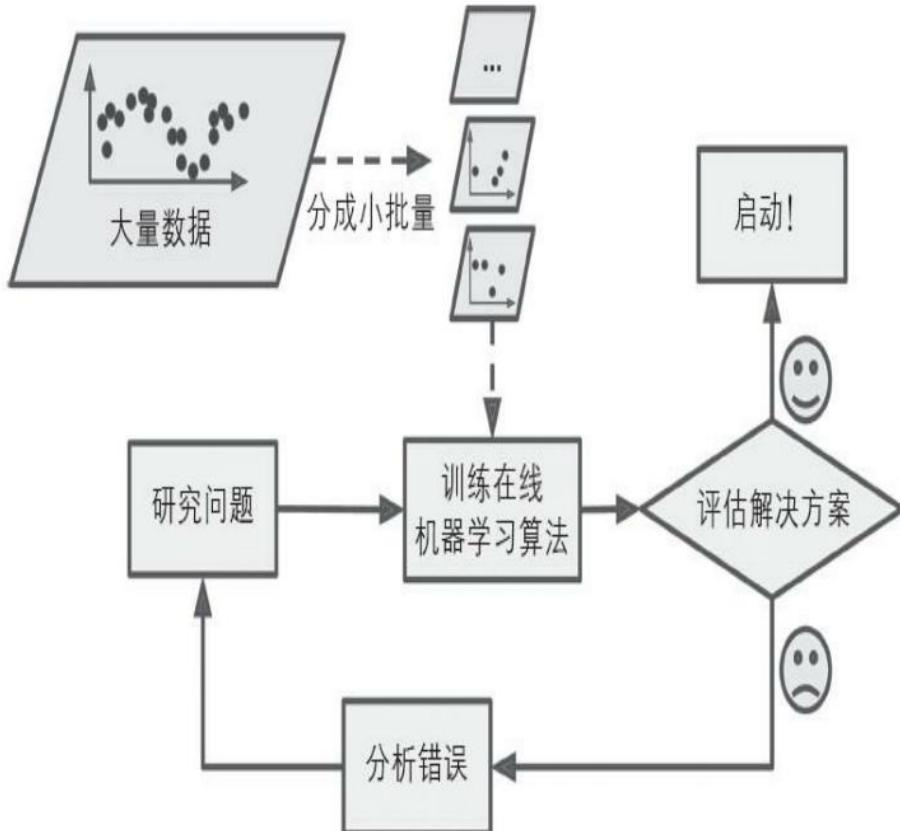
## 有监督学习与无监督学习对比

	有监督学习	无监督学习
样本	必须要有训练集与测试样本。在训练集中找规律，而对测试样本使用这种规律。	没有训练集，只有一组数据，在该组数据集内寻找规律。
目标	方法是识别事物，识别的结果表现在给待识别数据加上了标签。因此训练样本集必须由带标签的样本组成。	方法只有要分析的数据集的本身，预先没有什么标签。如果发现数据集呈现某种聚集性，则可按自然的聚集性分类，但不予以某种预先分类标签对上号为目的。

## 批量学习和在线学习

**在线学习：**需要接收持续的数据流（例如股票价格）同时对数据流的变化做出快速或自主的反应。如果你的计算资源有限，在线学习系统同样也是一个很好的选择：新的数据实例一旦经过系统的学习，就不再需要，你可以将其丢弃（除非你想要回滚到前一个状态，再“重新学习”数据），这可以节省大量的空间。

**挑战：**学习率，不良数据。



# 目录

## Contents

- ◆ 应用场景与发展历程
- ◆ 人工智能主要分支
- ◆ 机器学习工作流程
- ◆ 机器学习算法分类
- ◆ 模型评估

## 5-1 回归问题的评估

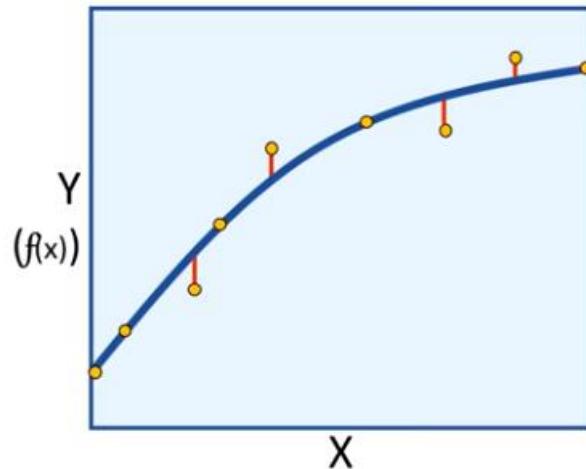


$$f([x_1, x_2, x_3, x_4, x_5, x_6, x_7]) = Y$$

## 5-1 回归问题的评估



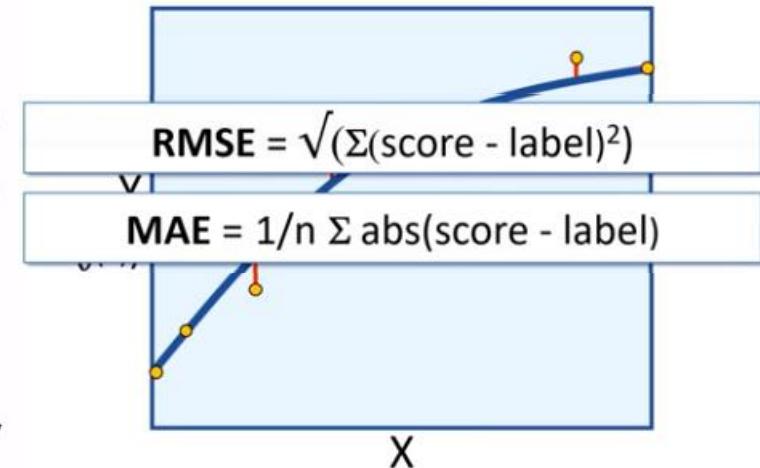
$$f([x_1, x_2, x_3, x_4, x_5, x_6, x_7]) = Y$$



## 5-1 回归问题的评估



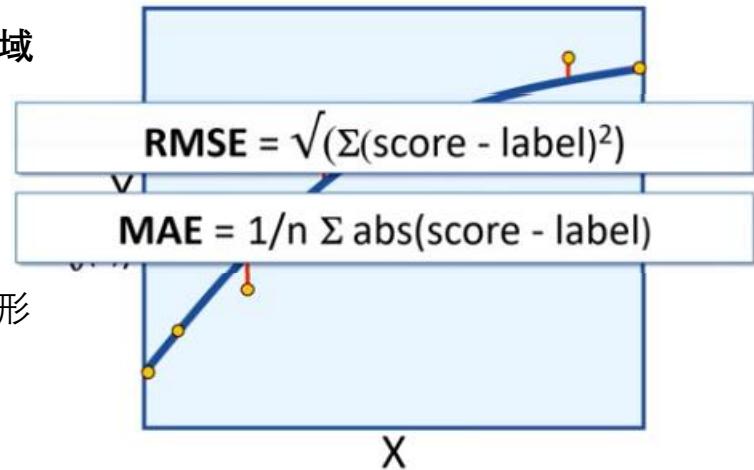
$$f([x_1, x_2, x_3, x_4, x_5, x_6, x_7]) = Y$$



## 5-1 回归问题的评估

RMSE通常是回归任务的首选性能衡量指标，但当有很多离群区域时，你可以考虑使用平均绝对误差（也称为平均绝对偏差MAE）

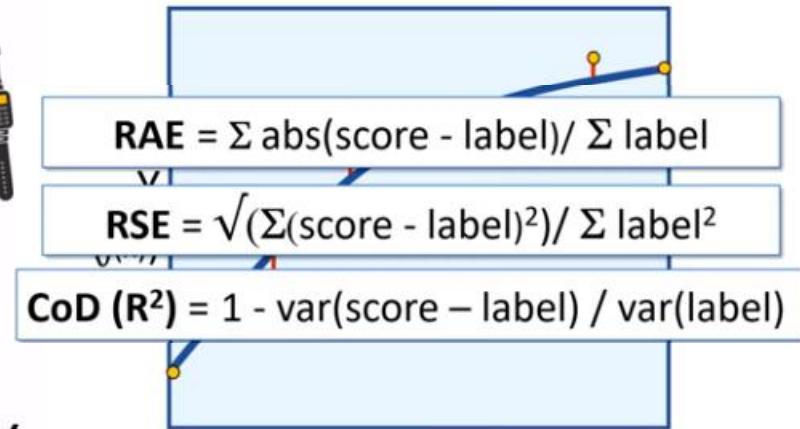
指数越高，则越关注大的价值，忽视小的价值。这就是为什么RMSE比MAE对异常值更敏感。但是当异常值非常稀少（例如钟形曲线）时，RMSE的表现优异，通常作为首选。



## 5-1 回归问题的评估



$$f([x_1, x_2, x_3, x_4, x_5, x_6, x_7]) = Y$$



## 5-1 分类问题的评估



## 5-2 分类问题的评估

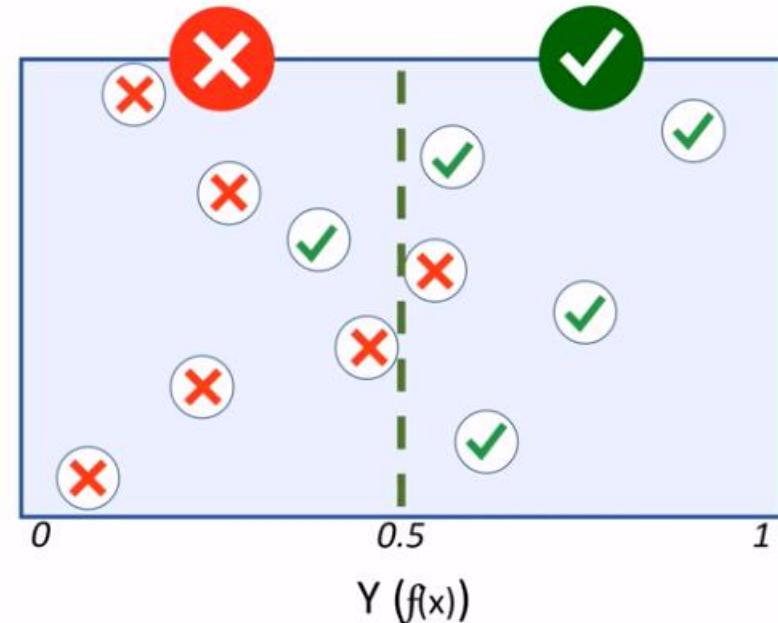


$$f([x_1, x_2, x_3, x_4]) = Y [1/0]$$

## 5-2 分类问题的评估



$$f([x_1, x_2, x_3, x_4]) = Y [1/0]$$



## 5-2 分类问题的评估

### 1. 混淆矩阵

预测结果

		预测结果	
		正例	假例
真实结果	正例	真正例TP	伪反例FN
	假例	伪正例FP	真反例TN

**True Positives,TP**: 预测为正样本，实际也为正样本的特征数

**False Positives,FP**: 预测为正样本，实际为负样本的特征数

**True Negatives,TN**: 预测为负样本，实际也为负样本的特征数

**False Negatives,FN**: 预测为负样本，实际为正样本的特征数

## 5-2 分类问题的评估

预测结果

	正例	假例
正例	真正例TP	伪反例FN
假例	伪正例FP	真反例TN

真实结果

如下所示：

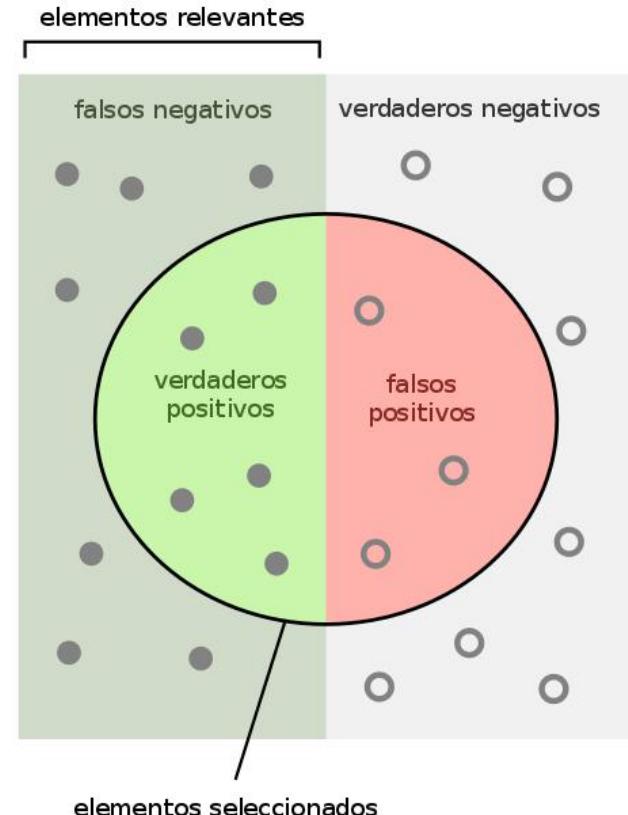
绿色的半圆TP(True Positives);

红色的半圆FP(False Positives);

左边的灰色长方形（不包括绿色半圆）是FN (False Negatives)

右边的浅灰色长方形（不包括红色半圆）是TN(True Negatives)。

绿色和红色组成的圆内代表我们分类得到模型结果认为是正值的样本。



# 5-2 分类问题的评估

## 2. 精确率与召回率

预测结果

	正例	假例
真实结果	正例	真正例TP 伪反例FN
假例	伪正例FP	真反例TN

How many selected items are relevant?

$$\text{Precision} = \frac{\text{ verdaderos positivos }}{\text{ verdaderos positivos } + \text{ falsos positivos }}$$

How many relevant items are selected?

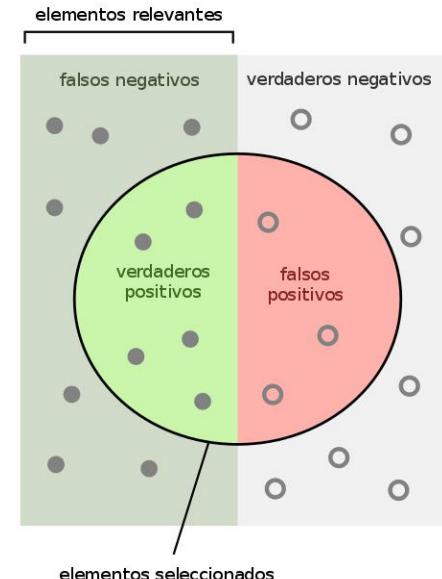
$$\text{Recall} = \frac{\text{ verdaderos positivos }}{\text{ verdaderos positivos } + \text{ falsos negativos }}$$

**精确率 (Precision) :** 绿色半圆除以红色绿色组成的圆

**召回率(Recall):** 是绿色半圆除以左边的长方形

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$



## 5-2 分类问题的评估

### 3, F分数

有时候我们对精确率和召回率并不是一视同仁，比如有时候我们更加重视精确率。我们用一个参数 $\beta$ 来度量两者之间的关系。如果 $\beta > 1$ , 召回率有更大影响, 如果 $\beta < 1$ , 精确率有更大影响。自然, 当 $\beta = 1$ 的时候, 精确率和召回率影响力相同, 和 $F_1$ 形式一样。含有度量参数 $\beta$ 的 $F_1$ 我们记为 $F_\beta$ , 严格的数学定义如下:

$$F_\beta = \frac{(1 + \beta^2) * P * R}{\beta^2 * P + R}$$

$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

# 5-2 分类问题的评估

## 4, ROC 曲线

**ROC曲线**是根据一系列不同的二分类方式（分界值或决定阈），以TPR真阳性率（灵敏度）为纵坐标，FPR假阳性率（1-特异度）为横坐标绘制的曲线

true positive rate ,TPR, 它是所有实际正例中，正确识别的正例比例：

$$TPR = \frac{TP}{TP + FN}$$

false positive rate, FPR, 它是实际负例中，错误得识别为正例的负例比例：

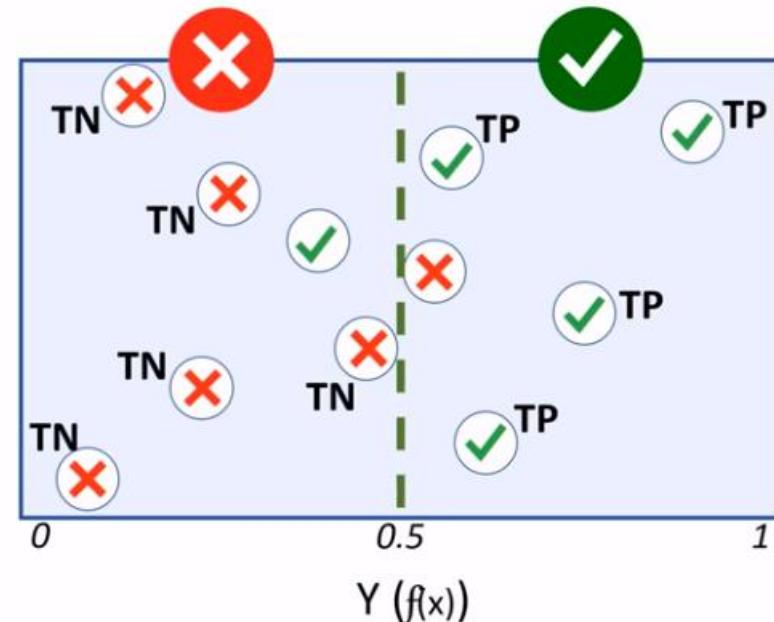
$$FPR = \frac{FP}{FP + TN}$$

## 5-2 分类问题的评估

### 4, ROC 曲线\_案例分析



$$f([x_1, x_2, x_3, x_4]) = Y [1/0]$$

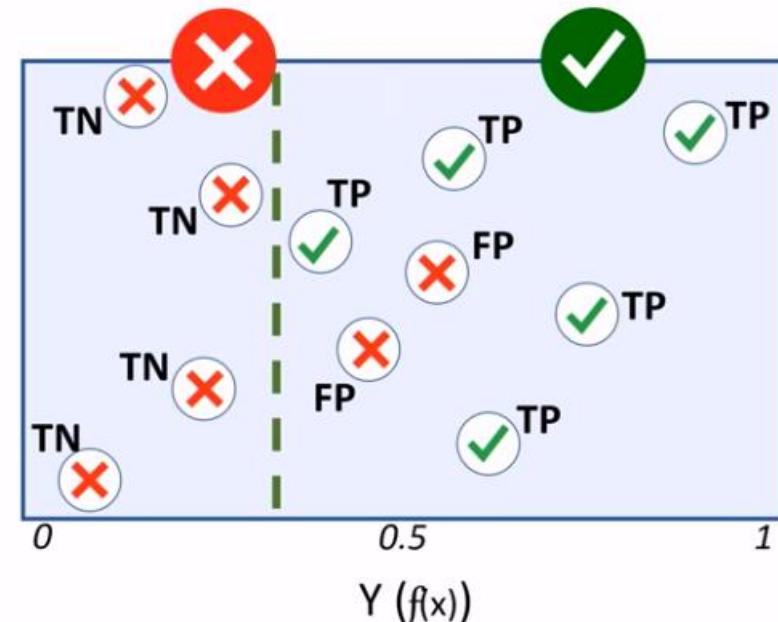


## 5-2 分类问题的评估

### 4, ROC 曲线\_案例分析



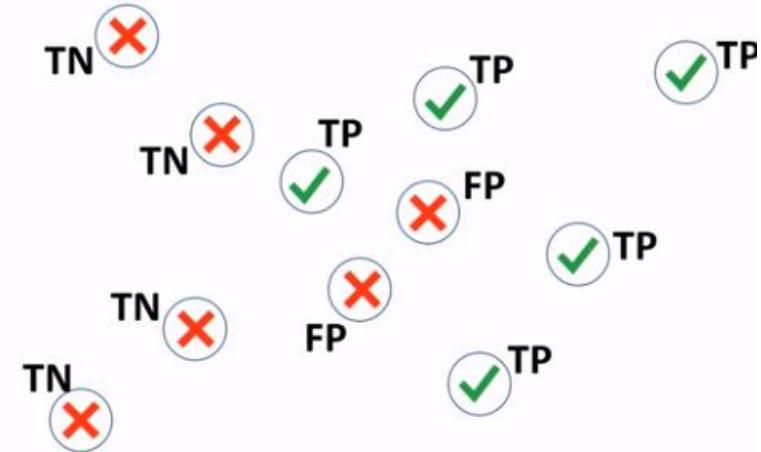
$$f([x_1, x_2, x_3, x_4]) = Y [1/0]$$



## 5-2 分类问题的评估

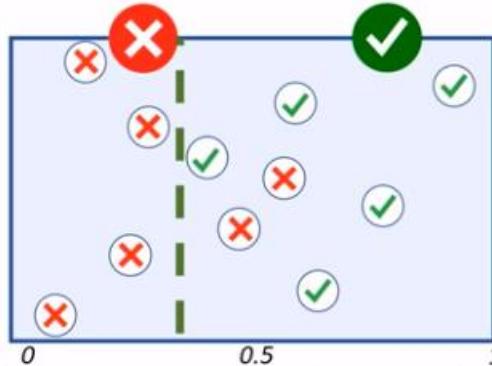
### 4, ROC 曲线\_案例分析

Predicted	Actual	
	1	0
1	5	2
0	0	4



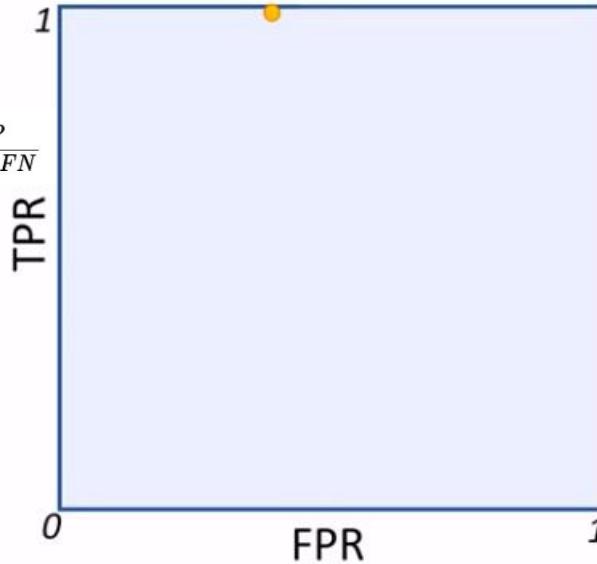
## 5-2 分类问题的评估

### 4, ROC 曲线\_案例分析



Predicted	Actual 1	0
1	5	2
0	0	4

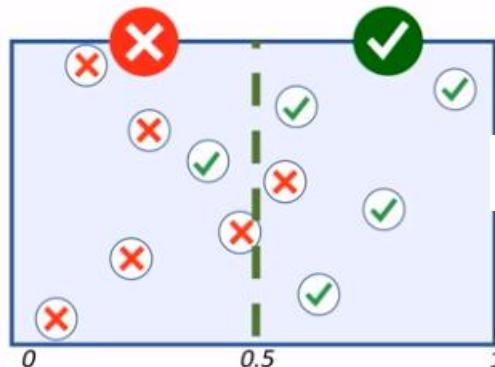
$$TPR = \frac{TP}{TP + FN}$$



$$FPR = \frac{FP}{FP + TN}$$

## 5-2 分类问题的评估

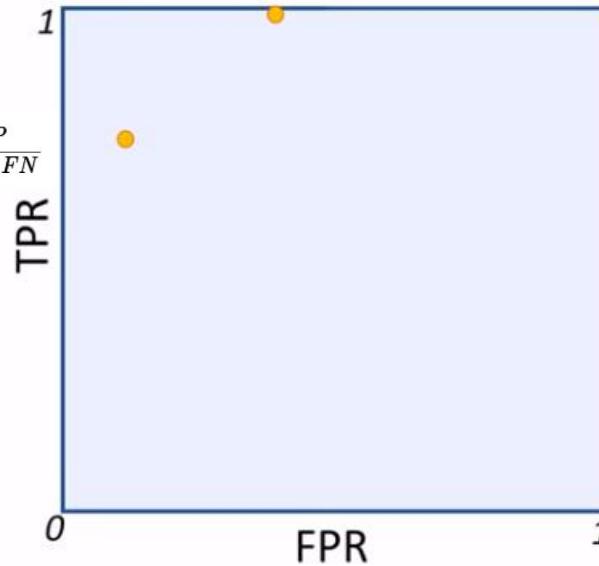
### 4, ROC 曲线\_案例分析



$$TPR = \frac{TP}{TP + FN}$$

Predicted	Actual 1	0
1	4	1
0	1	5

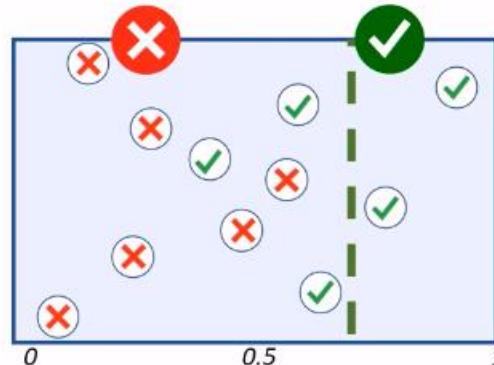
T  
F



$$FPR = \frac{FP}{FP + TN}$$

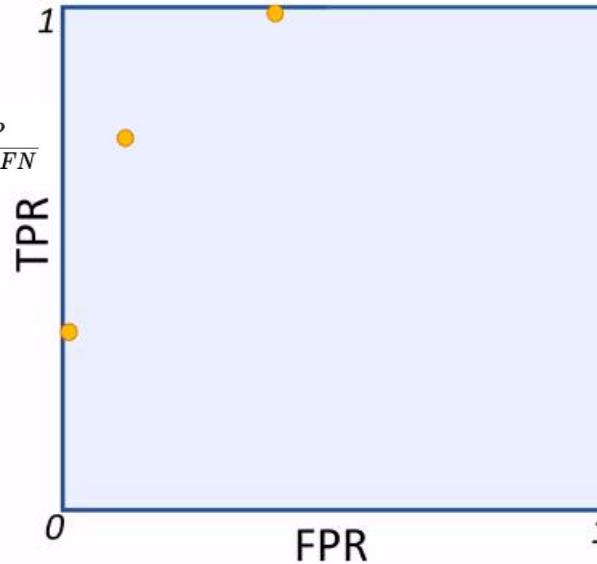
## 5-2 分类问题的评估

### 4, ROC 曲线\_案例分析



$$TPR = \frac{TP}{TP + FN}$$

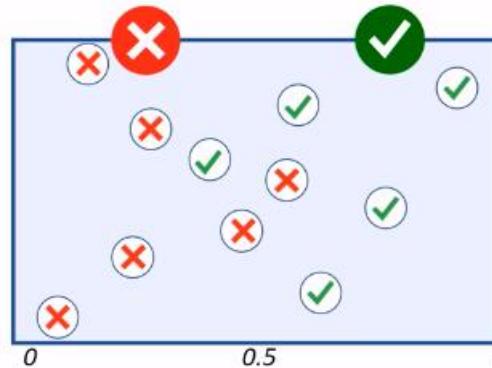
Predicted	Actual 1	Actual 0
Actual 1	2	0
Actual 0	3	6



$$FPR = \frac{FP}{FP + TN}$$

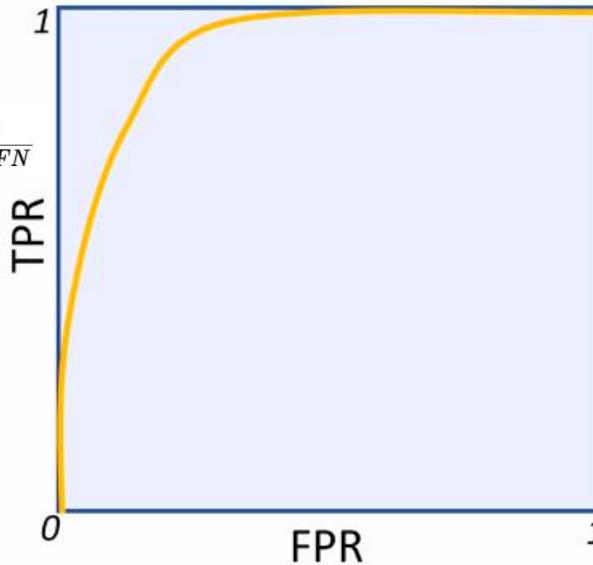
## 5-2 分类问题的评估

### 4. ROC 曲线\_案例分析



$$TPR = \frac{TP}{TP + FN}$$

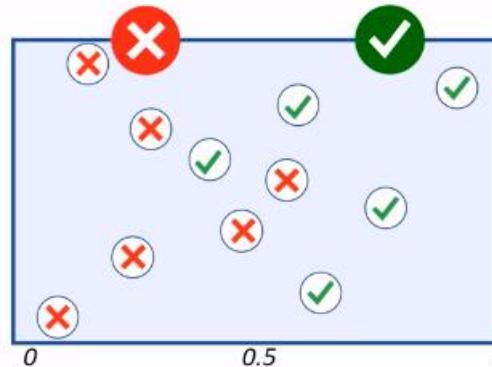
Predicted	Actual 1	0
1	N	N
0	N	N



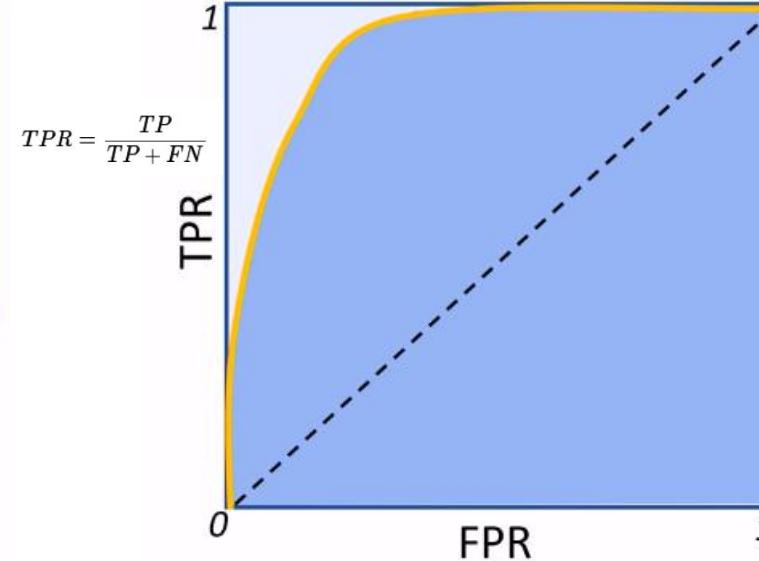
$$FPR = \frac{FP}{FP + TN}$$

## 5-2 分类问题的评估

### 4, ROC 曲线\_案例分析



Predicted	Actual 1	0
1	N	N
0	N	N



$$FPR = \frac{FP}{FP + TN}$$

## 5-2 分类问题的评估

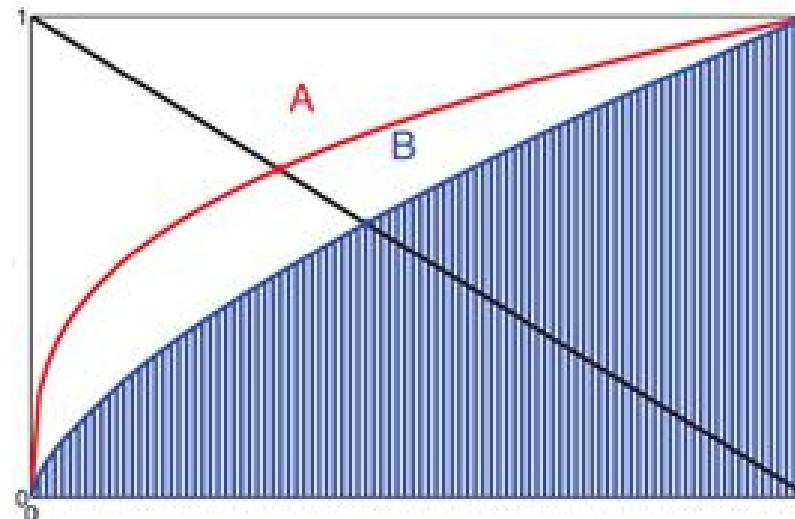
### 4, ROC 曲线\_案例分析

最完美的分类器（完全区分正负样例）：过  $(0,1)$  点，即没有 FP，全是 TP

曲线越是“凸”向左上角，说明分类器效果越好，即曲线上离  $(0,1)$  越近的点分类效果越好

随机预测会得到  $(0,0)$  和  $(1,1)$  的直线上的一个点

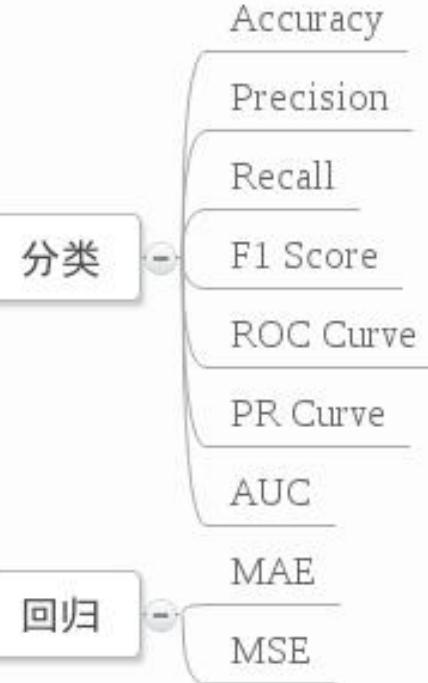
AUC(area under thecurve)，也就是ROC曲线的下夹面积，越大说明分类器越好，最大值是1，图中的蓝色条纹区域面积就是蓝色曲线对应的 AUC



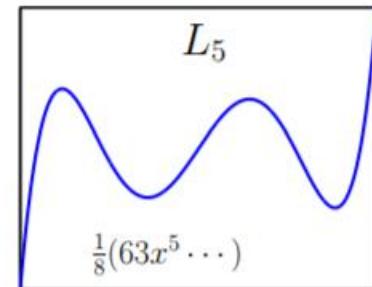
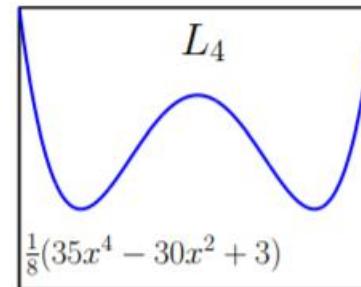
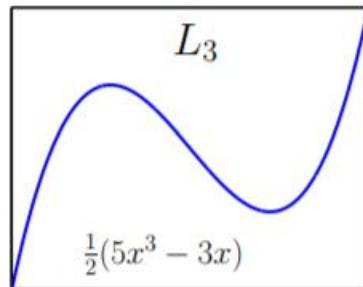
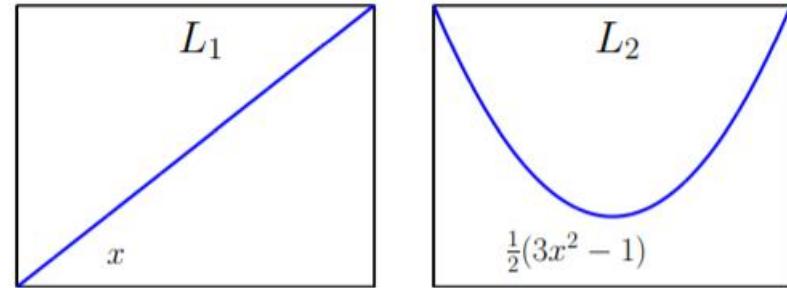
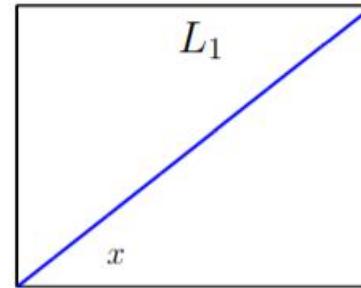
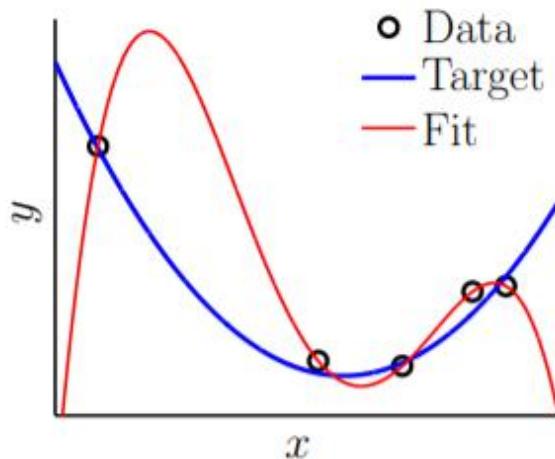
# 总结



机器学习性能评估指标



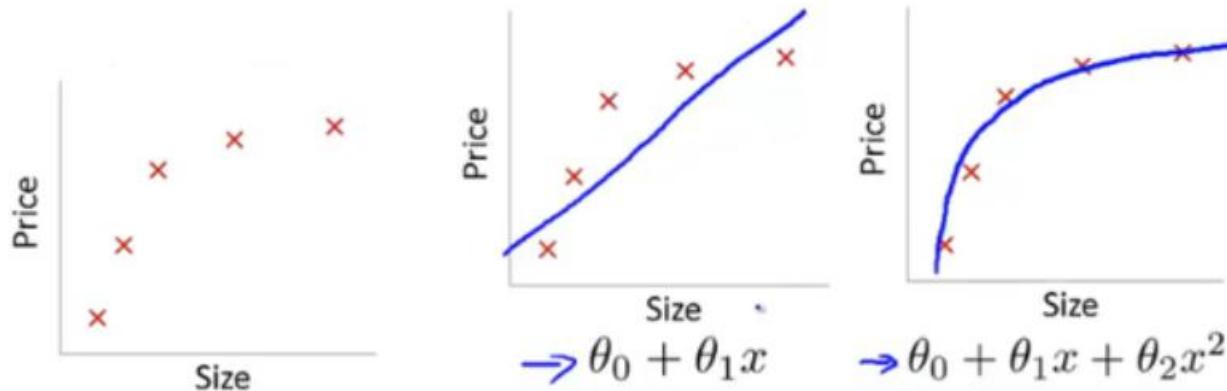
## 5-3 拟合与优化



## 5-3 拟合与优化

导入：如何评价模型，在训练集和测试集中的表现怎么样？

欠拟合（under-fitting）：所建的学习模型在训练集和测试集中均表现不佳。



size与prize关系的数据，中间的图就是出现欠拟合的模型，不能够很好地拟合数据，如果在中间的图的模型后面再加一个二次项，就可以很好地拟合图中的数据了，如右面的图所示。

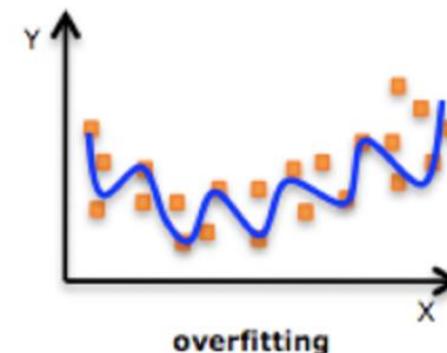
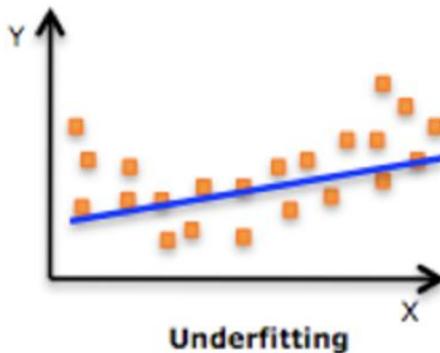
## 欠拟合解决方法：

- 1) **添加其他特征项**，有时候我们模型出现欠拟合的时候是因为特征项不够导致的，可以添加其他特征项来很好地解决。例如，“组合”、“泛化”、“相关性”三类特征是特征添加的重要手段，无论在什么场景，都可以照葫芦画瓢，总会得到意想不到的效果。除上面的特征之外，“上下文特征”、“平台特征”等等，都可以作为特征添加的首选项。
- 2) **添加多项式特征**，这个在机器学习算法里面用的很普遍，例如将线性模型通过添加二次项或者三次项使模型泛化能力更强。例如上面的图片的例子。

# 5-3 拟合与优化

过拟合 (over-fitting)：所建的学习模型在训练样本中表现优越，在测试数据集中表现不佳。

过拟合解释：当某个模型过度的学习训练数据中的细节和噪音，以至于模型在新的数据上表现很差，我们称过拟合发生了。这意味着训练数据中的噪音或者随机波动也被当做概念被模型学习了。而问题就在于这些概念不适用于新的数据，从而导致模型泛化性能的变差。



$$\theta_0 + \theta_1 x$$

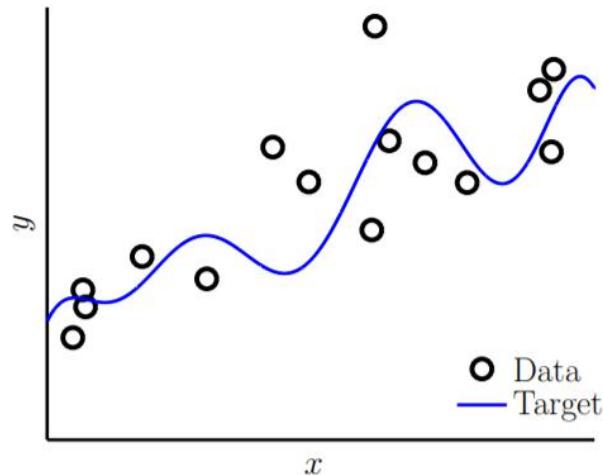
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

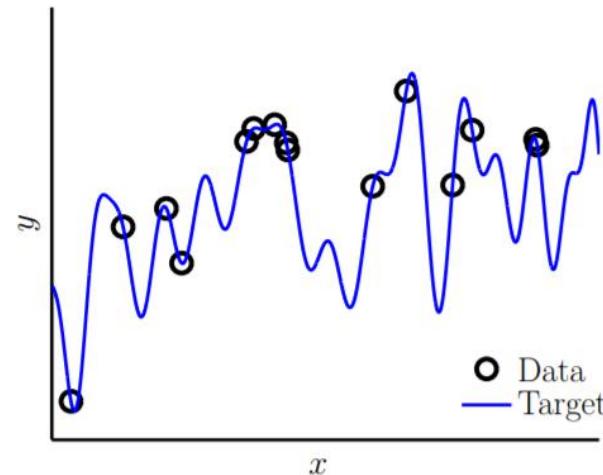
# 5-3 拟合与优化

虽然在训练的时候模型可以很好地匹配数据，但是很显然过拟合（过度扭曲）了曲线，不是真实的size与prize曲线。

## Case Study: 2nd vs 10th Order Polynomial Fit



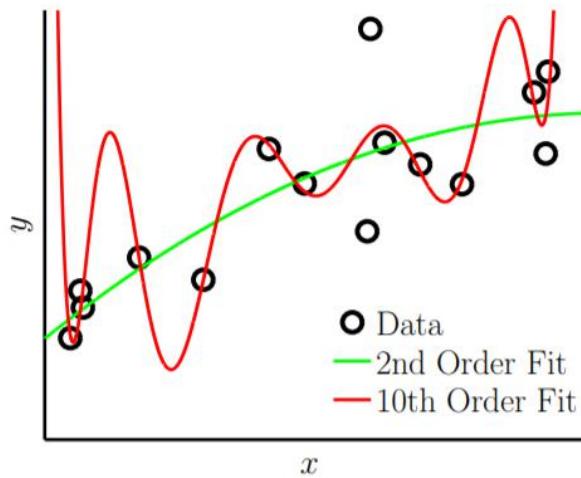
10th order  $f$  with noise.



50th order  $f$  with no noise.

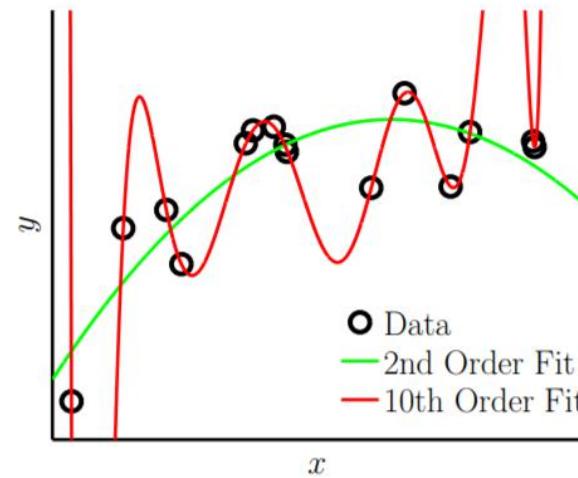
# 5-3 拟合与优化

## Case Study: 2nd vs 10th Order Polynomial Fit



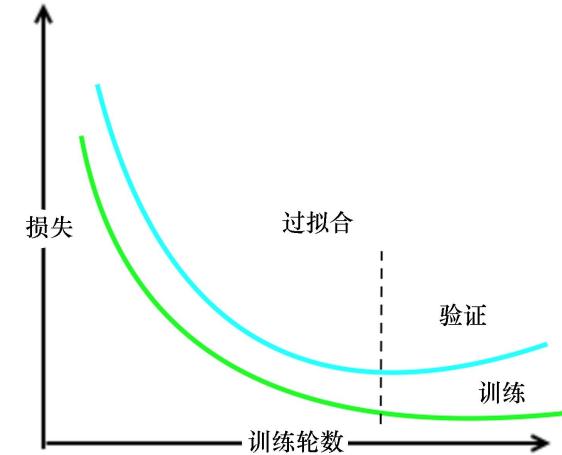
simple noisy target

	2nd Order	10th Order
$E_{in}$	0.050	0.034
$E_{out}$	0.127	9.00

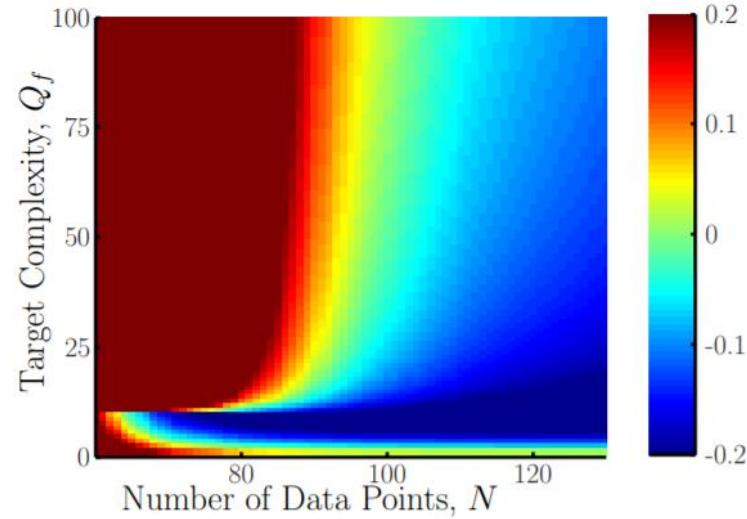
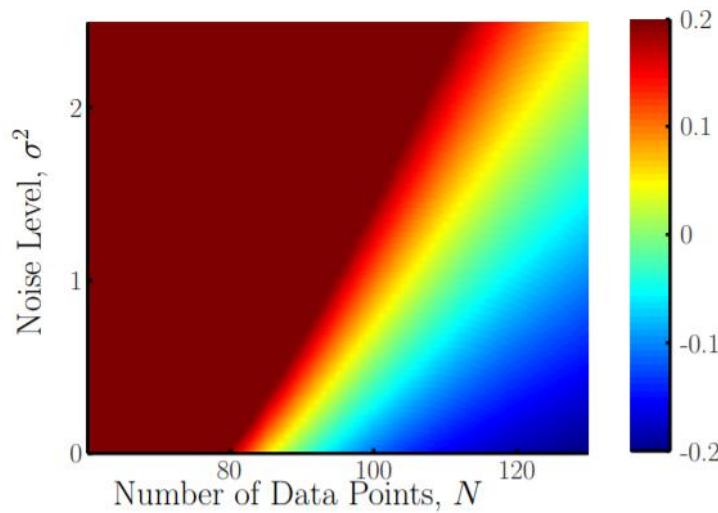


complex noiseless target

	2nd Order	10th Order
$E_{in}$	0.029	$10^{-5}$
$E_{out}$	0.120	7680

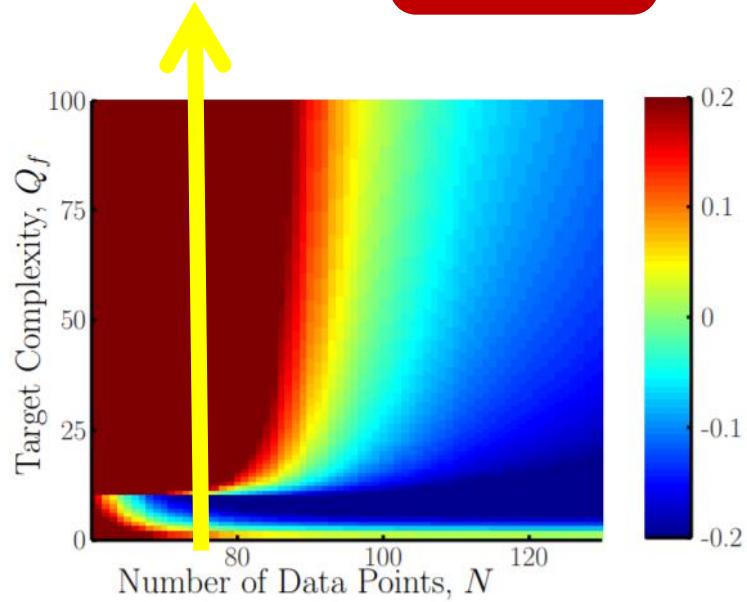
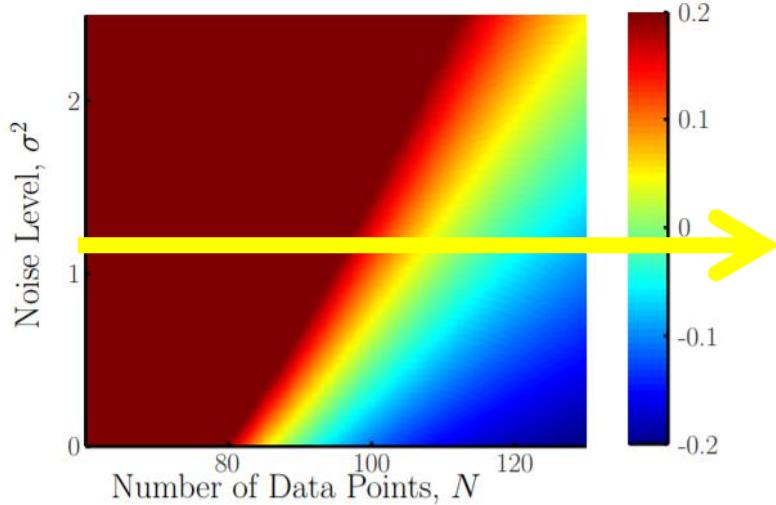


# 过拟合小结



Number of data points ↑	Overfitting ↓
Noise ↑	Overfitting ↑
Target complexity ↑	Overfitting ↑

# 过拟合小结



Number of data points ↑	Overfitting ↓
Noise ↑	Overfitting ↑
Target complexity ↑	Overfitting ↑

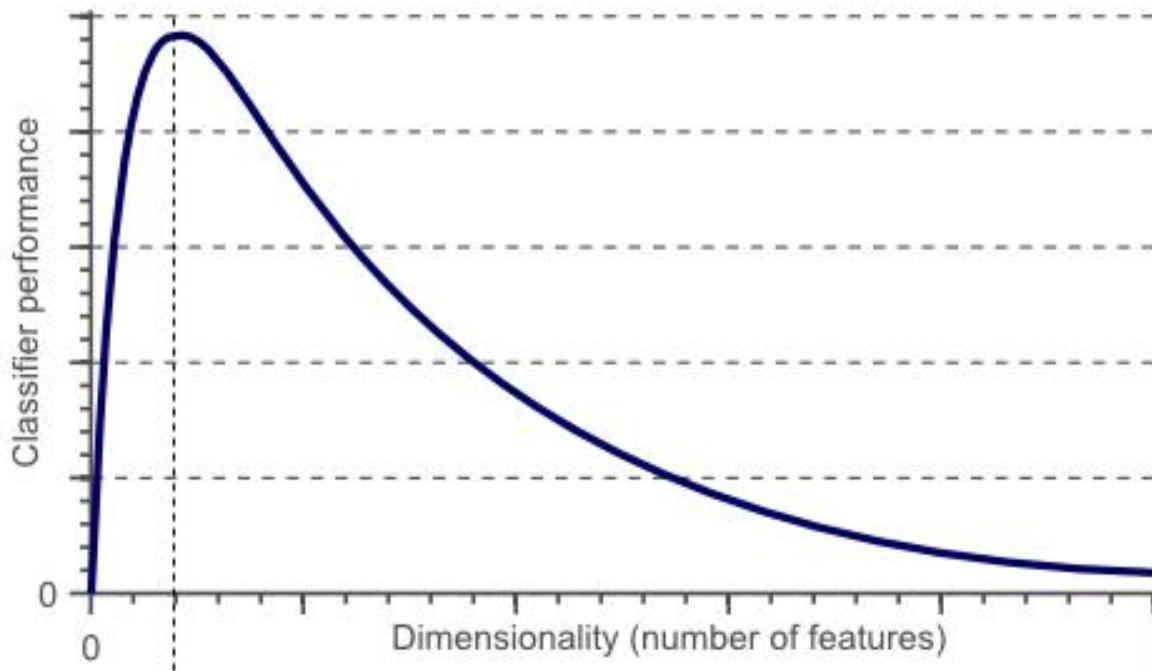
## 过拟合解决方法：

- 1) **重新清洗数据**，导致过拟合的一个原因也有可能是数据不纯导致的，如果出现了过拟合就需要我们重新清洗数据。
- 2) **增大数据的训练量**，还有一个原因就是我们用于训练的数据量太小导致的，训练数据占总数据的比例过小。
- 3) **减少特征维度,防止维灾难。**

# 5-4 维灾难

## 1. 什么是维灾难：

随着维度的增加，分类器性能逐步上升，到达某点之后，其性能便逐渐下降



## 1.什么是维灾难：

有一系列的图片，每张图片的内容可能是猫也可能是狗；我们需要构造一个分类器能够对猫、狗自动的分类。首先，要寻找到一些能够描述猫和狗的特征，这样我们的分类算法就可以利用这些特征去识别物体。猫和狗的皮毛颜色可能是一个很好的特征，考虑到红绿蓝构成图像的三基色，因此用图片三基色各自的平均值得上方便直观。这样就有了一个简单的Fisher分类器：

```
If 0.5*red + 0.3*green + 0.2*blue > 0.6 : return cat;  
else return dog;
```

## 1.什么是维灾难：

使用颜色特征可能无法得到一个足够准确的分类器，如果是这样的话，我们不妨加入一些诸如图像纹理(图像灰度值在其X、Y方向的导数dx、dy)，就有5个特征(Red、Blue、Green、dx、dy)来设计我们的分类器：

也许分类器准确率依然无法达到要求，加入更多的特征，比如颜色、纹理的统计信息等等，如此下去，可能会得到上百个特征。那是不是我们的分类器性能会随着特征数量的增加而逐步提高呢？答案也许有些让人沮丧，事实上，当特征数量达到一定规模后，分类器的性能是在下降的。**随着维度(特征数量)的增加，分类器的性能却下降了。**

## 5-4 维灾难

### 2. 维数灾难与过拟合：

我们假设猫和狗图片的数量是有限的(样本数量总是有限的)，假设有10张图片，接下来我们就用这仅有的10张图片来训练我们的分类器。



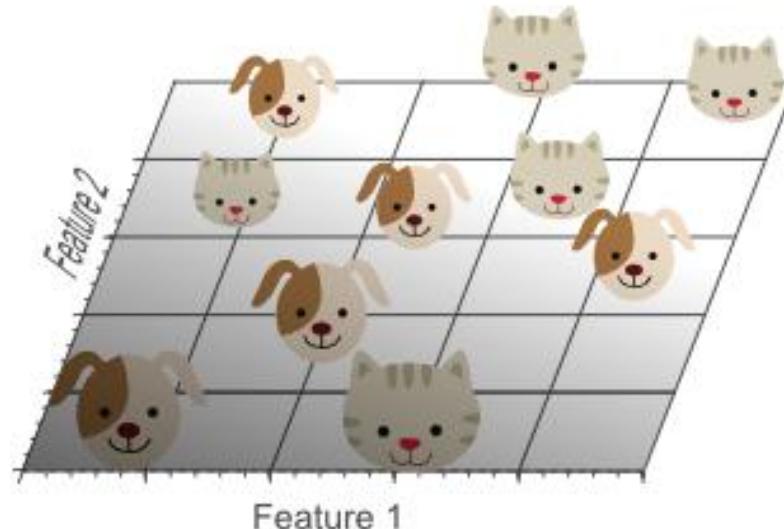
单一特征的分类器，在训练集上表现并不好

# 5-4 维灾难

## 2. 维数灾难与过拟合：

增加一个特征，比如绿色，这样特征维数扩展到了2维：

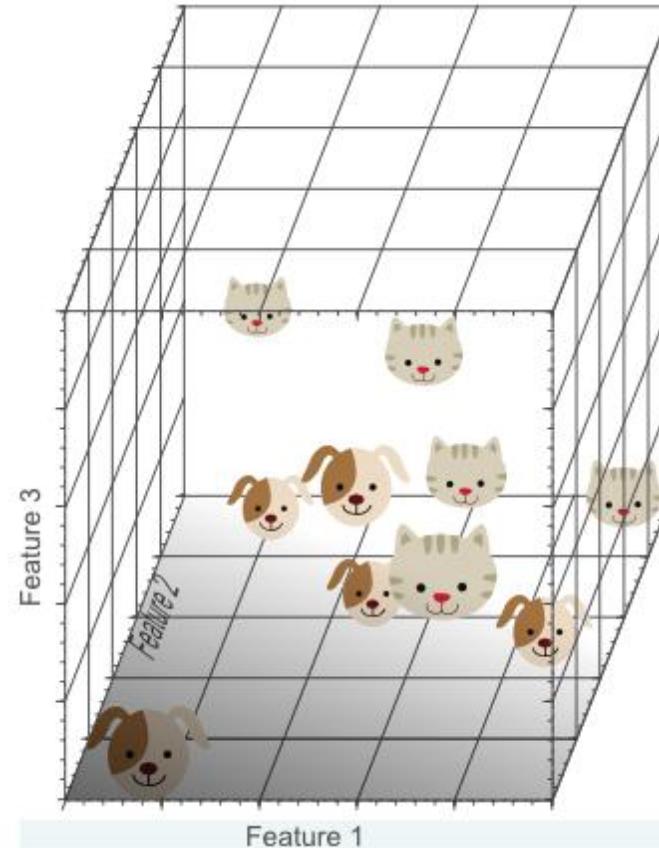
增加一个特征后，我们依然无法找到一条简单的直线将它们有效分类



# 5-4 维灾难

## 2. 维数灾难与过拟合：

再增加一个特征，比如蓝色，扩展到3维特征空间：

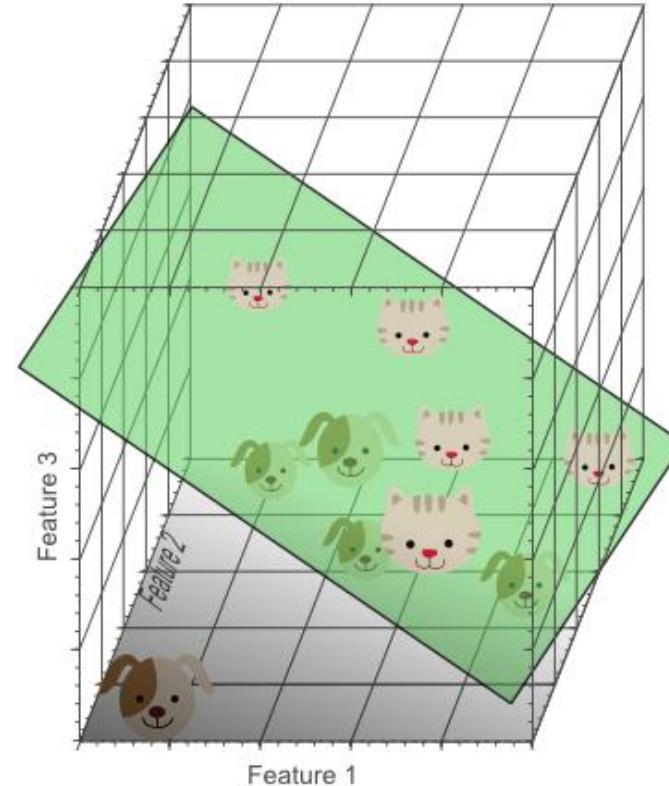


## 5-4 维灾难

### 2. 维数灾难与过拟合：

在3维特征空间中，我们很容易找到一个分类平面，能够在训练集上有效的将猫和狗进行分类：

在高维空间中，我们似乎能得到更优的分类器性能。



## 2.维数灾难与过拟合：

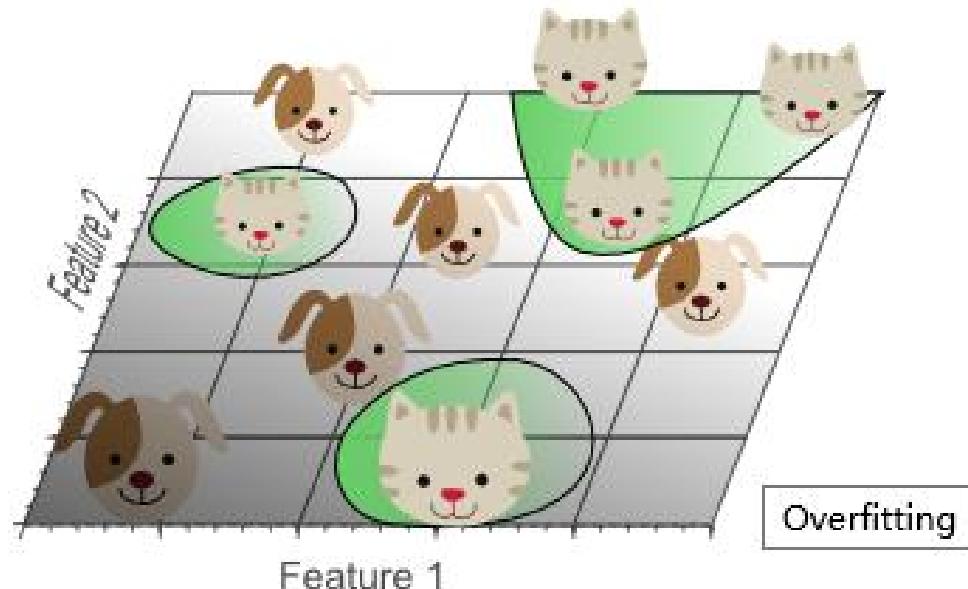
从1维到3维，给我们的感觉是：维数越高，分类性能越优。然而，维数过高将导致一定的问题：在一维特征空间下，我们假设一个维度的宽度为5个单位，这样样本密度为 $10/5=2$ ;在二维特征空间下，10个样本所分布的空间大小 $5*5=25$ ，这样样本密度为 $10/25=0.4$ ;在三维特征空间下，10个样本分布的空间大小为 $5*5*5=125$ ，样本密度就为 $10/125=0.08$ .

# 5-4 维灾难

## 2. 维数灾难与过拟合：

如果继续增加特征数量，随着维度的增加，样本将变得越来越稀疏，在这种情况下，也更容易找到一个超平面将目标分开。然而，如果我们将高维空间向低维空间投影，高维空间隐藏的问题将会显现出来：

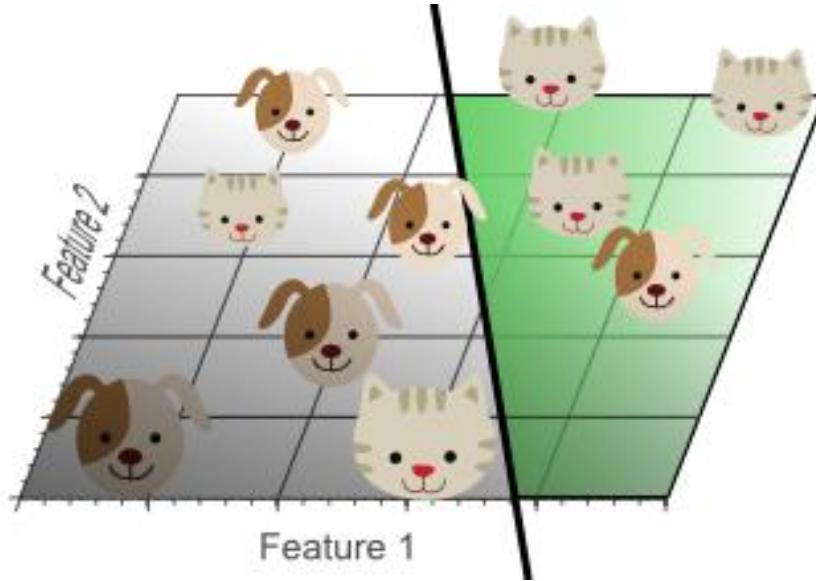
**过多的特征导致的过拟合现象：**训练集上表现良好，但是对新数据缺乏泛化能力。



# 5-4 维灾难

## 2. 维数灾难与过拟合：

高维空间训练形成的线性分类器，相当于在低维空间的一个复杂的非线性分类器，这种分类器过多的强调了训练集的准确率甚至于对一些错误/异常的数据也进行了学习，而正确的数据却无法覆盖整个特征空间。为此，这样得到的分类器在对新数据进行预测时将会出现错误。这种现象称之为过拟合，同时也是维灾难的直接体现。

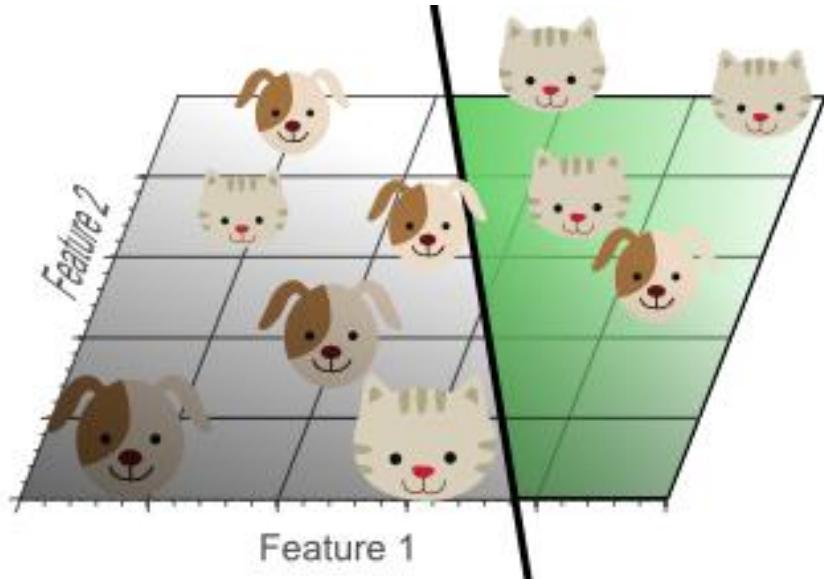


用2个特征代替三个特征进行分类器的学习：尽管训练集上分类准确率不如3维下的高，但是具备更好的泛化能力

# 5-4 维灾难

## 2. 维数灾难与过拟合：

简单的线性分类器在训练数据上的表现不如非线性分类器，但由于线性分类器的学习过程中对噪声没有对非线性分类器敏感，因此对新数据具备更优的泛化能力。换句话说，通过使用更少的特征，避免了维数灾难的发生（也即避免了高维情况下的过拟合）



用2个特征代替三个特征进行分类器的学习：尽管训练集上分类准确率不如3维下的高，但是具备更好的泛化能力

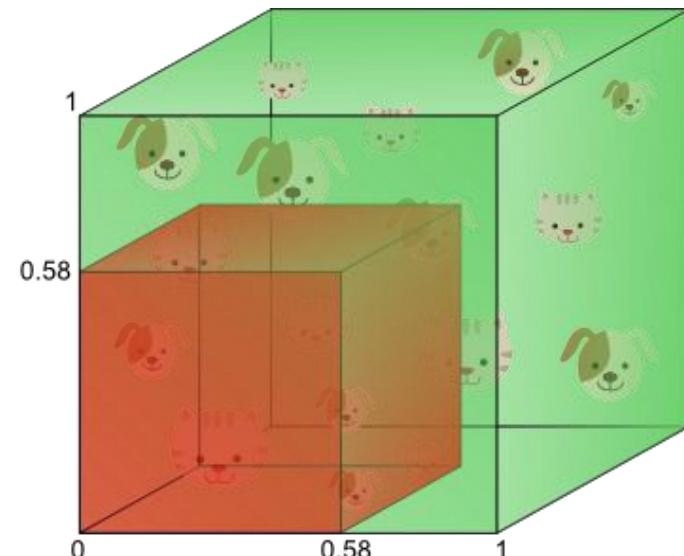
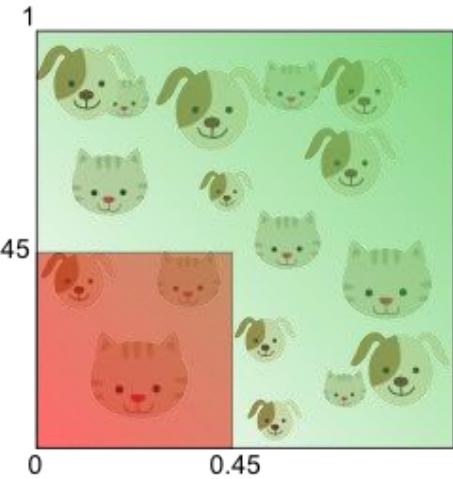
## 2.维数灾难与过拟合：

由于高维而带来的数据稀疏性问题：假设有一个特征，它的取值范围D在0到1之间均匀分布，并且对狗和猫来说其值都是唯一的，我们现在利用这个特征来设计分类器。如果我们的训练数据覆盖了取值范围的20%(e.g 0到0.2)，那么所使用的训练数据就占总样本量的20%。上升到二维情况下，覆盖二维特征空间20%的面积，则需要在每个维度上取得45%的取值范围。在三维情况下，要覆盖特征空间20%的体积，则需要在每个维度上取得58%的取值范围...在维度接近一定程度时，要取得同样的训练样本数量，则几乎要在每个维度上取得接近100%的取值范围，或者增加总样本数量，但样本数量也总是有限的。

# 5-4 维灾难

## 2. 维数灾难与过拟合：

如果一直增加特征维数，由于样本分布越来越稀疏，如果要避免过拟合的出现，就不得不持续增加样本数量。

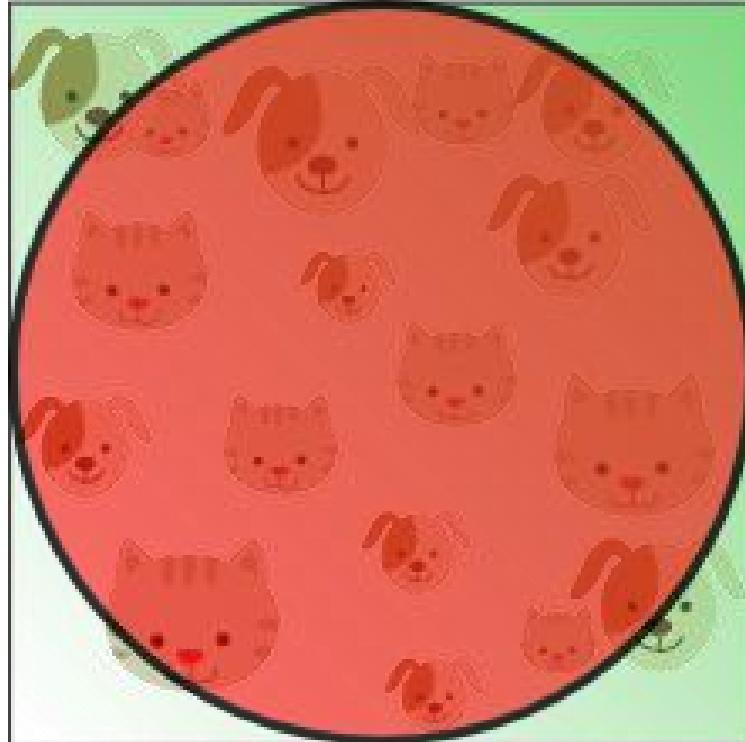


## ■ 5-4 维灾难

### 2. 维数灾难与过拟合：

数据在高维空间的中心比在边缘区域具备更大的稀疏性，数据更倾向于分布在空间的边缘区域：

不属于单位圆的训练样本比搜索空间的中心更接近搜索空间的角点。这些样本很难分类，因为它们的特征值差别很大（例如，单位正方形的对角的样本）。



## 2.维数灾难与过拟合：

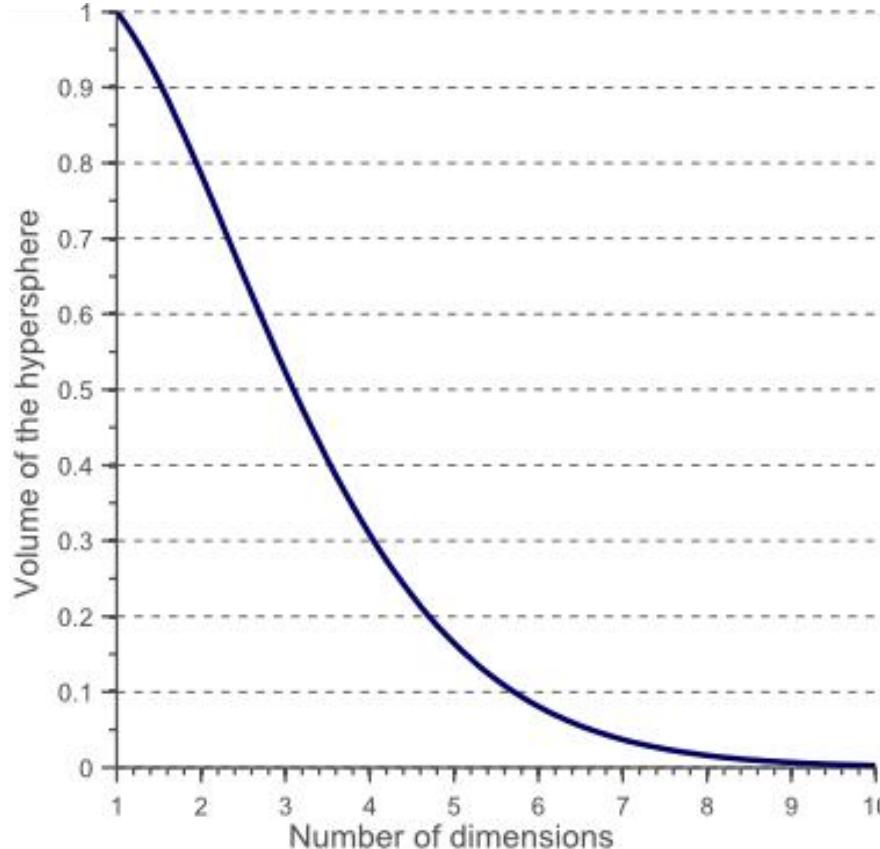
一个有趣的问题是，当我们增加特征空间的维度时，圆（超球面）的体积如何相对于正方形（超立方体）的体积发生变化。尺寸d的单位超立方体的体积总是 $1^d = 1$ 。尺寸d和半径0.5的内切超球体的体积可以计算为：

$$V(d) = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2} + 1\right)} 0.5^d$$

# 5-4 维灾难

## 2. 维数灾难与过拟合：

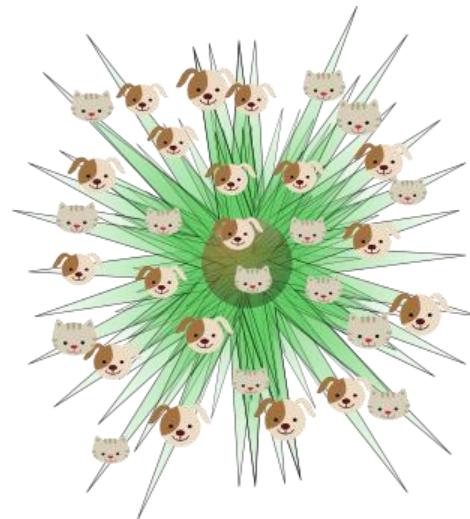
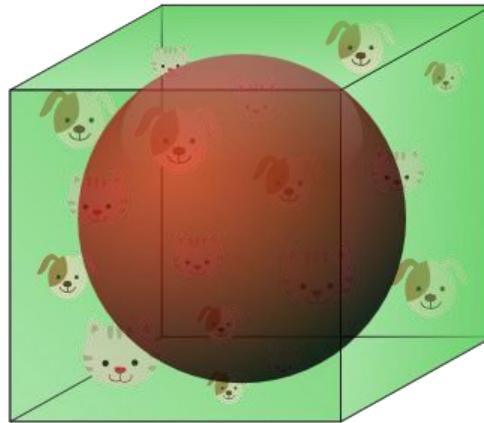
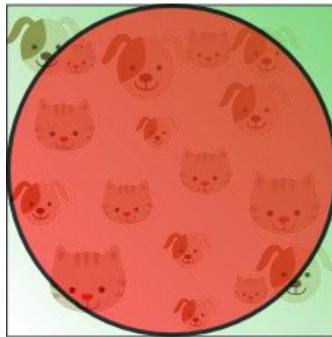
在高维空间中，大多数训练数据驻留在定义特征空间的超立方体的角落中。如前所述，特征空间角落中的实例比围绕超球体质心的实例难以分类。



# 5-4 维灾难

## 2. 维数灾难与过拟合：

在高维空间中，大多数训练数据驻留在定义特征空间的超立方体的角落中。如前所述，特征空间角落中的实例比围绕超球体质心的实例难以分类：



an 8D hypercube which has  $2^8 = 256$  corners

## 2.维数灾难与过拟合：

事实证明，许多事物在高维空间中表现得非常不同。例如，如果你选择一个单位平方（ $1 \times 1$  平方）的随机点，它将只有大约0.4%的机会位于小于0.001的边界（换句话说，随机点将沿任何维度“极端”这是非常不可能的）。但是在一个10000维单位超立方体（ $1 \times 1 \times 1$  立方体，有1万个1）中，这个概率大于99.999999%。高维超立方体中的大部分点都非常靠近边界。更难区分的是：如果你在一个单位正方形中随机抽取两个点，这两个点之间的距离平均约为0.52。如果在单位三维立方体中选取两个随机点，则平均距离将大致为0.66。但是在100万维的超立方体中随机抽取两点呢？那么平均距离将是大约408.25（大约 $1,000,000 / 6$ ）！

非常违反直觉：当两个点位于相同的单位超立方体内时，两点如何分离？这个事实意味着高维数据集有可能非常稀疏：大多数训练实例可能彼此远离。当然，这也意味着一个新实例可能离任何训练实例都很远，这使得预测的可信度表现得比在低维度数据中要来的差。**训练集的维度越多，过度拟合的风险就越大。**

理论上讲，维度灾难的一个解决方案可能是增加训练集的大小以达到足够密度的训练实例。不幸的是，在实践中，达到给定密度所需的训练实例的数量随着维度的数量呈指数增长。如果只有100个特征（比MNIST问题少得多），那么为了使训练实例的平均值在 $0.1$ 以内，需要比可观察宇宙中的原子更多的训练实例，假设它们在所有维度上均匀分布。

对于8维超立方体，大约98%的数据集中在其256个角上。结果，当特征空间的维度达到无穷大时，从采样点到质心的最小和最大欧几里得距离的差与最小距离本身只比趋于零：

$$\lim_{d \rightarrow \infty} \frac{dist_{\max} - dist_{\min}}{dist_{\min}} \rightarrow 0$$

# 总结

距离测量开始失去其在高维空间中测量的有效性,由于分类器取决于这些距离测量,因此在较低维空间中分类通常更容易, 其中较少特征用于描述感兴趣对象。

如果理论无限数量的训练样本可用, 则维度的诅咒不适用, 我们可以简单地使用无数个特征来获得完美的分类。训练数据的大小越小, 应使用的功能就越少。如果N个训练样本足以覆盖单位区间大小的1D特征空间, 则需要 $N^2$ 个样本来覆盖具有相同密度的2D特征空间, 并且在3D特征空间中需要 $N^3$ 个样本。换句话说, **所需的训练实例数量随着使用的维度数量呈指数增长。**



传智播客旗下高端IT教育品牌