基础知识
Basic Knowledge






目 录 (CONTENT)

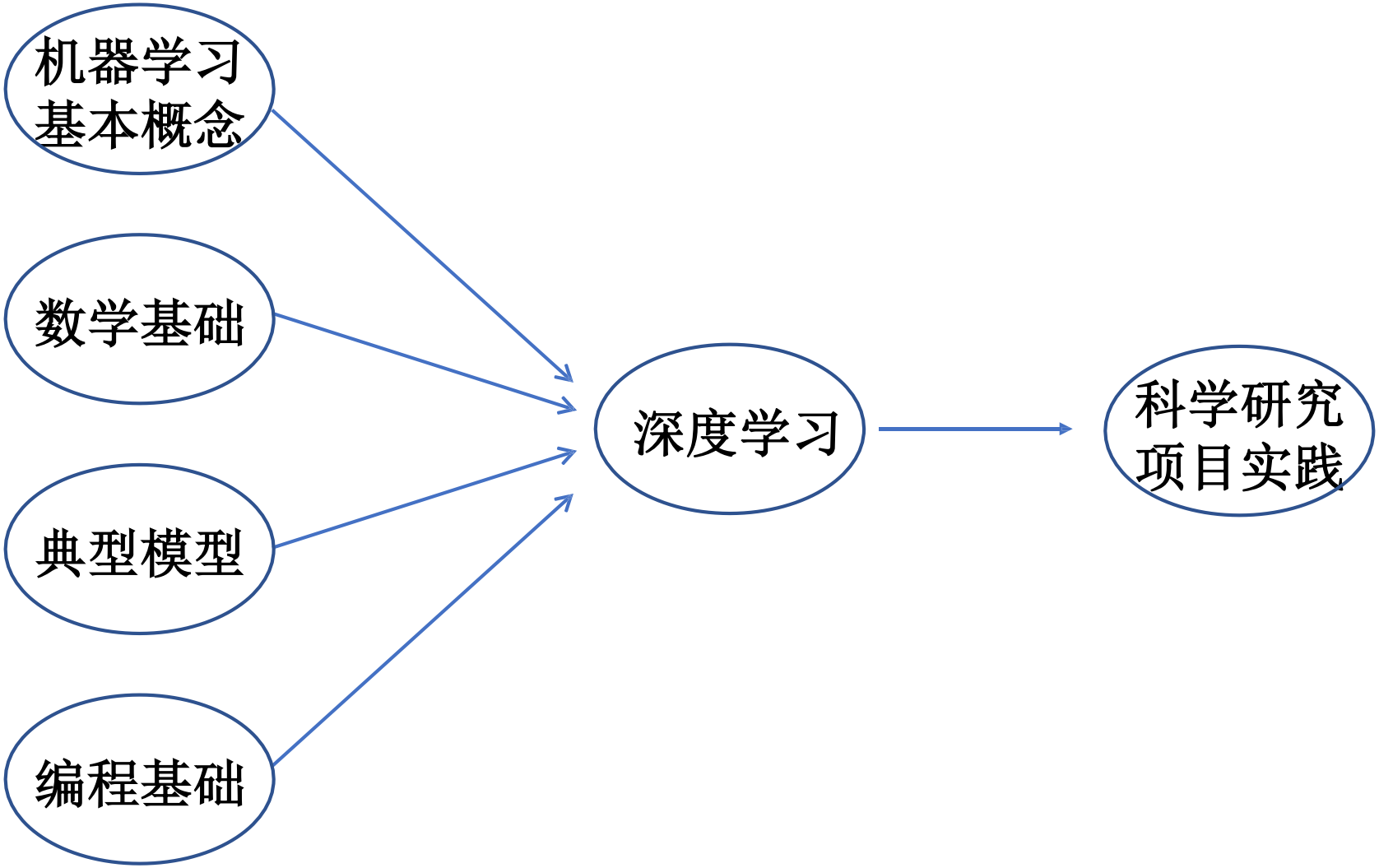
大连理工大学 Dalian University Of Technology

大连理工大学
Dalian University Of Technology

01

PART ONE

引言
Introduction


机器学习
基本概念
数学基础
典型模型
编程基础
深度学习
科学研究
项目实践

大连理工大学 Dalian University Of Technology

02

PART TWO

机器学习基本概念
Basic ML Concepts

什么是机器学习？

大连理工大学 Dalian University Of Technology

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

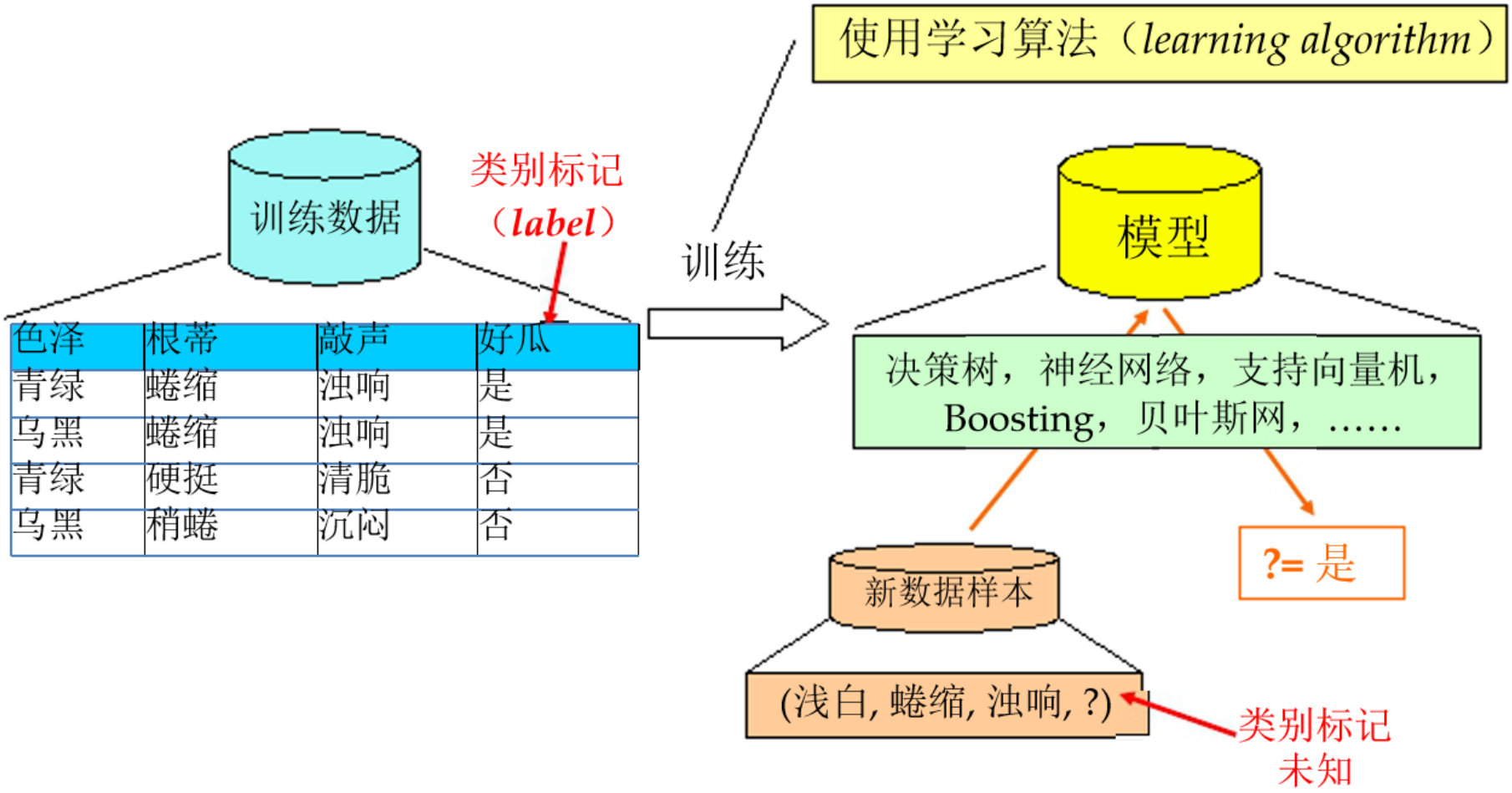
近期机器学习相关ACM图灵奖：

2011, Leslie Valiant, “计算学习理论”

2012, Judea Pearl, “图模型学习方法”

2018, **Geoffrey Hinton, Yoshua Bengio, Yann LeCun**, “神经网络与深度学习”

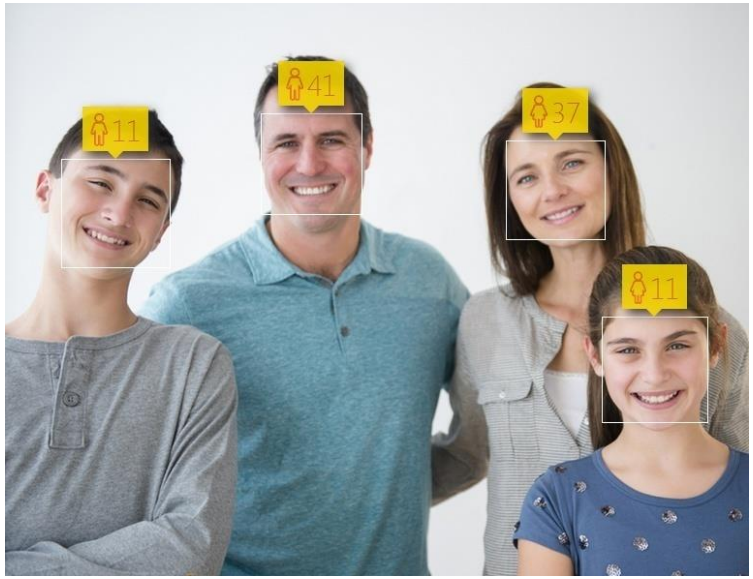
典型的机器学习过程


使用学习算法（learning algorithm）
训练数据
类别标记
（label）
训练
模型
色泽 根蒂 敲声 好瓜
青绿 蜷缩 浊响 是
乌黑 蜷缩 浊响 是
青绿 硬挺 清脆 否
乌黑 稍蜷 沉闷 否
决策树，神经网络，支持向量机，
Boosting，贝叶斯网，……
新数据样本
?= 是
(浅白, 蜷缩, 浊响, ?)
类别标记
未知

机器学习应用

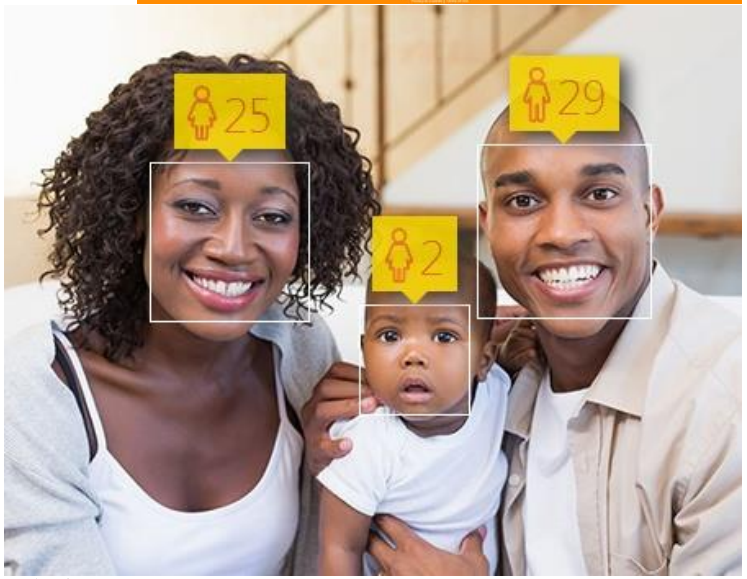
大连理工大学 Dalian University Of Technology

➢**how-old.net**



包含子问题：

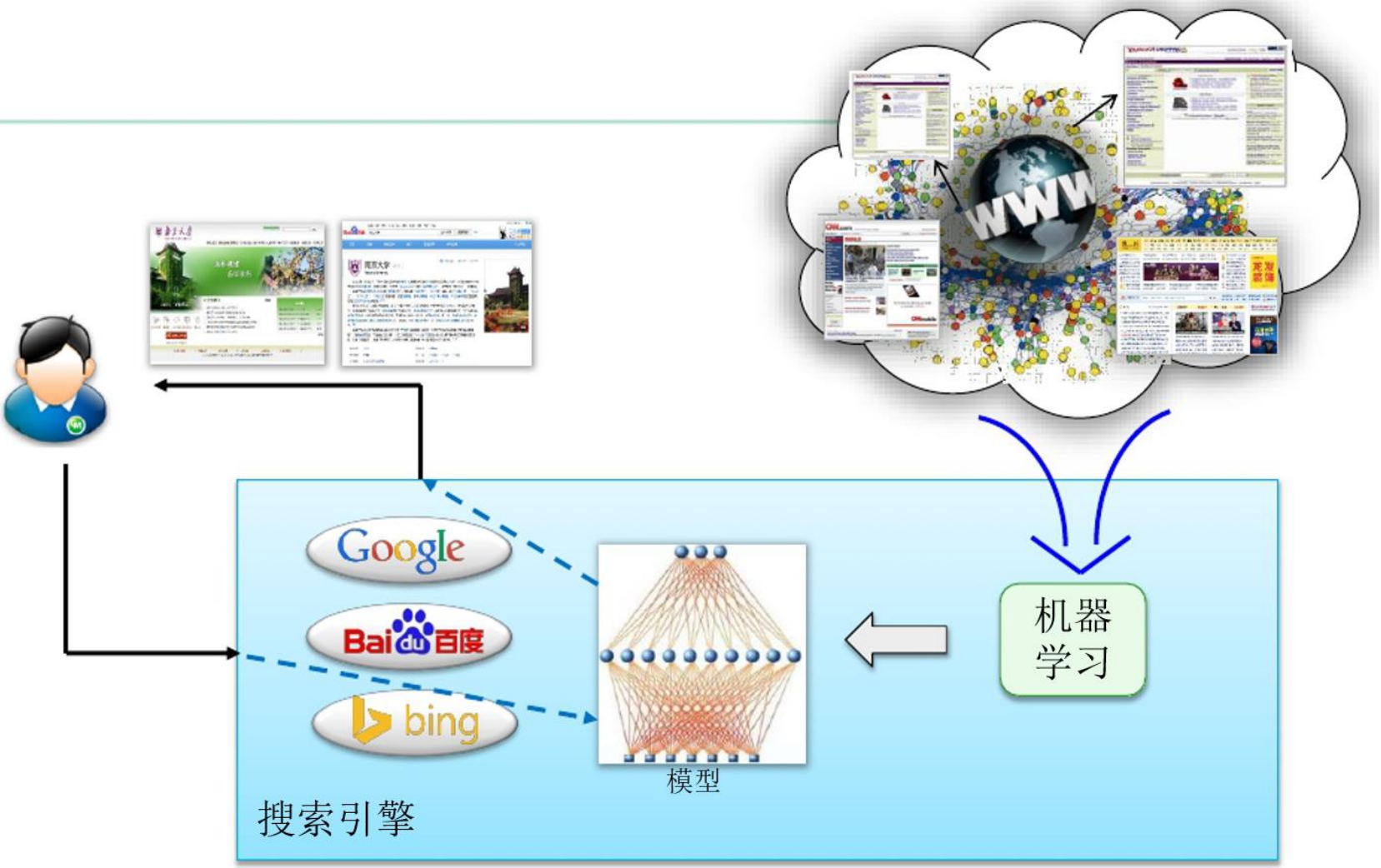
a) 人脸检测，Face Detection
b) 人脸对齐，Face Alignment
c) 年龄分类，Age Classification



其他子问题：

a) 人脸识别，Face Recognition
b) 性别识别，Gender Recognition
c) 表情识别，Expression Recognition
d) 种族识别，Race Recognition

机器学习应用(网络搜索)



机器学习技术正在支撑着各种搜索引擎

机器学习应用(自动驾驶)

车载摄像头

车载雷达

机器学习

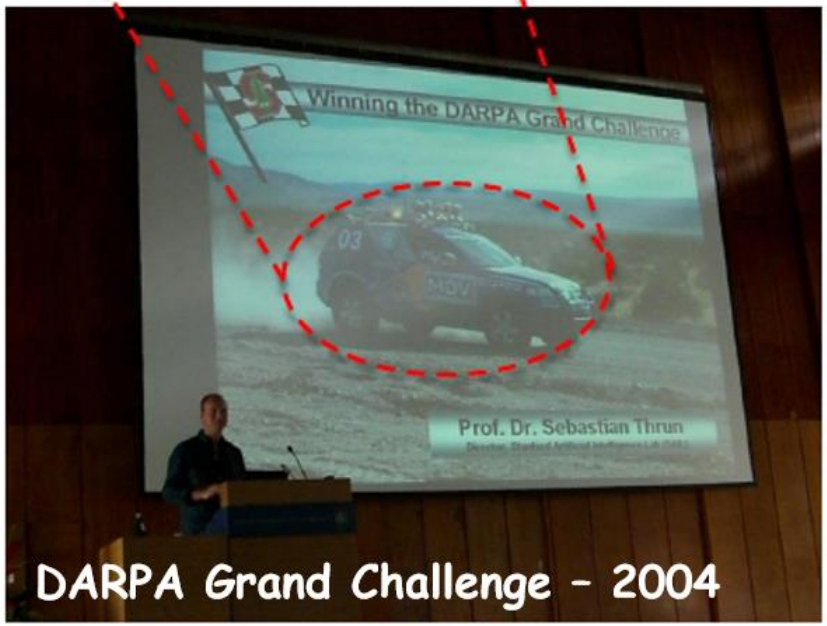
模型

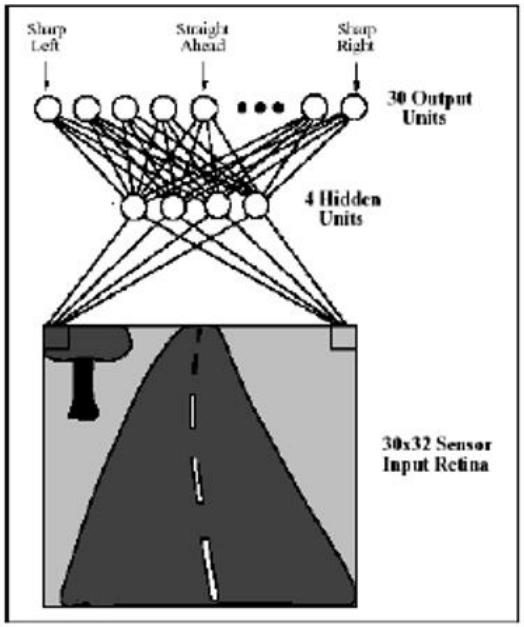
方向盘旋转幅度

油门幅度

刹车幅度

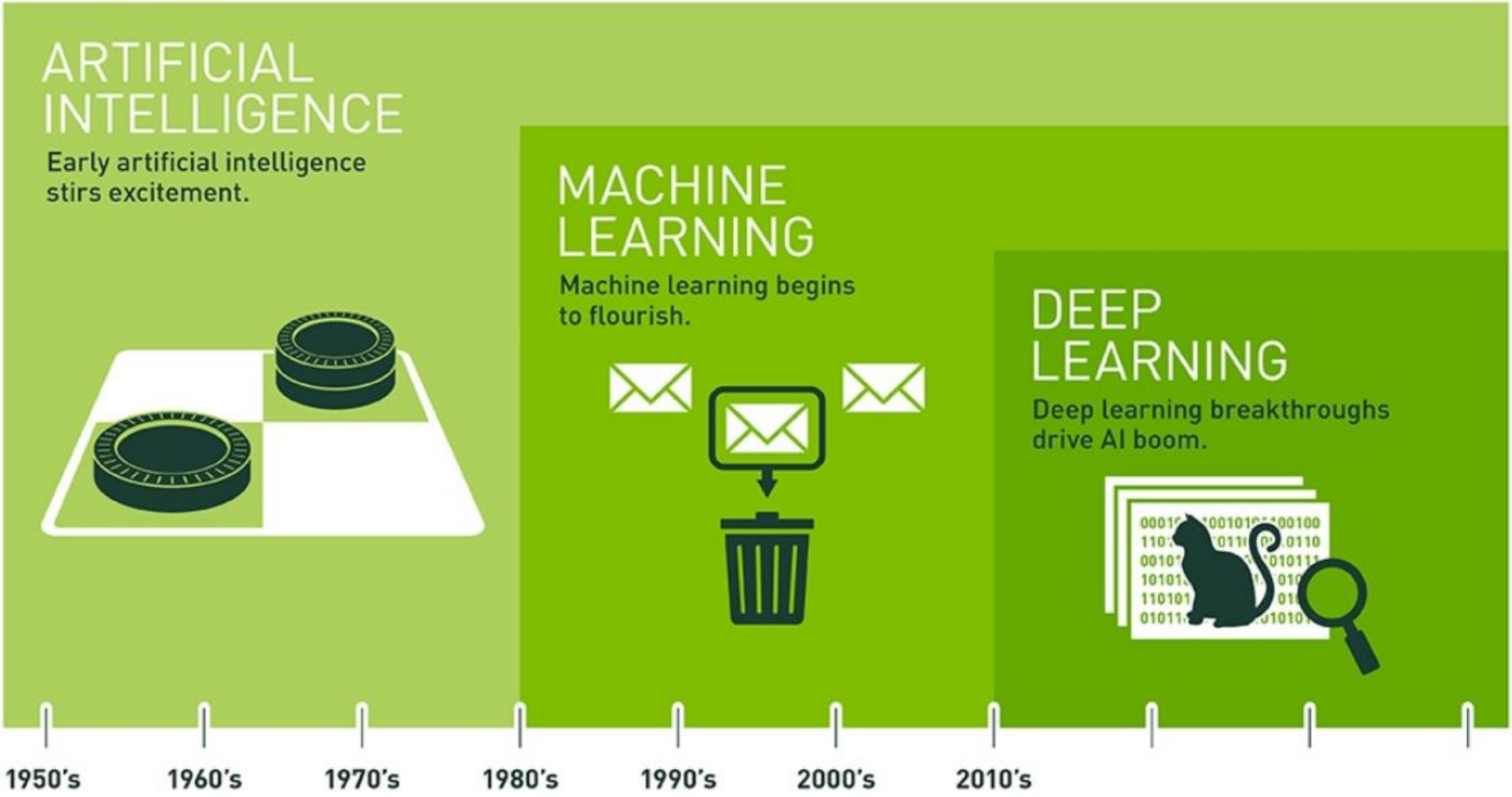
控制汽车





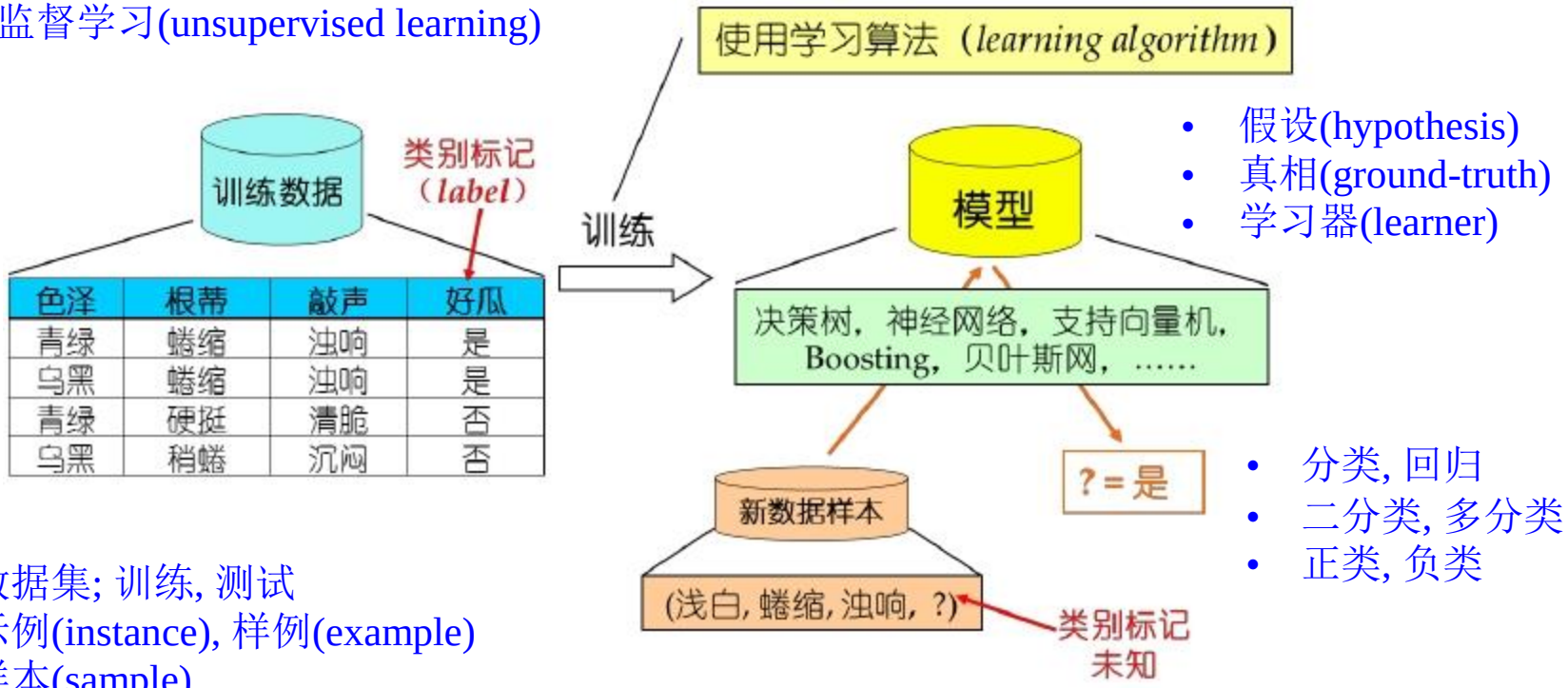
美国在20世纪80年代就开始研究基于机器学习的汽车自动驾驶技术

人工智能，机器学习，深度学习



基本术语

- 监督学习(supervised learning)
- 无监督学习(unsupervised learning)



色泽	根蒂	敲声	好瓜
青绿	蜷缩	浊响	是
乌黑	蜷缩	浊响	是
青绿	硬挺	清脆	否
乌黑	稍蜷	沉闷	否

- 假设(hypothesis)
- 真相(ground-truth)
- 学习器(learner)

- 分类, 回归
- 二分类, 多分类
- 正类, 负类

- 数据集; 训练, 测试
- 示例(instance), 样例(example)
- 样本(sample)
- 属性(attribute), 特征(feature); 属性值
- 属性空间, 样本空间, 输入空间
- 特征向量(feature vector)
- 标记空间, 输出空间

- 未见样本(unseen instance)
- 未知“分布”
- 独立同分布(i.i.d.)
- 泛化**(generalization)**

➢ 参考《机器学习》，周志华著，p.1-3

假设(Hypothesis)->假设空间

大连理工大学 Dalian University Of Technology

■ **假设(Hypothesis)**

学到模型对应了数据的某种潜在的规律，因此亦称“假设(hypothesis)”，这种潜在规律自身被成为“真相(ground truth）。

■ **假设空间(Hypothesis Space)**

编号	色泽	根蒂	敲声	好瓜
1	青绿	蜷缩	浊响	是
2	乌黑	蜷缩	浊响	是
3	青绿	硬挺	清脆	否
4	乌黑	稍蜷	沉闷	否

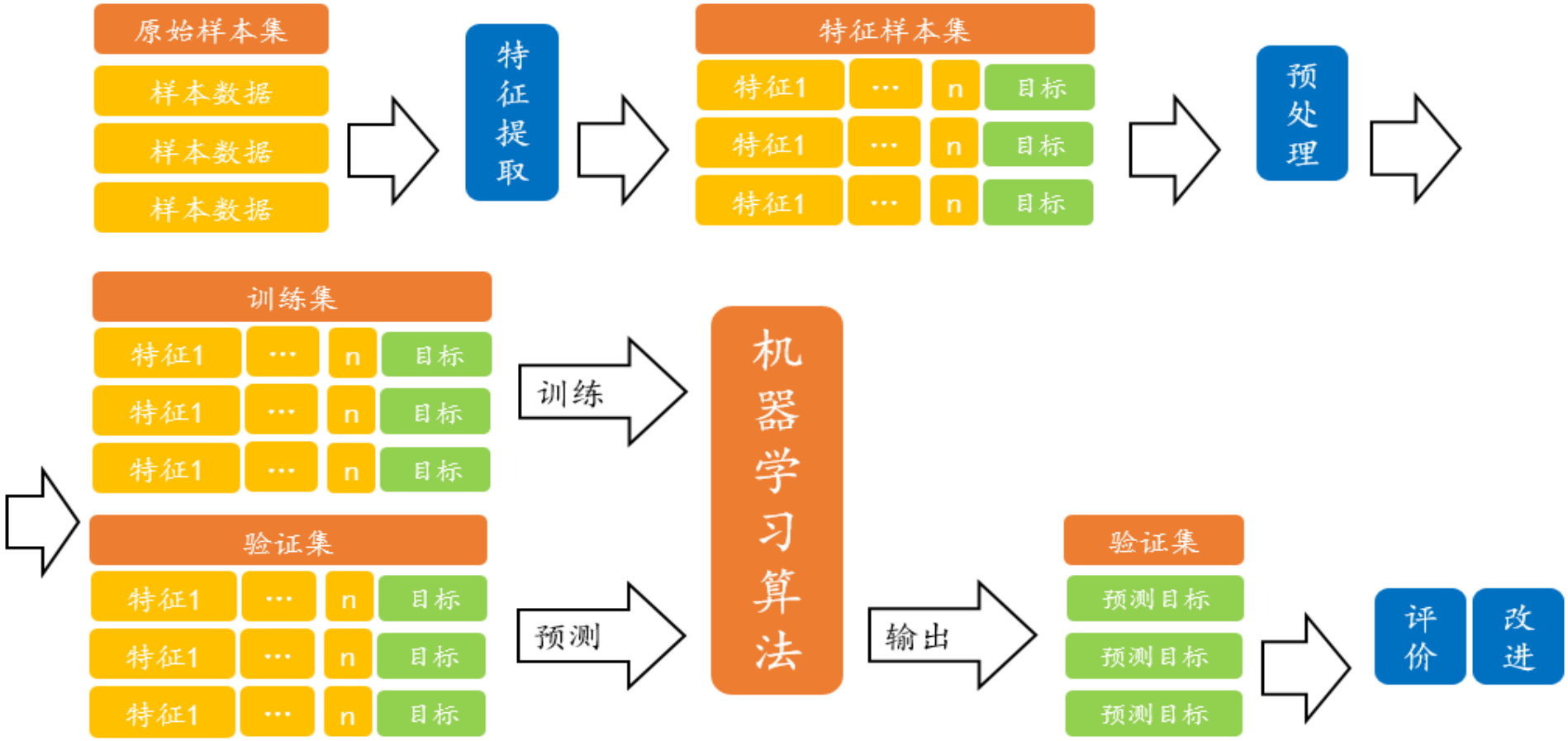
(色泽=?)∧(根蒂=?)∧(敲声=?)↔好瓜

学习过程 → 在所有假设(hypothesis)组成的空间中进行搜索的过程
目标: 找到与训练集“匹配”(fit)的假设

■ **版本空间(Version Space):** 与训练集一致的假设集合

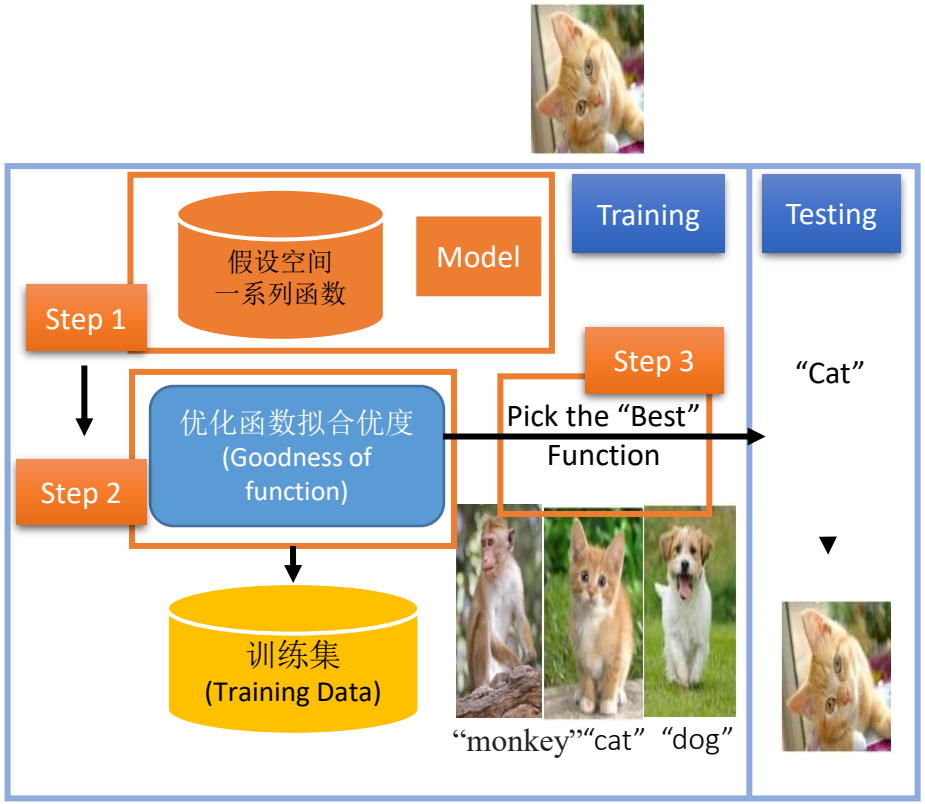
➢ **如何理解假设空间与版本空间？**
https://www.cnblogs.com/HongjianChen/p/8383816.html

机器学习实施过程


原始样本集
样本数据
样本数据
样本数据
特征提取
特征样本集
特征1 … n 目标
特征1 … n 目标
特征1 … n 目标
预处理
训练集
特征1 … n 目标
特征1 … n 目标
特征1 … n 目标
验证集
特征1 … n 目标
特征1 … n 目标
特征1 … n 目标
训练
预测
机器学习算法
输出
验证集
预测目标
预测目标
预测目标
评价
改进

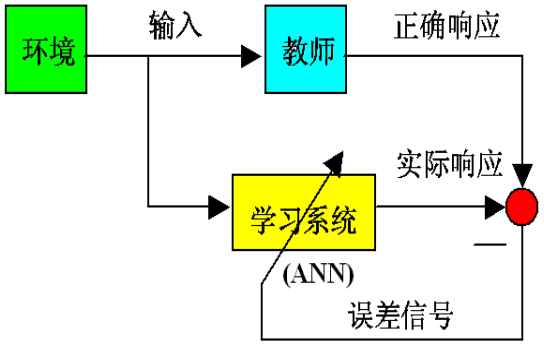
机器学习与数学函数

- 机器学习≈寻找函数“ f ”
  - Image recognition

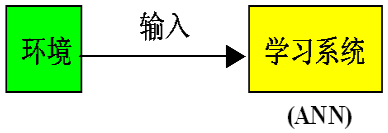


监督学习/无监督学习

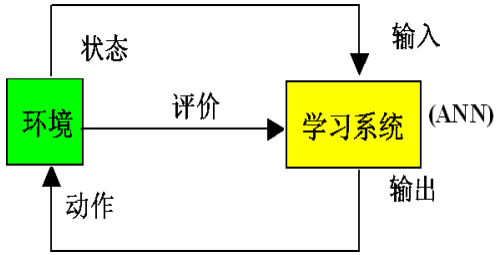
- **监督学习**(Supervised Learning)：监督学习是从标记的训练数据来推断一个功能的机器学习任务。如**分类、回归**。



- **无监督学习**(Unsupervised Learning)：无监督学习的问题是，在未标记的数据中，试图找到隐藏的结构。如**聚类**、密度估计。



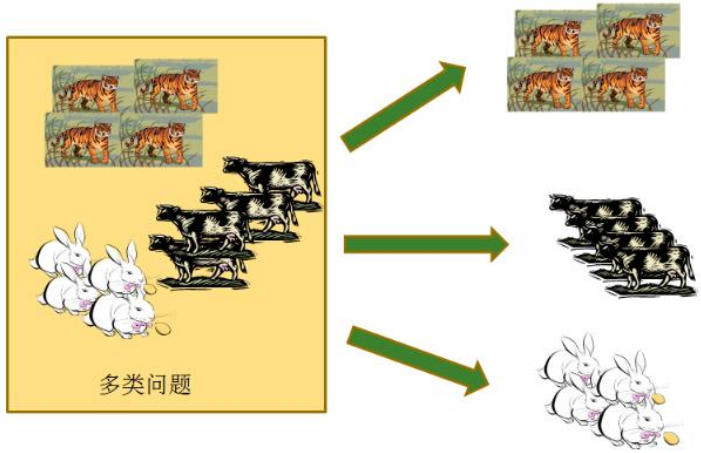
- **强化学习**(Reinforcement Learning)：强化学习是机器学习中的一个领域，强调如何基于环境而行动，以取得最大化的预期利益。



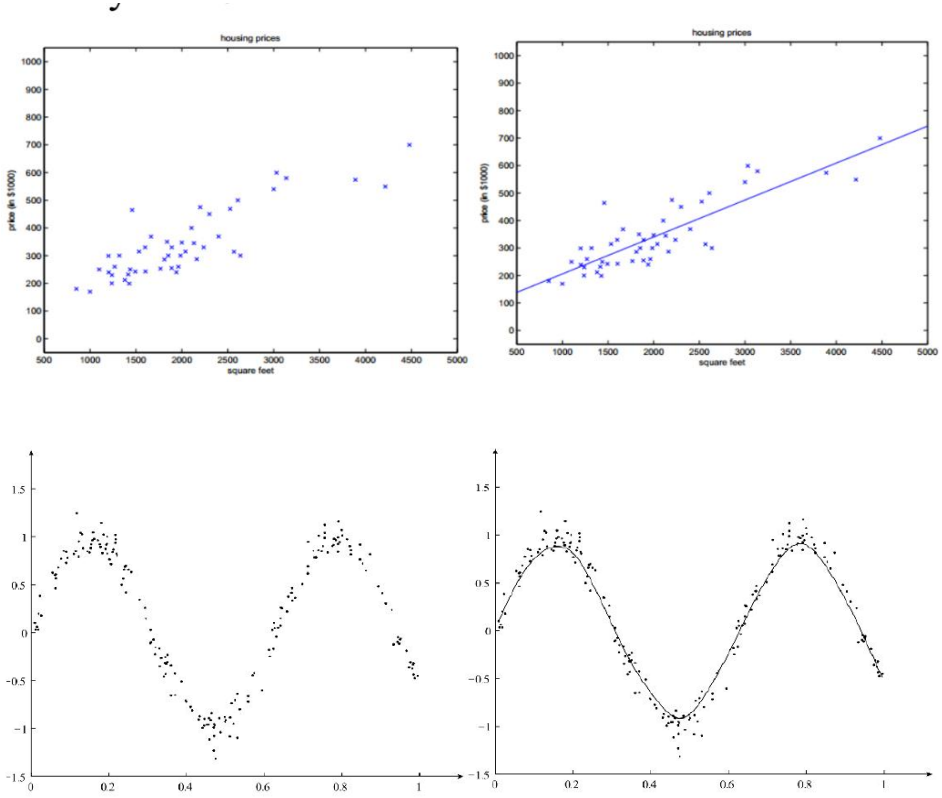
分类/回归/聚类

大连理工大学 Dalian University Of Technology

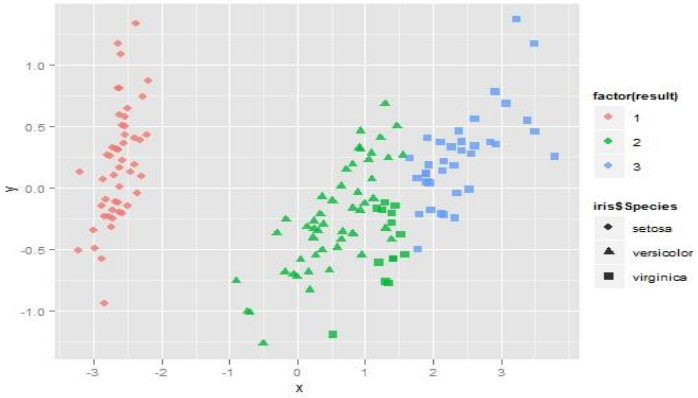
- 分类



- 回归



- 聚类



机器学习的基本过程

将数据对象进行特征(feature)化表示

给定一个数据样本集，从中学习出规律（模型）

目标：该规律不仅适用于训练数据，也适用于未知数据(称为泛化能力)

对于一个新的数据样本，利用学到的模型进行预测

样本表示

■ 向量表示法 $[x_1, \quad x_2, \ldots \quad x_n]$

➢ 天气预测：

样本：每一天

问题：如何把每天表示成一个向量？选取哪些特征？

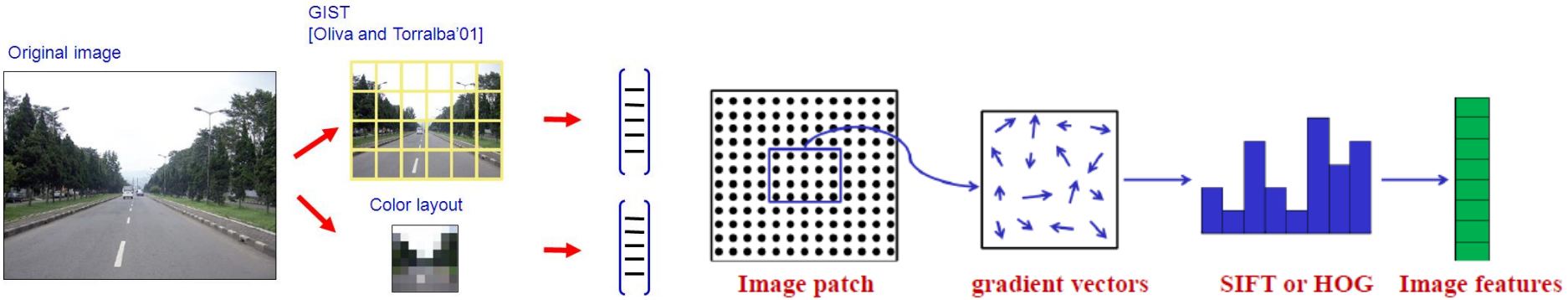
特征：温度，相对湿度，风向，风速，气压

➢ 判断好瓜坏瓜：

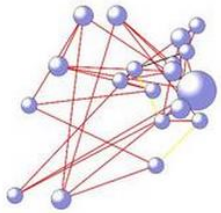
样本：每个瓜

特征：色泽，根蒂，敲声

➢ 图像识别：

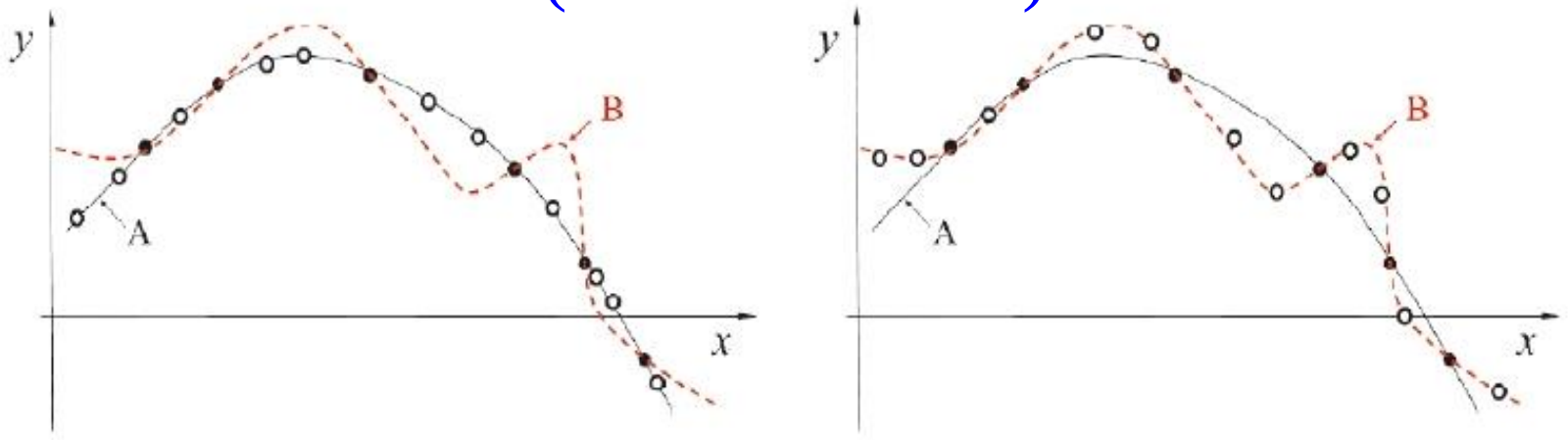


■ 图表示法



哪个算法/模型更好?

没有免费的午餐!
(No-free Lunch!)



(a) A 优于 B　　(b) B 优于 A

图 1.4 没有免费的午餐. (黑点: 训练样本; 白点: 测试样本)

NFL定理：一个算法 $\mathfrak{L}_a$ 若在某些问题上比另一个算法 $\mathfrak{L}_b$ 好，必存在另一些问题，$\mathfrak{L}_b$ 比 $\mathfrak{L}_a$ 好。

No-free Lunch(NFL)定理寓意

NFL定理的重要前提：

所有“问题”出现的机会相同、或所有问题同等重要

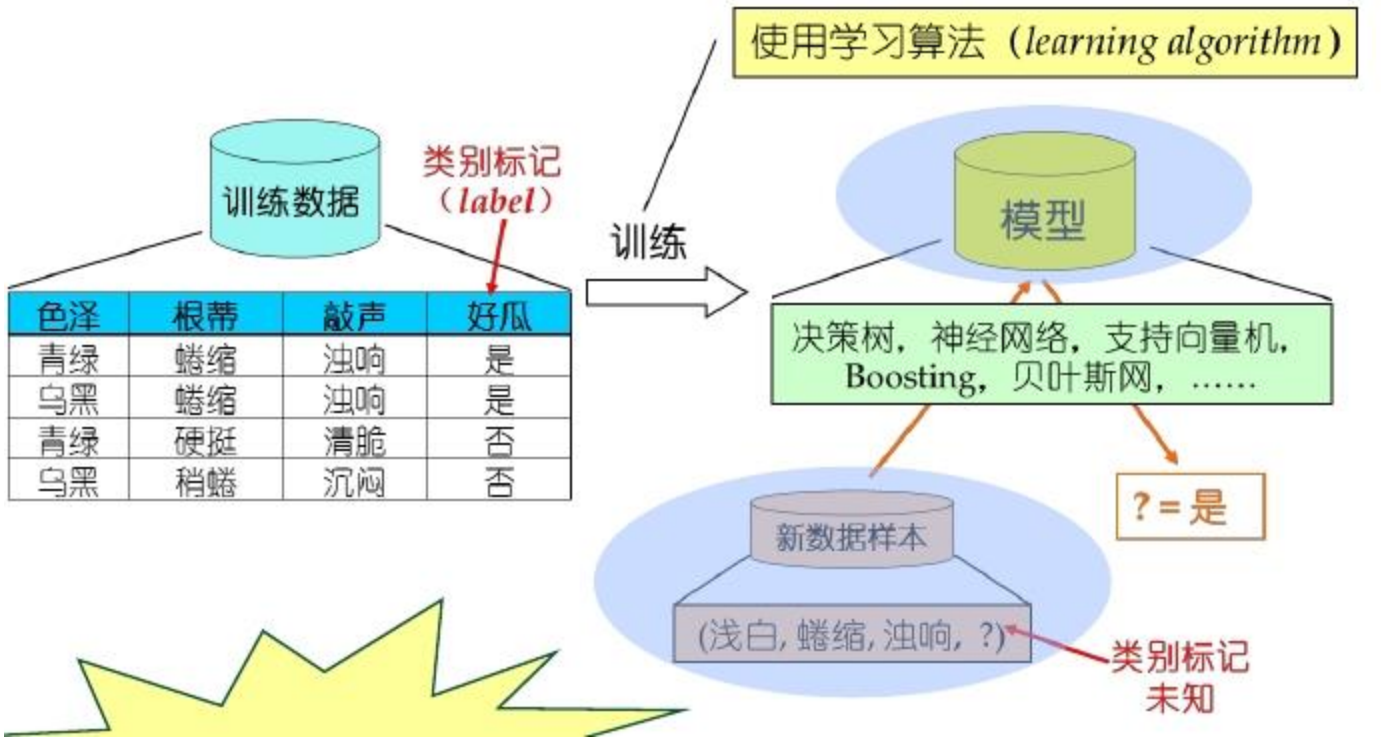
实际情形并非如此；我们通常只关注自己正在试图解决的问题

脱离具体问题，空泛地谈论“什么学习算法更好”

03
PART THREE

模型评估与选择
Model Evalution and Selection

典型的机器学习过程



色泽	根蒂	敲声	好瓜
青绿	蜷缩	浊响	是
乌黑	蜷缩	浊响	是
青绿	硬挺	清脆	否
乌黑	稍蜷	沉闷	否

泛化能力强！

能很好地适用于未知样本

例如，错误率低、精度高

然而，我们手上没有未知样本，……

误差(Error)/风险(Risk)

误差 (Error)：预测结果与真实值间的差异称为误差

泛化误差(Generalization Error)：在“未来”样本上的误差

经验误差(Empirical Error)：在训练集上的误差，亦称“训练误差”

- 泛化误差越小越好
- 经验误差是否越小越好？

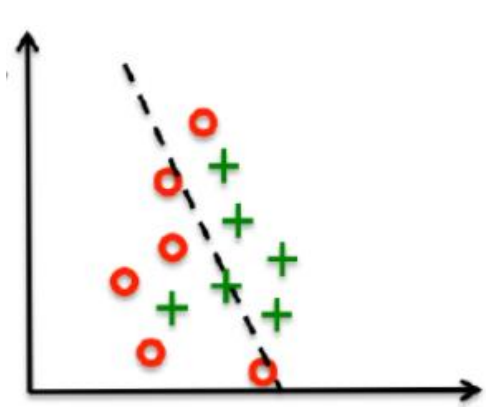
NO! 因为会出现“过拟合”(overfitting)

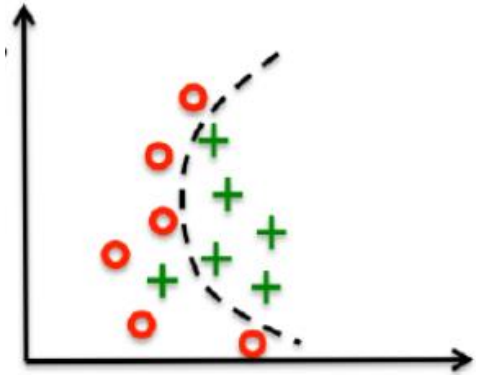
测试误差(Test Error)：在测试集上产生的误差，称为“测试误差”

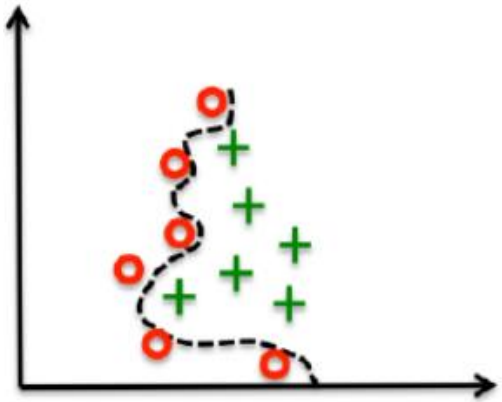
过拟合/欠拟合

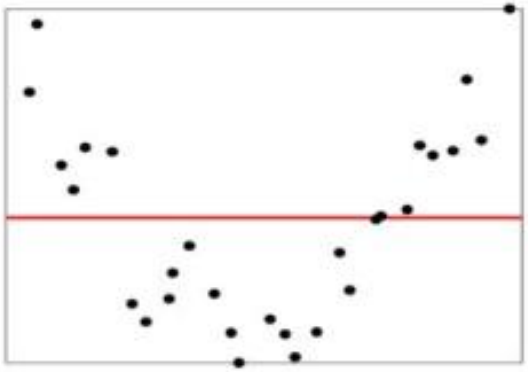
- 欠拟合：对训练样本的一般性质尚未学好
- 过拟合: 学习器把训练样本学习的“太好”，将训练样本本身的特点当做所有样本的一般性质，导致泛化性能下降

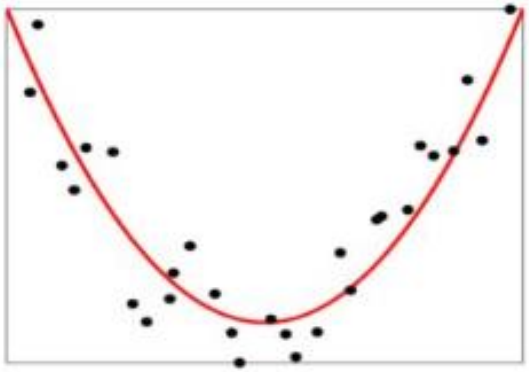
过拟合/欠拟合

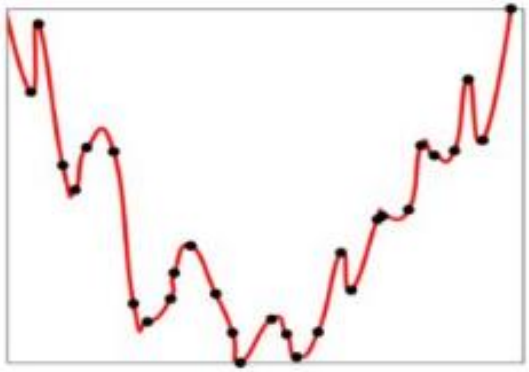












欠拟合　　拟合　　过拟合

模型选择（Model Selection）

三个关键问题：

- 如何获得测试结果？ 评估方法
- 如何评估性能优劣？ 性能度量
- 如何判断实质差别？ 比较检验

评估方法

关键：怎么获得“测试集”(test set)？

测试集应该与训练集“互斥”

常见方法：

- 留出法 (hold-out)　直接将数据集D划分为两个互斥的集合
- **交叉验证法 (cross validation)**
- 自助法 (bootstrap)

  直接以自主采样法为基础，从中有放回的采样执行m次，得到包含m个样本的训练集。从未在训练集中出现的样本则作为测试样本放入测试集中

评估方法-*k*-折交叉验证法

- *k*-fold cross validation

若 k = m，则得到“留一法”(leave-one-out, LOO)

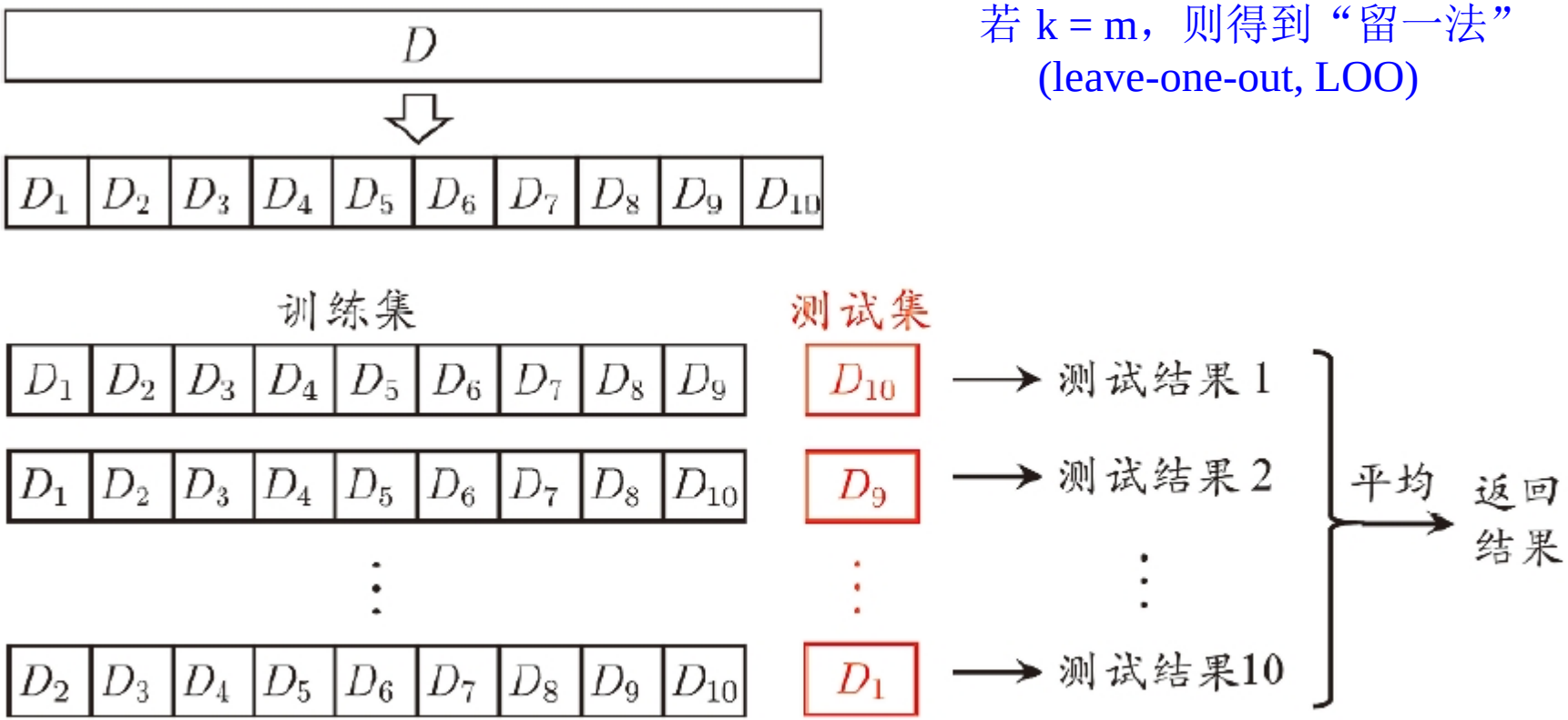


图 2.2 10 折交叉验证示意图

“调参”与最终模型

算法的参数：一般由人工设定，亦称“超参数”

模型的参数：一般由学习确定

调参过程相似：先产生若干模型，然后基于某种评估方法进行选择

参数调得好不好对性能往往对最终性能有关键影响

区别：训练集 vs. 测试集 vs. 验证集 (validation set)

算法参数选定后，要用“训练集+验证集”重新训练最终模型

性能度量

性能度量(performance measure)是衡量模型泛化能力的评价标准，反映了任务需求

使用不同的性能度量往往会导致不同的评判结果

什么样的模型是“好”的，不仅取决于算法和数据，还取决于任务需求

- 回归(regression) 任务常用均方误差：

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{x}_i\right)-y_i\right)^2$$

性能度量-错误率v.s.精度

- 错误率：

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)$$

- 精度：

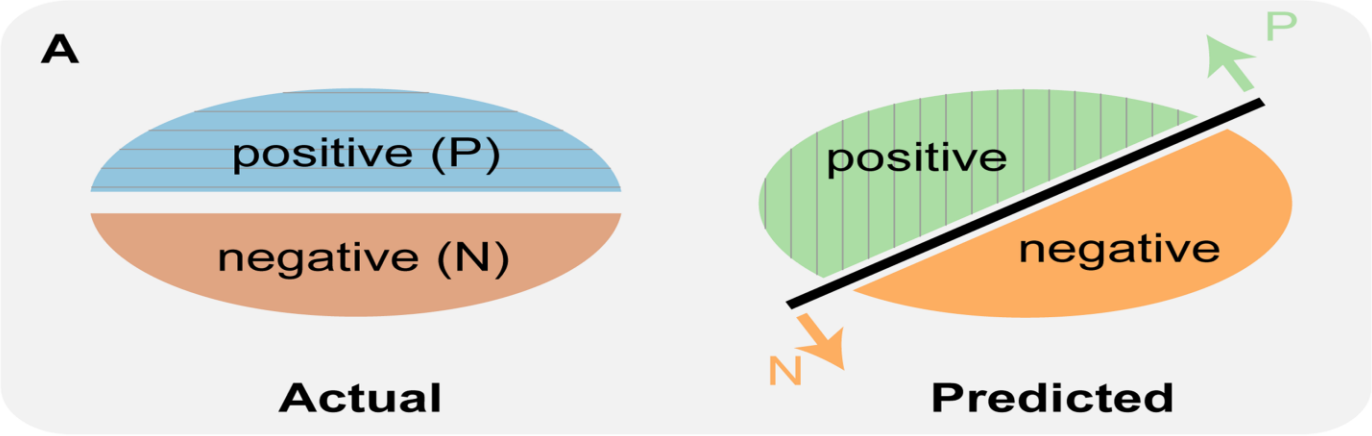
$$\begin{aligned}\mathrm{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

性能度量-错误率v.s.精度

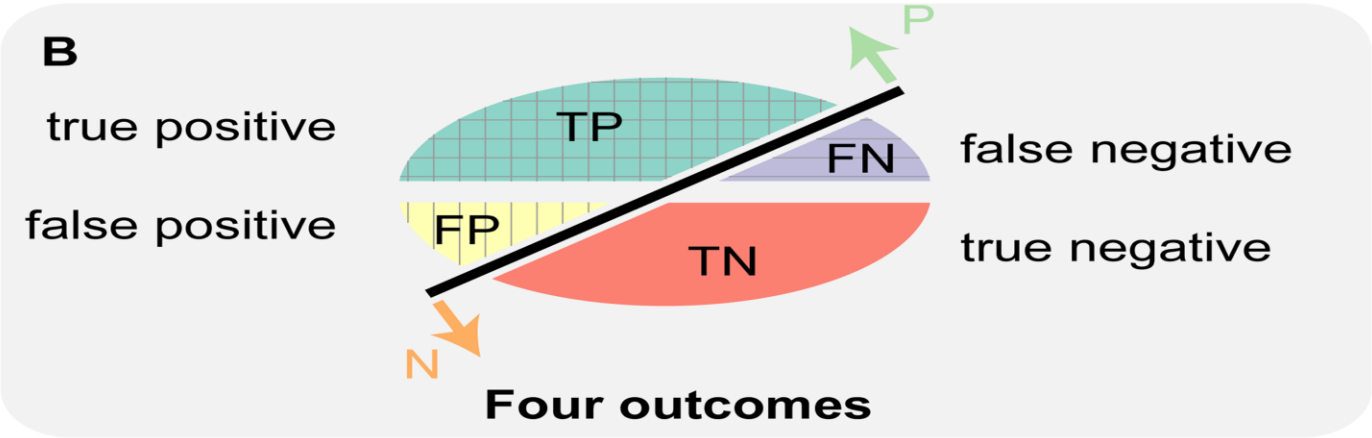
真实情况	预测结果	
	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

- 查准率（precision）：$P = \frac{TP}{TP + FP}$
- 查全率（Recall）：$R = \frac{TP}{TP + FN}$

性能度量-错误率v.s.精度



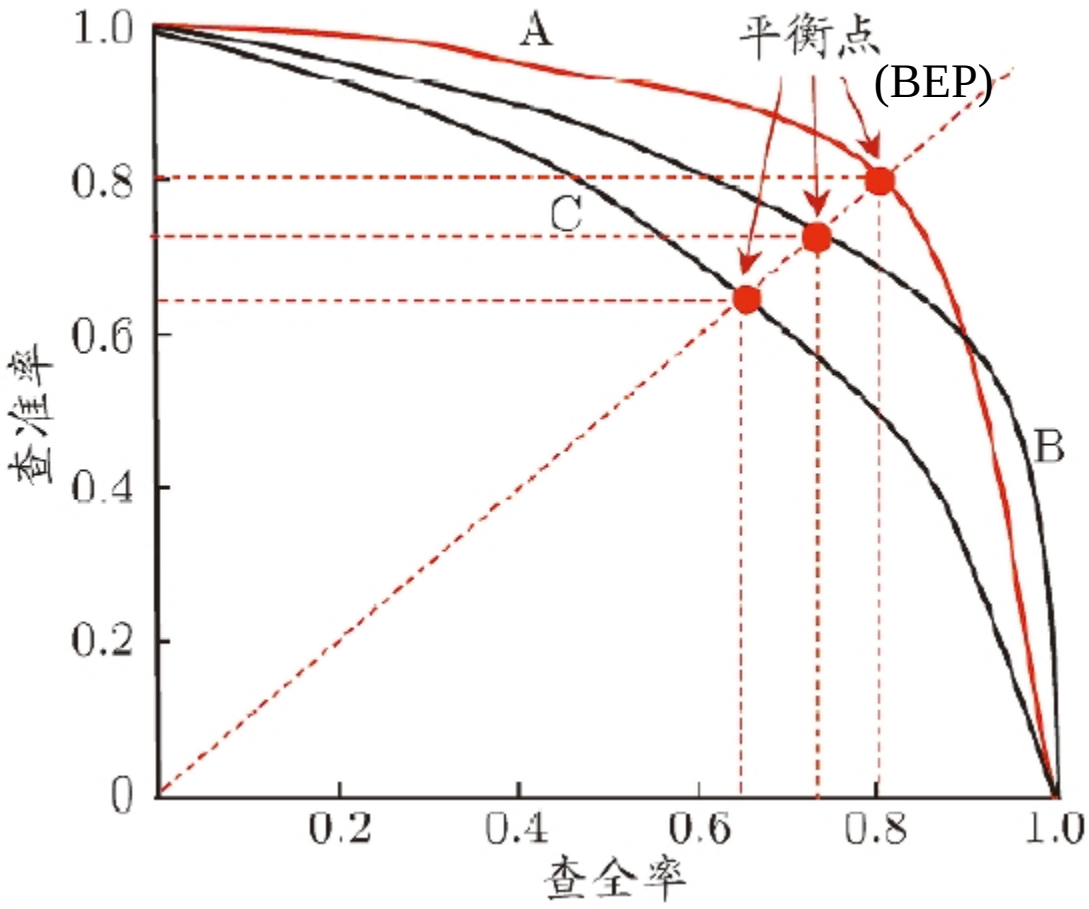
$$P = \frac{TP}{TP + FP}$$



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

性能度量-PR图, BEP

根据学习器的预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测



PR图：

- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C
- 学习器 A ?? 学习器 B

BEP：

- 学习器 A 优于 学习器 B
- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C

性能度量-F-Score

比 BEP 更常用的 F1 度量：

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{\text{样例总数} + TP - TN}$$

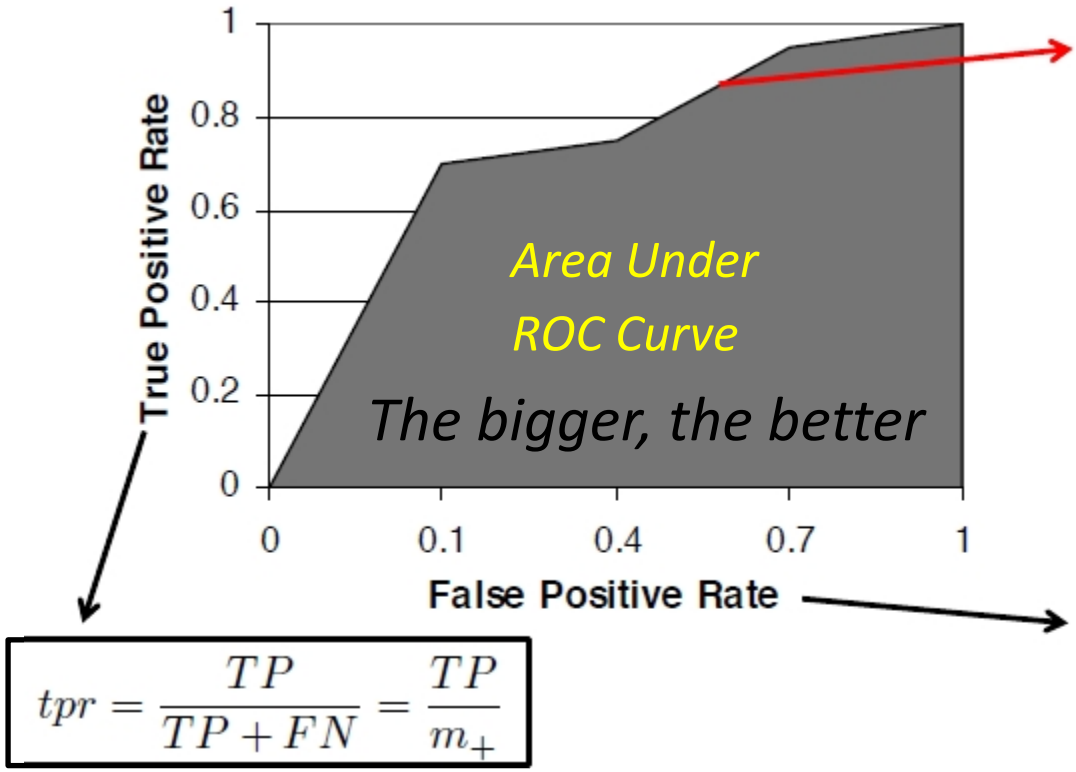
若对查准率/查全率有不同偏好：

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

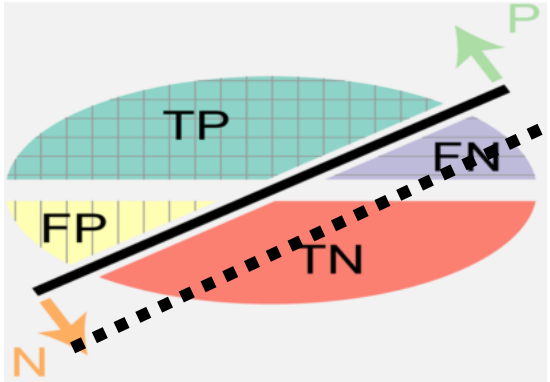
$\beta > 1$ 时查全率有更大影响；$\beta < 1$ 时查准率有更大影响

性能度量-ROC, AUC

AUC: **A**rea **U**nder the ROC **C**urve



ROC (Receiver Operating Characteristic) Curve [Green & Swets, Book 66; Spackman, IWML'89]



$$tpr = \frac{TP}{TP+FN} = \frac{TP}{m_+}$$

$$fpr = \frac{FP}{FP+TN} = \frac{FP}{m_-}$$

- 更多性能指标讨论：http://charleshm.github.io/2016/03/Model-Performance/

03

PART THREE

数学基础
Fundamentals of Mathematics

数学基础

- 线性代数
  - 向量、矩阵、矩阵运算
  - 范数、秩、迹、子空间、奇异值分解
- 微积分基础
  - 泰勒公式、导数、梯度
- 概率与统计基础
  - 概率公式、常见分布、统计量
  - 最大似然估计、随机场、马尔可夫过程
- 优化方法
  - 梯度下降法
  - 拉格朗日乘子法、交替乘子迭代法

➢线性代数

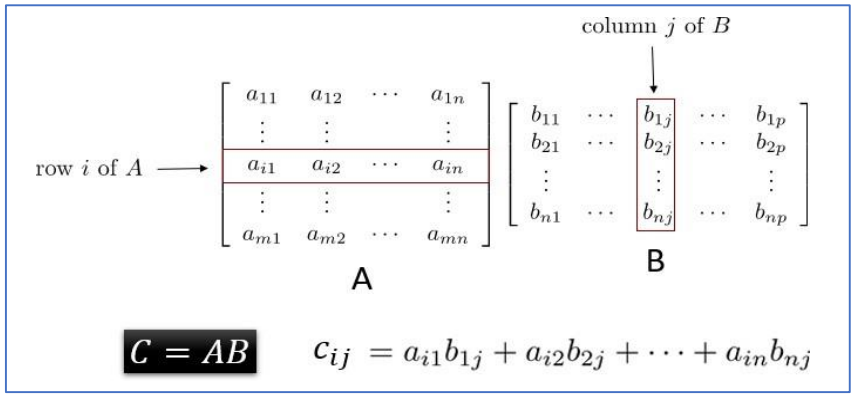
• 基本概念

- 标量（scalar）：一个标量就是一个单独的数
- 向量（vector）：一个向量是一列数
- 矩阵（matrix）：矩阵是一个二维数组，其中的每一个元素被两个索引（而非一个）所确定
- 张量（tensor）：在某些情况下，我们会讨论坐标超过两维的数组

• 矩阵运算

- 转置
- 矩阵加法
- 矩阵乘法
- 逆矩阵运算

$$[\boldsymbol{AB}]_{mn} = \sum_{k=1}^{K} a_{mk} b_{kn}.$$



➢线性代数

- 向量范数

ℓ_1 范数　ℓ_1 范数为向量的各个元素的绝对值之和.

$$\|\boldsymbol{v}\|_1 = \sum_{n=1}^{N} |v_n|.$$

ℓ_2 范数　ℓ_2 范数为向量的各个元素的平方和再开平方.

$$\|\boldsymbol{v}\|_2 = \sqrt{\sum_{n=1}^{N} v_n^2} = \sqrt{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}}.$$

ℓ_∞ 范数　ℓ_∞ 范数为向量的各个元素的最大绝对值，

$$\|\boldsymbol{v}\|_\infty = \max\{v_1, v_2, \cdots, v_N\}.$$

- 矩阵的范数，常用的ℓ_p范数定义

$$\|\boldsymbol{A}\|_p = \left(\sum_{m=1}^{M} \sum_{n=1}^{N} |a_{mn}|^p \right)^{1/p}.$$

➢微积分

- **泰勒公式**
  - 一个函数 $f(x)$ 在已知某一点的各阶导数值的情况之下， 可以用这些导数值做系数构建一个多项式来近似函数在这一点的邻域中的值。

$$f(x) = f(a) + \frac{1}{1!}f'(a)(x-a) + \frac{1}{2!}f^{(2)}(a)(x-a)^2 + \cdots + \frac{1}{n!}f^{(n)}(a)(x-a)^n + R_n(x),$$

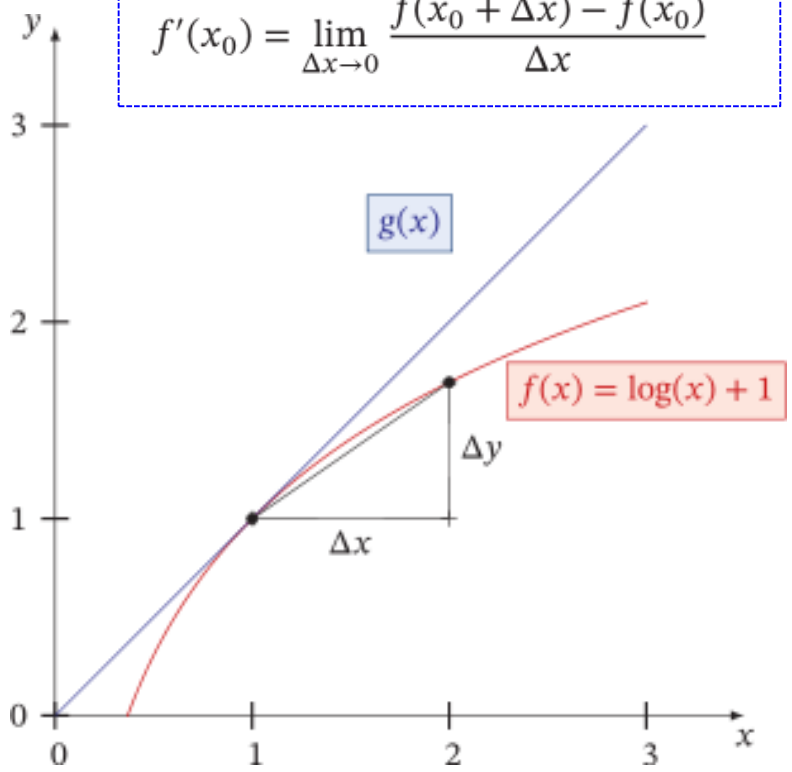
泰勒公式的余项，
$(x-a)^n$的高阶无穷小

- **导数与梯度**
  - 梯度的方向是函数在该点变化最快的方向

➢微积分

- **导数：** 曲线的斜率，反应曲线变化的快慢

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



- **常见函数的导数**

函数	函数形式	导数
常函数	$f(x) = C$，其中 C 为常数	$f'(x) = 0$
幂函数	$f(x) = x^r$，其中 r 是非零实数	$f'(x) = rx^{r-1}$
指数函数	$f(x) = \exp(x)$	$f'(x) = \exp(x)$
对数函数	$f(x) = \log(x)$	$f'(x) = \frac{1}{x}$

- **高阶导数、偏导数**
  - 高阶导数：导数的继续求导
  - 偏导数：关于其中一个变量的导数， 而保持其他变量固定

➢微积分

- 矩阵微分
  - 多元微积分的一种表达方式， 即使用矩阵和向量来表示因变量每个成分关于自变量每个成分的偏导数。

标量关于向量的偏导数

$$\frac{\partial y}{\partial \boldsymbol{x}} = [\frac{\partial y}{\partial x_1}, \cdots, \frac{\partial y}{\partial x_p}]^{\mathrm{T}}$$

向量关于向量的偏导数

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_q}{\partial x_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial y_1}{\partial x_p} & \cdots & \frac{\partial y_q}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{p \times q}$$

➢微积分

• 链式法则

• 在微积分中求复合函数导数的一种常用方法。

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

$$\frac{\partial \boldsymbol{z}}{\partial x} = \frac{\partial \boldsymbol{y}}{\partial x}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \quad \in \mathbb{R}^{1\times N}.$$

（2）若 $\boldsymbol{x} \in \mathbb{R}^M, \boldsymbol{y} = g(\boldsymbol{x}) \in \mathbb{R}^K, \boldsymbol{z} = f(\boldsymbol{y}) \in \mathbb{R}^N$，则

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \quad \in \mathbb{R}^{M\times N}.$$

（3）若 $\boldsymbol{X} \in \mathbb{R}^{M\times N}$ 为矩阵，$\boldsymbol{y} = g(\boldsymbol{X}) \in \mathbb{R}^K, z = f(\boldsymbol{y}) \in \mathbb{R}$，则

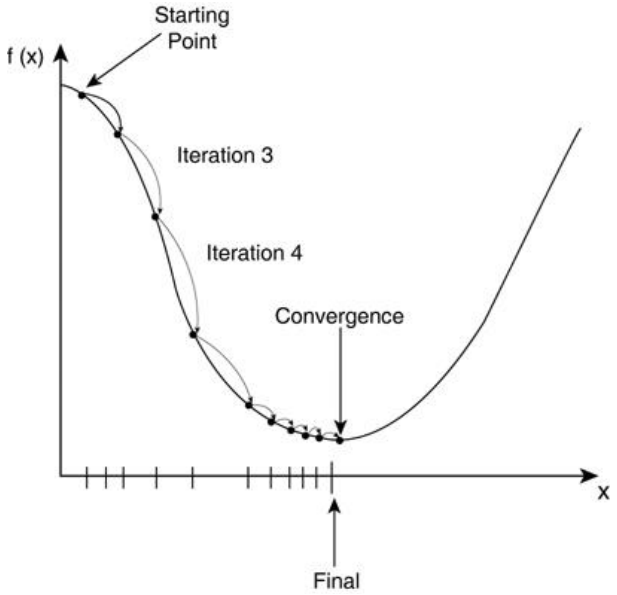
$$\frac{\partial z}{\partial x_{ij}} = \frac{\partial \boldsymbol{y}}{\partial x_{ij}}\frac{\partial z}{\partial \boldsymbol{y}} \quad \in \mathbb{R}.$$

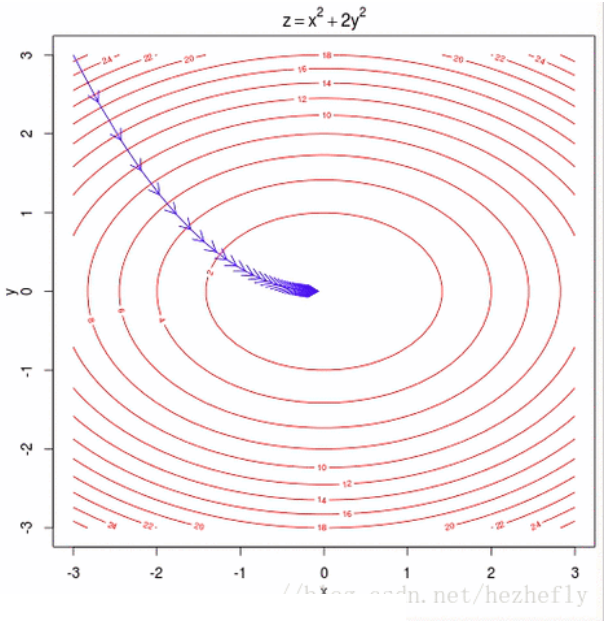
➢概率与统计

- **概率公式**
  - 条件概率 $$p(y|x) \triangleq P(Y=y|X=x) = \frac{p(x,y)}{p(x)}.$$
  - 贝叶斯公式 $$p(y|x) = \frac{p(x|y)p(y)}{p(x)}.$$
- **常见概率分布**
  - 离散随机变量的概率分布有：
    - 伯努利分布、二项分布 $$P\{X=k\} = \binom{n}{k} p^k (1-p)^{n-k}.$$
  - 连续随机变量的概率分布有：
    - 均匀分布、正态分布

➢优化方法

- 梯度下降





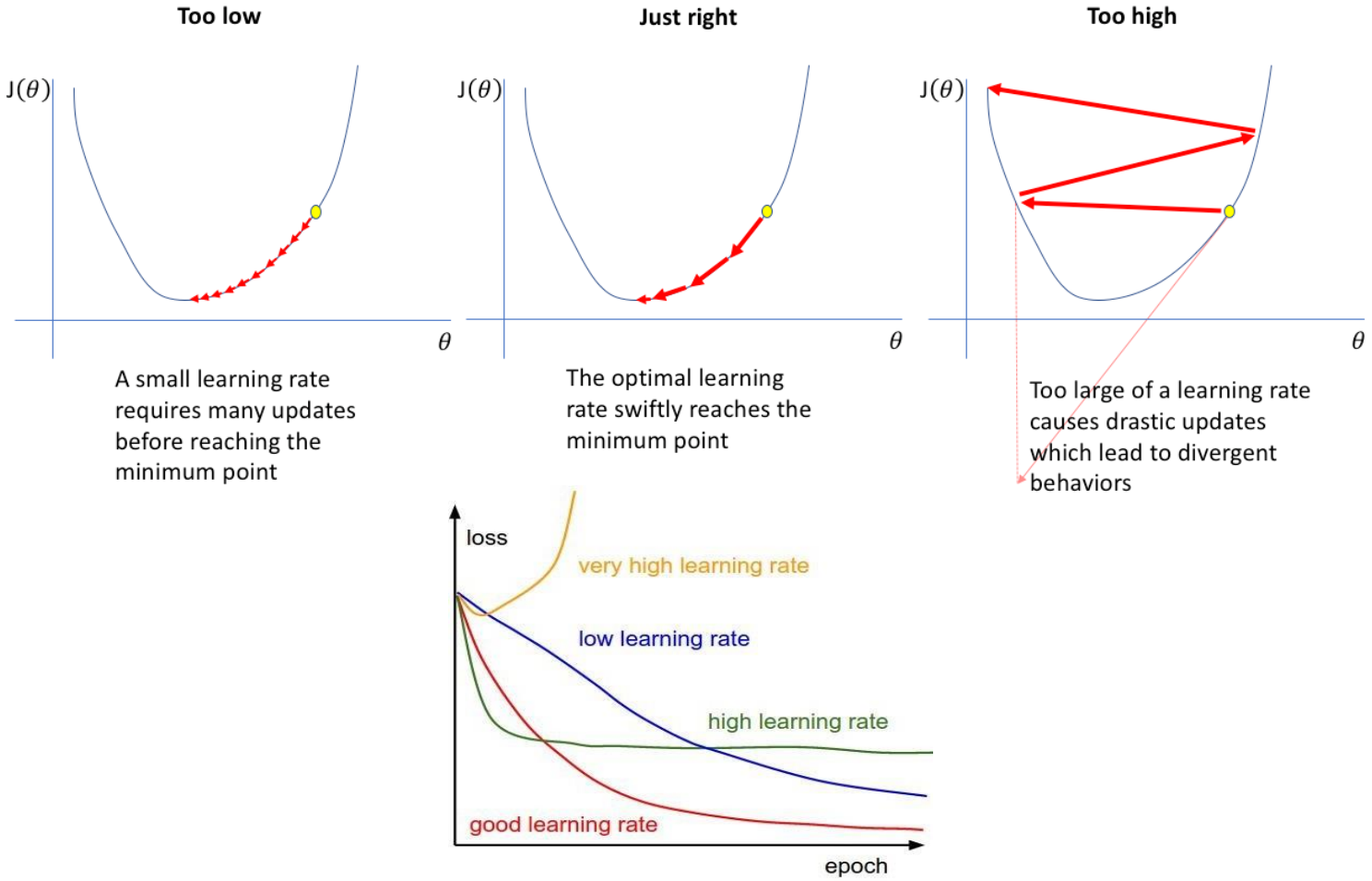
$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t}$$

$$= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}\left(\theta_t; x^{(i)}, y^{(i)}\right)}{\partial \theta}.$$

学习率

➢优化方法

- 学习率是十分重要的超参数！



➢优化方法

- **批量梯度下降法 vs 随机梯度下降法**

（批量）梯度下降法在每次迭代都需计算每个样本上损失函数的梯度并加和，计算复杂度较大；为了降低迭代的计算复杂度，可以每次迭代只采集一个样本，计算该样本的损失函数的梯度并更新参数，即随机梯度下降法。

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t}$$
$$= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}\left(\theta_t; x^{(i)}, y^{(i)}\right)}{\partial \theta}.$$

速度慢！大数据内存不足！

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t}$$
$$= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}\left(\theta_t; x^{(i)}, y^{(i)}\right)}{\partial \theta}.$$

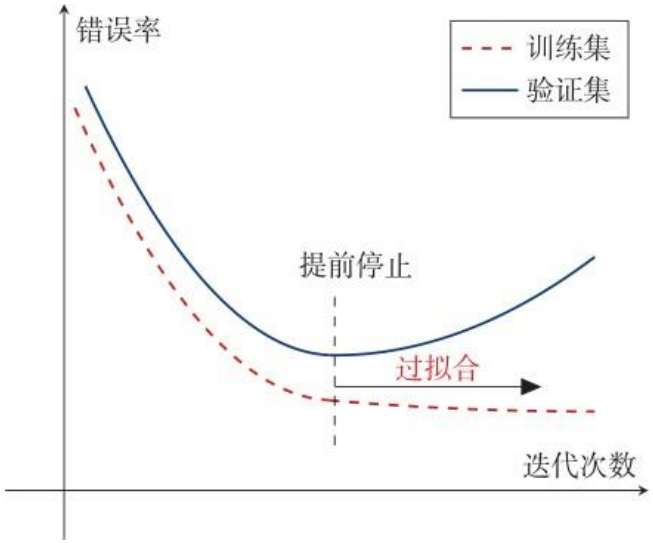
速度慢！大数据内存不足！

➢优化方法

• 小批量梯度下降法

小批量梯度下降法（Mini-Batch Gradient Descent）是批量梯度下降和随机梯度下降的折中。每次迭代时，随机选取一小部分训练样本来计算梯度并更新参数，这样既可以兼顾随机梯度下降法的优点，也可以提高训练效率。

• 提前停止



数学基础

线性代数, 微积分, 概率论, 矩阵论, 优化理论, 信息论...

- 【深度学习数学基础-深度学习大讲堂】 推荐自学！

  https://study.163.com/course/introduction/1005022007.htm

- 【神经网络与深度学习-附录《数学基础》】 重要参考！

  https://nndl.github.io/ => https://nndl.github.io/nndl-book.pdf

- 【Matrix Cookbook】 矩阵必备！

  https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

- 【机器学习的数学基础-SIGAI】 补充参考！

  https://www.bilibili.com/video/BV1Mb411c74N

- 【机器学习-白板推导系列(二)-数学基础-概率】 补充参考！

  https://www.bilibili.com/video/av32905863/

大连理工大学 Dalian University Of Technology

04

PART FOUR

深度学习框架简介

Deep Learning Framework

深度学习框架简介

在这个图像中有不同的分类：**猫，骆驼，鹿，大象**等。

卷积神经网络（CNNs）对于这类图像分类任务十分有效。

从头开始实现一个卷积神经网络模型？我们可能需要几天（甚至几周）才能得到一个有效的模型，时间成本太高！

深度学习框架让一切变得可能！

什么是深度学习框架？

深度学习框架是一种接口、库或工具，利用预先构建和优化好的组件集合定义模型。

一个良好的深度学习框架应具备的关键特性：

- ✓ 性能优化
- ✓ 易于理解和编码
- ✓ 良好的社区支持
- ✓ 并行处理以减少计算
- ✓ 自动计算梯度

深度学习框架核心组件：

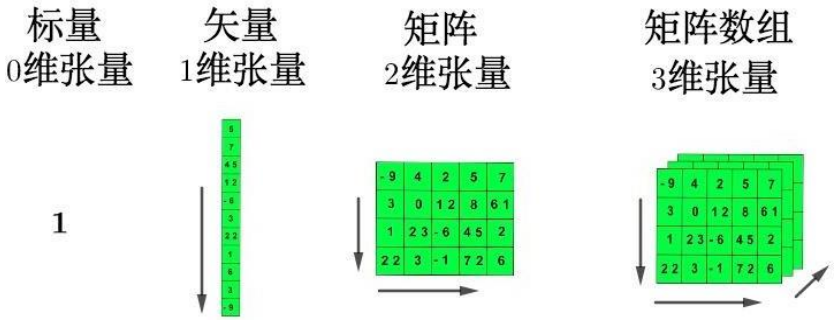
- 张量
- 基于张量的操作
- 计算图一个机器学习任务的核心是模型的定义以及模型的参数求解方式，对这两者进行抽象之后，可以确定一个唯一的计算逻辑，将这个逻辑用图表示，称之为计算图
- 自动微分工具
- **BLAS、cuBLAS、cuDNN等拓展包**

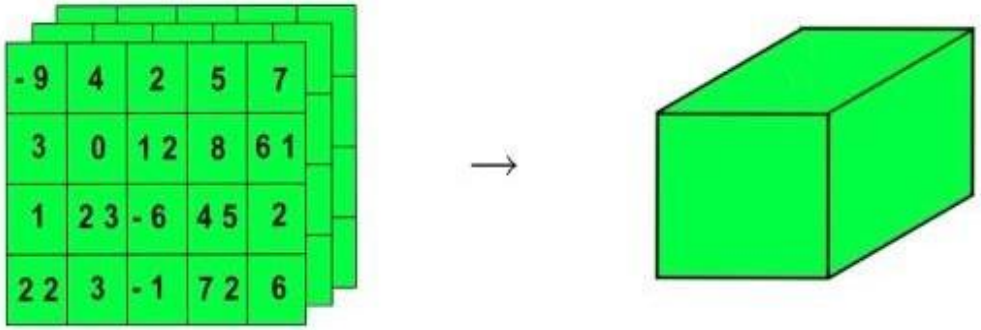
basic linear algebra subroutine

- 张量是深度学习框架中最核心的组件，

  Tensor实际上就是一个多维数组。

- **Tensor对象具有3个属性：**
  - **rank：number of dimensions**
  - **shape: number of rows and columns**
  - **type: data type of tensor's elements**

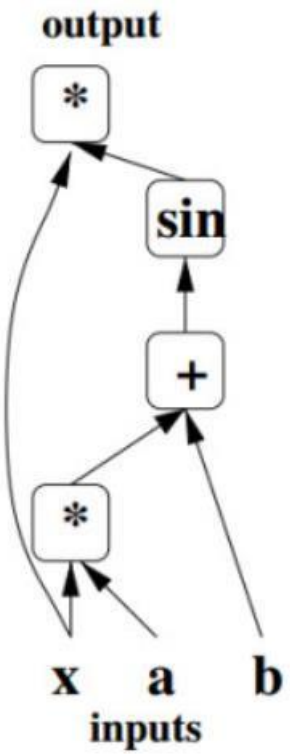




■ 张量的相关操作

- 类型转换：字符串转为数字、转为64（32）位浮点类型（整型）等。
- 数值操作：按指定类型与形状生成张量、正态（均匀）分布随机数、设置随机数种子。
- 形状变换：将数据的shape按照指定形状变化、插入维度、将指定维度去掉等。
- 数据操作：切片操作、连接操作等。
- 算术操作：求和、减法、取模、三角函数等。
- 矩阵相关的操作：返回给定对角值的对焦tensor、对输入进行反转、矩阵相乘、求行列式…

■ 计算图



■ 计算图结构

- 用不同的占位符（*，+，sin）构成操作结点，以字母x、a、b构成变量结点。
- 用有向线段将这些结点连接起来，组成一个表征运算逻辑关系的清晰明了的“图”型数据结构。

■ 自动微分

- 计算图带来的一个好处是让模型训练阶段的梯度计算变得模块化且更为便捷，也就是自动微分法。

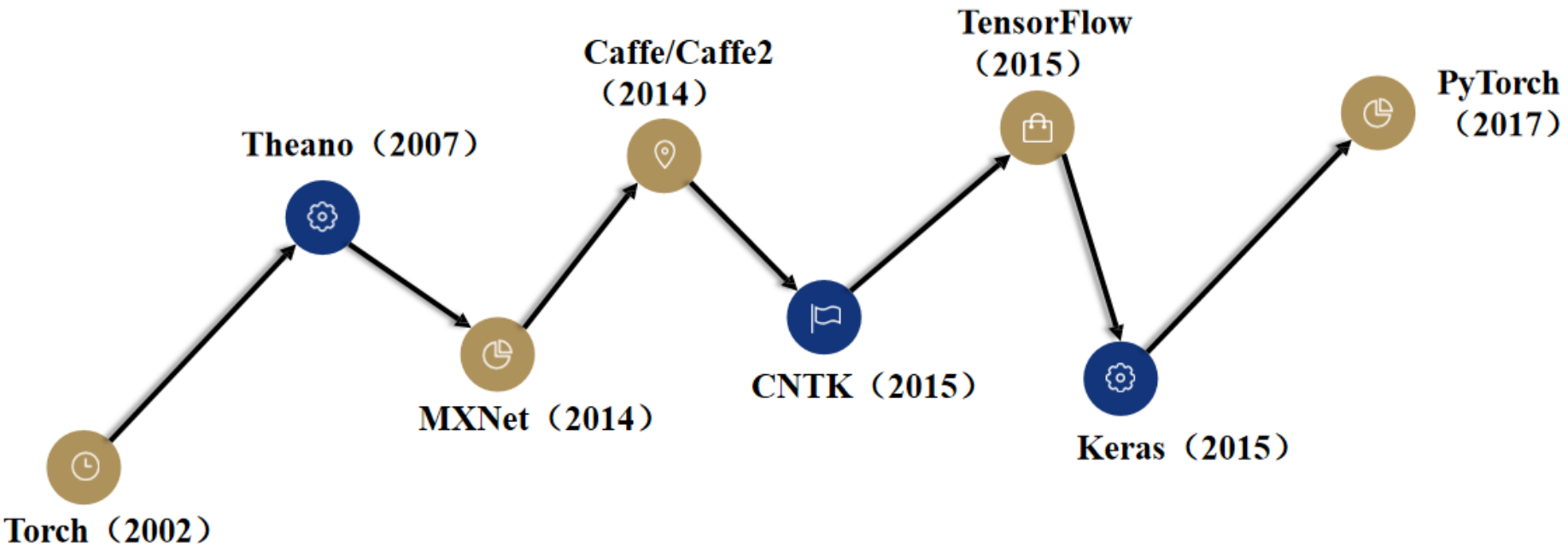
[1] Bengio Y.Learning deep architectures for AI[J]. Foundations and trends® in Machine Learning, 2009, 2(1): 1-127.

■ BLAS、cuBLAS、cuDNN等拓展包

- 通过前面所介绍的组件，已经可以搭建一个全功能的深度学习框架：
  将待处理数据转换为张量，针对张量施加各种需要的操作，通过自动微分对模型展开训练，然后得到输出结果开始测试。
- 存在的缺陷是运行缓慢

- 利用扩展包来进行加速，例如：
  - ✓ Fortran实现的BLAS（基础线性代数子程序）
  - ✓ 英特尔的MKL（Math Kernel Library）
  - ✓ NVIDIA推出的针对GPU优化的cuBLAS和cuDNN等更据针对性的库

■ 发展历程


Torch（2002）
Theano（2007）
MXNet（2014）
Caffe/Caffe2
（2014）
CNTK（2015）
TensorFlow
（2015）
Keras（2015）
PyTorch
（2017）

深度学习框架简介

下图是各个主流开源框架在Github上的数据统计（2020年5月）。

框架	机构	支持语言	Stars	Forks	Contributors
Torch	Facebook	Lua	8.5K	2.4K	130
Theano	U.Montreal	Python	9.1K	2.5K	332
MXNet	DMLC	Python/C++/…	18.7K	6.6K	794
Caffe	BVLC	C++/Python	30.3K	18.3K	265
CNTK	Microsoft	C++	16.8K	4.4K	199
TensorFlow	Google	Python/C++/…	145K	81.3K	2496
Keras	fchollet	Python	48.3K	18.3K	817
PyTorch	Facebook	Python/C++/…	38.9K	10K	1406

深度学习框架简介

■ 深度学习的主流框架

- ✓ **Caffe**
- ✓ **Theano**
- ✓ **Torch & PyTorch**
- ✓ **TensorFlow**
- ✓ **Keras**
- ✓ **MXNet**


Caffe

- Caffe的全称是Convolutional Architecture for Fast Feature Embedding，它是一个清晰、高效的深度学习框架，于2013年底由加州大学伯克利分校开发，核心语言是C++。它支持命令行、Python和 MATLAB接口。Caffe的一个重要特色是可以在不编写代码的情况下训练和部署模型。

  https://github.com/BVLC/caffe

- Caffe2是由Facebook组织开发的一个轻量级的深度学习框架，具有模块化和可扩展性等特点。它 在原来的Caffe的基础上进行改进，提高了它的表达性，速度和模块化，现在被并入Pytorch项目。 Caffe曾经名噪一时，但由于使用不灵活、代码冗长、安装困难、不适用构建循环网络等问题， 已经很少被使用。

  https://github.com/caffe2

Caffe

Caffe

Caffe2

优点

- ✓ 核心程序用C++编写，因此更高效，适合工业界开发
- ✓ 网络结构都是以配置文件形式定义，不需要用代码设计网络
- ✓ 拥有大量的训练好的经典模型（AlexNet、VGG、Inception)在其 Model Zoo里

缺点

- ✓ 缺少灵活性和扩展性
- ✓ 依赖众多环境，难以配置
- ✓ 缺乏对递归网络RNN和语言建模的支持，不适用于文本、声音或时间序列数据等其他类型的深度学习应用

Theano

- Theano是深度学习框架的鼻祖，由Yoshua Bengio和蒙特利尔大学的研究小组于2007年创建，是率先广泛使用的深度学习框架。Theano 是一个 Python 库，速度更快，功能强大，可以高效的进行数值表达和计算，可以说是从NumPy矩阵表达向tensor表达的一次跨越，为后来的深度学习框架提供了样板。遗憾的是Theano团队2017年已经停止了该项目的更新，深度学习应用框架的发展进入到了背靠工业界大规模应用的阶段。

- **缺点**
  - ✓ 原始的 Theano 只有比较低级的 API；大型模型可能需要很长的编译时间；不支持多 GPU；有的错误信息的提示没有帮助。

https://github.com/Theano

Torch & PyTorch

- Torch是一个有大量机器学习算法支持的科学计算框架，其诞生已经有十余年，但真正起势得益于Facebook开源了大量Torch的深度学习模块和扩展。Torch的一个特殊之处是采用了Lua编程语言（曾被用来开发视频开发）。

  https://github.com/torch

- PyTorch于2016年10月发布，是一款专注于直接处理数组表达式的低级API。 前身是Torch。 Facebook人工智能研究院对PyTorch提供了强力支持。PyTorch支持动态计算图，为更具数学倾向的用户提供了更低层次的方法和更多的灵活性，目前许多新发表的论文都采用PyTorch作为论文实现的工具，成为学术研究的首选解决方案。

  https://github.com/pytorch

PyTorch

优点

- ✓ 简洁易用
- ✓ 可以为使用者提供更多关于深度学习实现的细节，如反向传播和其他训练过程
- ✓ 活跃的社区：提供完整的文档和指南
- ✓ 代码很**Pythonic**（简洁、优雅）

缺点

- ✓ 无可视化接口和工具
- ✓ 导出模型不可移植，工业部署不成熟
- ✓ 代码冗余量较大

TensorFlow

- TensorFlow最初由Google Brain团队针对机器学习和深度神经网络进行研究所开发的，目前开源之后可以在几乎各种领域适用。它灵活的架构可以部署在一个或多个CPU、GPU的台式及服务器中， 或者使用单一的API应用在移动设备中。TensorFlow可以说是当今十分流行的深度学习框架，特别在工业界。TensorFlow提供了全面的服务，构建了活跃的社区，完善的文档体系，大大降低了我们的学习成本，另外，TensorFlow有很直观的计算图可视化呈现。模型能够快速的部署在各种硬件机器上，从高性能的计算机到移动设备，再到更小的更轻量的智能终端。
- 但是，TensorFlow相比Pytorch，Caffe等框架，计算速度很慢。而且通过它构建一个深度学习框架需要更复杂的代码，还要忍受重复的多次构建静态图。

  https://github.com/tensorflow

TensorFlow

■ 优点

- ✓ 自带tensorboard可视化工具，能够让用户实时监控观察训练过程
- ✓ 拥有大量的开发者，有详细的说明文档、可查询资料多
- ✓ 支持多GPU、分布式训练，跨平台运行能力强
- ✓ 具备不局限于深度学习的多种用途，还有支持强化学习和其他算法的工具

■ 缺点

- ✓ 频繁变动的接口
- ✓ 接口设计过于晦涩难懂

Keras

- Keras于2015年3月首次发布，拥有“为人类而不是机器设计的API”，由Google的Francis Chollet创建与维护，它是一个用于快速构建深度学习原型的高层神经网络库，由纯Python编写而成，以TensorFlow，CNTK，Theano和MXNet为底层引擎，提供简单易用的API接口，能够极大地减少一般应用下用户的工作量。能够和TensorFlow，CNTK或Theano配合使用。通过Keras的API可以仅使用数行代码就构建一个网络模型，Keras+Theano，Keras+CNTK的模式曾经深得开发者喜爱。目前Keras整套架构已经封装进了TensorFlow，在TF.keras可以完成Keras的所有事情。

  https://keras.io/

Keras

优点

- ✓ 更简洁，更简单的API
- ✓ 丰富的教程和可重复使用的代码
- ✓ 更多的部署选项（直接并且通过TensorFlow后端），更简单的模型导出
- ✓ 支持多GPU训练

缺点

- ✓ 过度封装导致丧失灵活性，导致用户在新增操作或是获取底层的数据信息时过于困难
- ✓ 许多BUG都隐藏于封装之中，无法调试细节
- ✓ 初学者容易依赖于 Keras 的易使用性而忽略底层原理

MXNet

- MXNet于2014年由上海交大校友陈天奇与李沐组建团队开发，2017年1月，MXNet项目进入Apache基金会，成为Apache的孵化器项目。MXNet主要用C++编写，强调提高内存使用的效率，甚至能在智能手机上运行诸如图像识别等任务。
- 它拥有类似于Theano 和 TensorFlow 的数据流图，为多GPU配置提供了良好的配置，还有着类似于Blocks等更高级别的模型构建块，并且可以在任何硬件上运行（包括手机）。 同时MXNet 是一个旨在提高效率和灵活性的深度学习框架，提供了强大的工具来帮助开发人员利用GPU和云计算的全部功能。

https://mxnet.apache.org/

大连理工大学 Dalian University Of Technology

MXNet

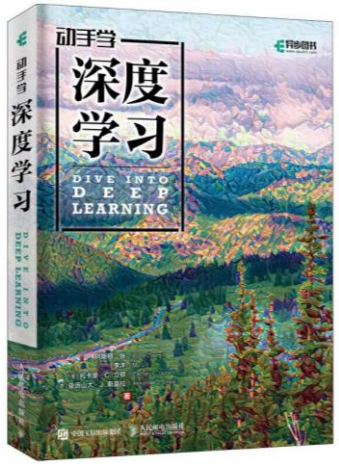
■ 优点

- ✓ 灵活的编程模型
- ✓ 从云端到客户端可移植
- ✓ 多语言支持
- ✓ 本地分布式训练
- ✓ 性能优化

■ 缺点

- ✓ 社区相对PyTorch和TensorFlow来说相对小众

- https://github.com/d2l-ai/d2l-zh
- https://github.com/dsgiitr/d2l-pytorch



深度学习框架简介

vs

Google **Facebook**

百度 旷视 清华 华为

深度学习框架简介

TensorFlow™ Google

vs

PyTorch Facebook

飞桨 百度

天元 MegEngine 旷视

Jittor 清华

MindSpore 华为

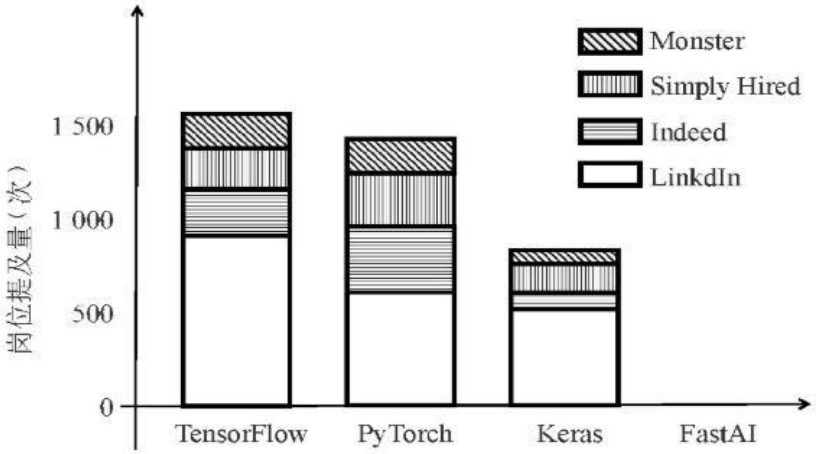
Tensorflow vs PyTorch

目前来看TensorFlow和PyTorch框架是业界使用最为广泛的两个深度学习框架，TensorFlow在工业界拥有完备的解决方案和用户基础，PyTorch得益于其精简灵活的接口设计，可以快速设计和调试网络模型，在学术界获得好评如潮。

■ TensorFlow和PyTorch在学术界、工业界的使用现状。

学术界（左图）：基于五大级会论文的使用率； 工业界（右图）：通过线上招聘启事中的提及率。

Conference	PT 2018	PT 2019	PT Growth	TF 2018	TF 2019	TF Growth
CVPR	82	280	241. 5%	116	125	7. 8%
NAACL	12	66	450. 0%	34	21	−38. 2%
ACL	26	103	296. 2%	34	33	−2. 9%
ICLR	24	70	191. 7%	54	53	−1. 9%
ICML	23	69	200. 0%	40	53	32. 5%



[1] HE H. The State of Machine Learning Frameworks in 2019.(2019-10-10). https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/.
[2] HALE J. Which Deep Learning Framework is Growing Fastest.(2019-04-01).https://towardsdatascience.com/which-deeplearning- framework-is-growing-fastest-3f77f14aa318

■ 对比分析

- **运行机制：** 均是基于张量和有向非循环图（DAG）

  - TensorFlow：运行代码时，DAG是以静态方式定义的，若需要实现动态DAG，则需要借助TensorFlow Fold库；TensorFlow需要借助特殊的调试工具tfdbg才能进行调试。
  - PyTorch：动态DAG是内置的，可以随时定义、随时更改、随时执行节点，相当灵活；使用者可以使用任何一个喜欢的调试工具，比如PDB、IPDB、PyCharm调试器或者原始的print语句。

- **训练模式：**

  - TensorFlow：使用者必须手动编写代码，并微调要在特定设备上运行的每个操作，以实现分布式训练。
  - PyTorch：需要利用异步执行的本地支持来实现的，其自身在分布式训练比较欠缺。

- **可视化情况：**

  - TensorFlow：内置TensorBoard库强大，可显示模型图，绘制标量变量，实现图像、嵌入可视化、播放音频等功能。
  - PyTorch：可以使用Visdom，但是Visdom提供的功能很简单且有限，可视化效果远远比不上TensorBoard。

Tensorflow vs PyTorch

■ 对比分析

- **细化特征比较：**

序号	参量	TensorFlow	PyTorch
1	安装难易程度	良好	优秀
2	上手难易程度	良好	优秀
3	代码理解程度	良好	优秀
4	API丰富程度	优秀	良好
5	模型丰富程度	优秀	良好
6	社区支持程度	优秀	良好
7	语言支持程度	优秀	良好
8	可视化程度	优秀	良好

- **使用场景：**

➢ TensorFlow：当需要拥有丰富的入门资源、开发大型生产模型、可视化要求较高、大规模分布式模型训练时。

➢ PyTorch：如果想要快速上手、对于功能性需求不苛刻、追求良好的开发和调试体验、擅长Python化的工具时。

[1] 黄玉萍, 梁炜萱, 肖祖环. 基于TensorFlow和PyTorch的深度学习框架对比分析. 现代信息科技, 2020, 4(04): 80-82+87.

- 《PyTorch深度学习实践》，Hongpu Liu
- https://www.bilibili.com/video/BV1Y7411d7Ys
- https://pytorch.org/resources/
- https://github.com/apachecn/pytorch-doc-zh

- 《Python编程与实践》公开课，北京邮电大学，陈光
- https://github.com/fly51fly/Practical_Python_Programming
- https://www.bilibili.com/video/BV1b7411N7P2

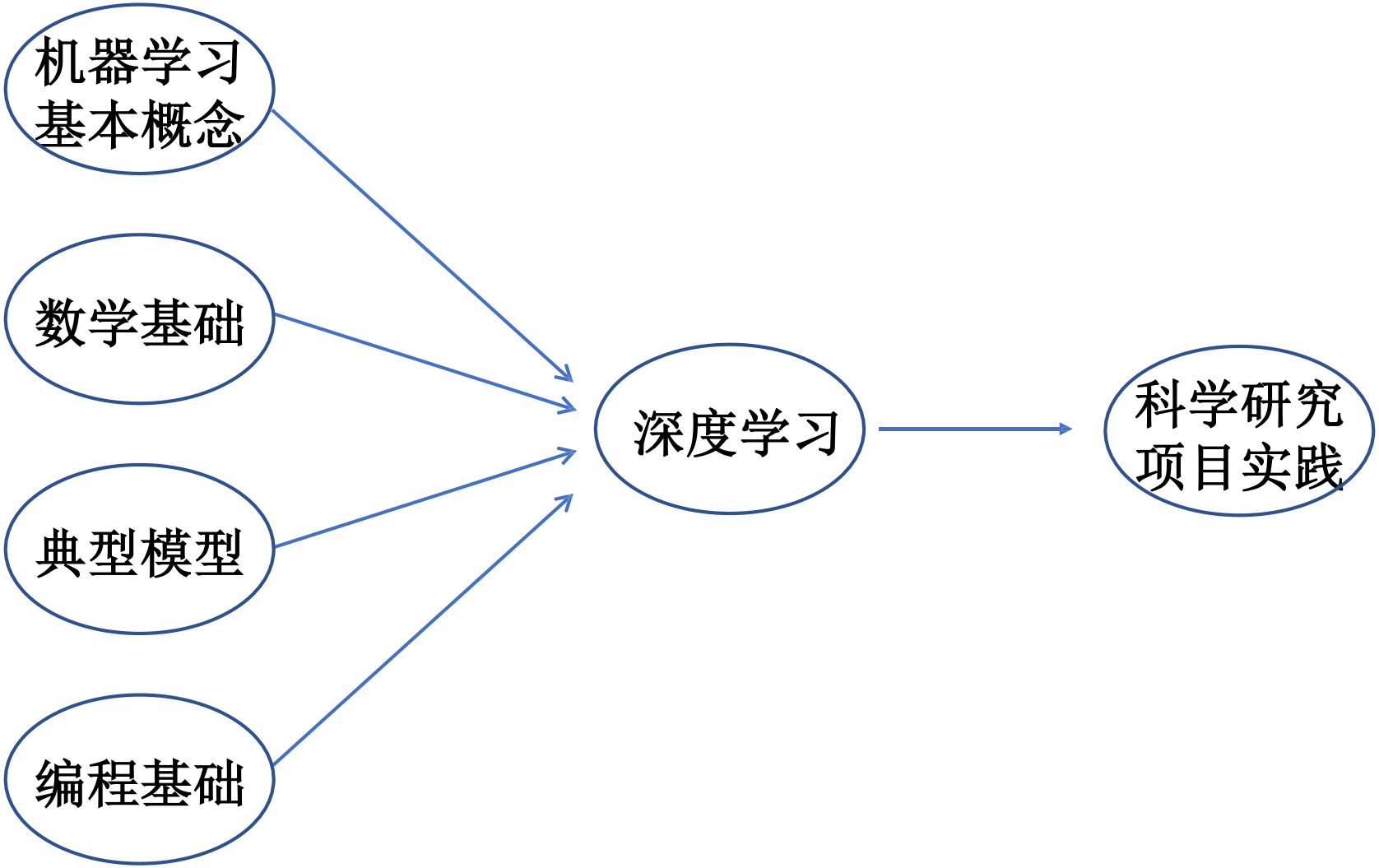
05

PART FIVE

小结和作业

Summary&Homework

小结和作业



小结和作业

- 《Python编程与实践》公开课，北京邮电大学，陈光
- https://github.com/fly51fly/Practical_Python_Programming
- https://www.bilibili.com/video/BV1b7411N7P2

- 《PyTorch深度学习实践》，Hongpu Liu
- https://www.bilibili.com/video/BV1Y7411d7Ys
- https://pytorch.org/resources/
- https://github.com/apachecn/pytorch-doc-zh

【深度学习数学基础-深度学习大讲堂】

https://study.163.com/course/introduction/1005022007.htm