

# 深度学习

## 第2讲：深度学习基础

王亮

智能感知与计算研究中心 (CRIPAC)  
模式识别国家重点实验室 (NLPR)  
中科院自动化研究所 (CASIA)

# 目录

---

**1/ 课程回顾**

**2/ 线性代数**

**3/ 概率和信息论**

**4/ 深度学习中的数值计算**

**5/ 机器学习基本知识**

# 回顾: 课程目标

---

- 在现实世界的不同任务和数据集合的背景下，初步了解深度学习这样一种代表性的多变量数据分析方法
- 理解深度学习算法的原理
- 如何将现实任务抽象为学习的问题
- 怎样利用现有工具来实现设计的模型
- 了解对于不同类型的问题，哪类具体学习方法更加合适

# 回顾: 课程要求

---

- 课堂讲授(3小时/周) + 课程设计作业 + 文献阅读
- 最终成绩 = 50% 期末考试 + 25% 课程设计作业 + 25% 文献阅读
- 课程设计作业需要分组进行, 每组原则上要求5-10人, 最终需要提交按照规范格式完成的技术报告
- 课程设计包含presentation环节, 鼓励大家主动申请进行报告(有加分), 如最终申请人数较少, 将采取随机抽签的方式进行

# 回顾: 深度学习

## ARTIFICIAL INTELLIGENCE

Any technique that enables computers to mimic human behavior



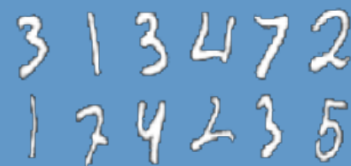
## MACHINE LEARNING

Ability to learn without explicitly being programmed



## DEEP LEARNING

Learn underlying features in data using neural networks



# 回顾: 深度神经网络 (DNN)

- 起源:

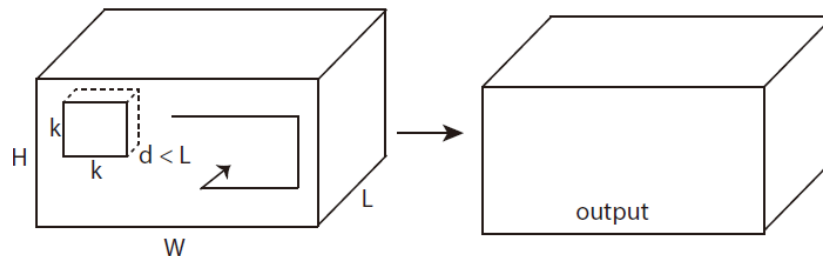
- 1962 – simple/complex cell, Hubel and Wiesel
- 1970 – efficient error backpropagation, Linnainmaa
- 1979 – deep neocognitron, convolution, Fukushima
- 1987 – autoencoder, Ballard
- 1989 – convolutional neural networks (CNN), Lecun
- 1991 – deep recurrent neural networks (RNN), Schmidhuber
- 1997 – long short-term memory (LSTM), Schmidhuber

- 缺陷:

参数量巨大 → 高计算复杂度

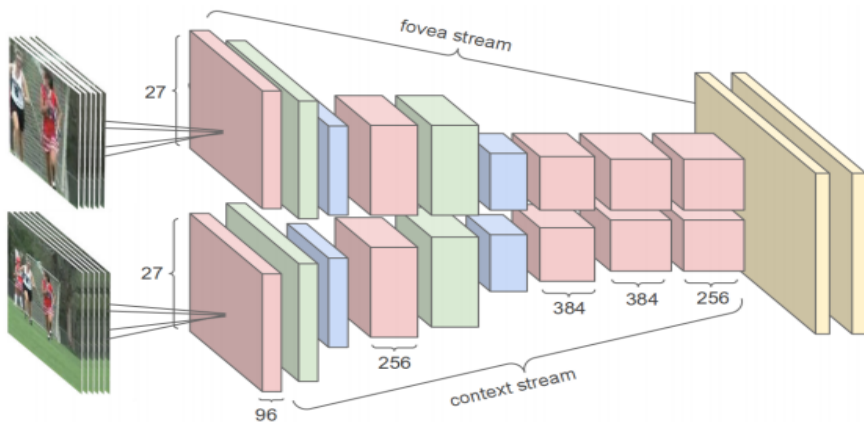
数据集规模有限 → 过拟合问题

CNN: convolutional neural networks; RNN: recurrent neural networks

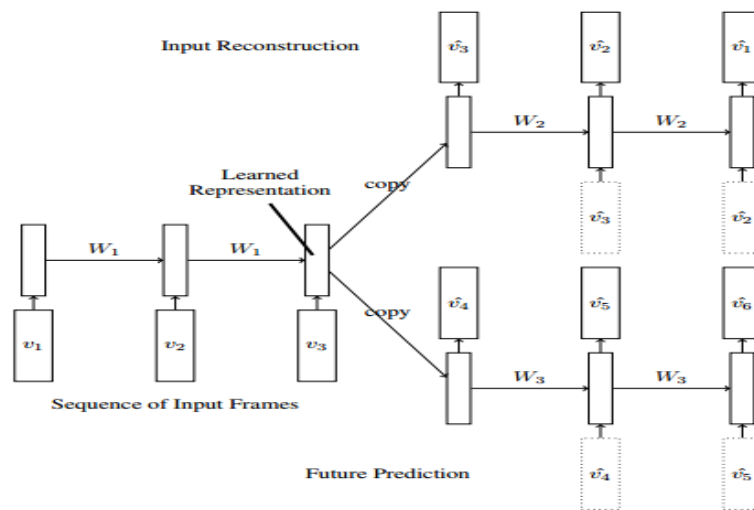


## 3D Convolution, ICCV2015

## Long-range Videos Modelling, CVPR2015

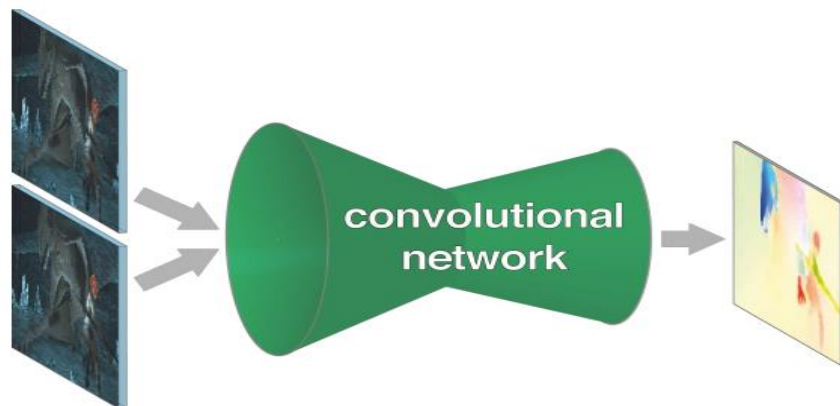


## Multi-resolution CNN, CVPR2014

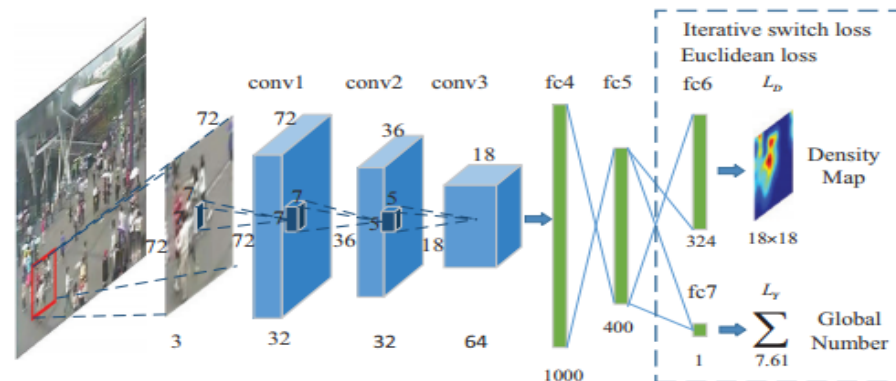


## Autoencoder-RNN, ICML2015

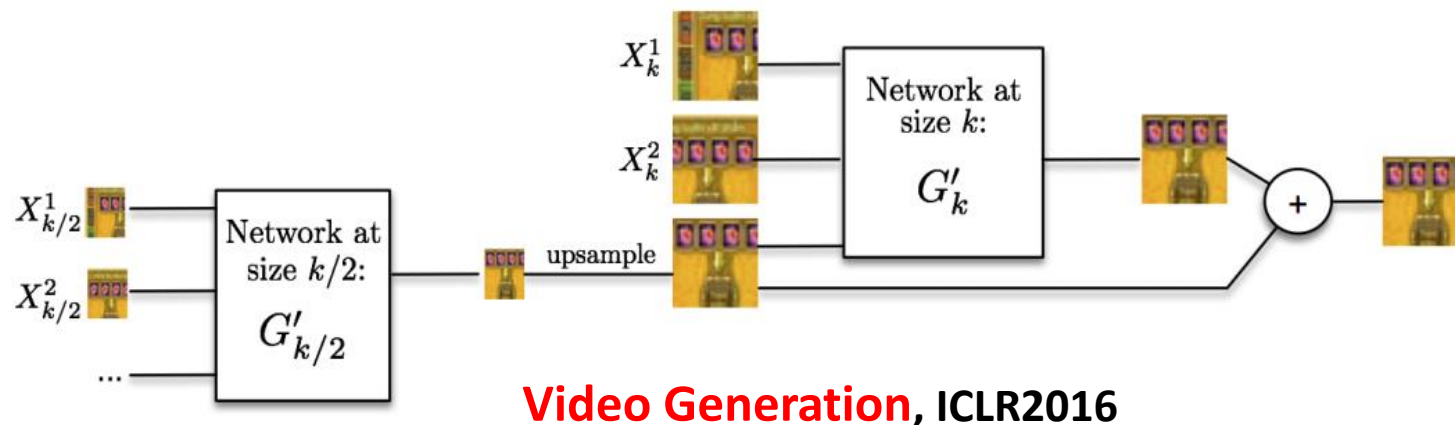
# 回顾: 其他...



Optical Flow Prediction, ICCV2015



Cross-scene Crowd Counting, CVPR2015



Video Generation, ICLR2016

在许多应用中均取得了最先进的性能



# 回顾: 未来研究方向

---

## ■ 大规模深度学习

- 多GPU学习
- 分布式系统

## ■ 无监督/半监督视频特征学习

- 实际应用中缺乏监督信息

## ■ 多模态学习

- 视频与其他数据模态密切相关, 例如文本、音频

## ■ 脑启发的深度学习模型

- 传统的神经网络受到人类认知机制的启发
- 未来可参考神经科学和脑科学领域的先进成果

# 目录

---

1/ 课程回顾

2/ 线性代数

3/ 概率和信息论

4/ 深度学习中的数值计算

5/ 机器学习基本知识

# 关于本节内容

---

- 不涉及所有线性代数知识
- 聚焦于与深度学习最相关的知识
- 更多的线性代数知识，可以参考Georgi Shilov所著《Linear Algebra》

# 标量 (Scalars)

---

- 用来定义向量空间域的一个元素，一个标量对应一个单一的数值
- 包括整数、有理数以及实数等
- 我们用斜体来表示:

*a, n, x*

# 向量 (Vectors)

- 向量是一个一维数组:

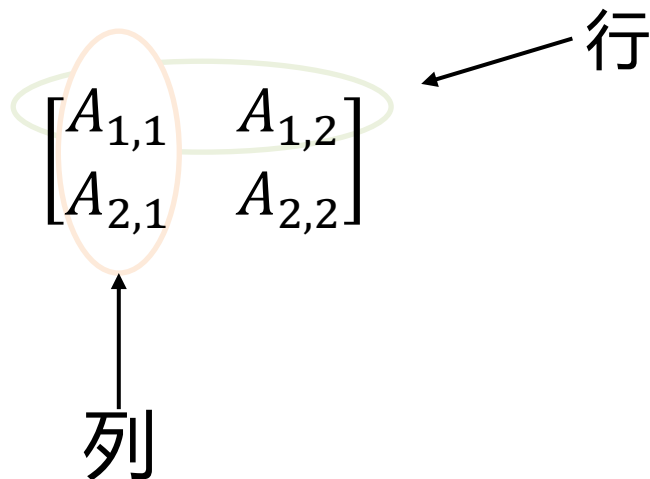
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- 向量中的元素的类型可以为二进制数、整数以及实数等
- 包括类型和大小的示例符号:

$$\mathbb{R}^n$$

# 矩阵 (Matrices)

- 矩阵是一个二维数组:



- 类型和形状的示例符号:

$$A \in \mathbb{R}^{m \times n}$$

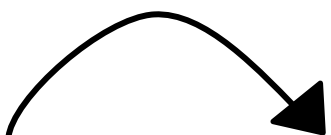
# 张量 (Tensors)

---

- 张量是一个维度为非负整数的数组
  - 零维, 表示一个数值
  - 一维, 表示一个向量
  - 两维, 表示一个矩阵
  - 更高维, 表示一个三维或三维以上的张量

# 矩阵转置 (Matrix Transpose)

$$(A^T)_{i,j} = A_{j,i}$$


$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

矩阵的转置可以视为以主对角线为轴的矩阵的镜像

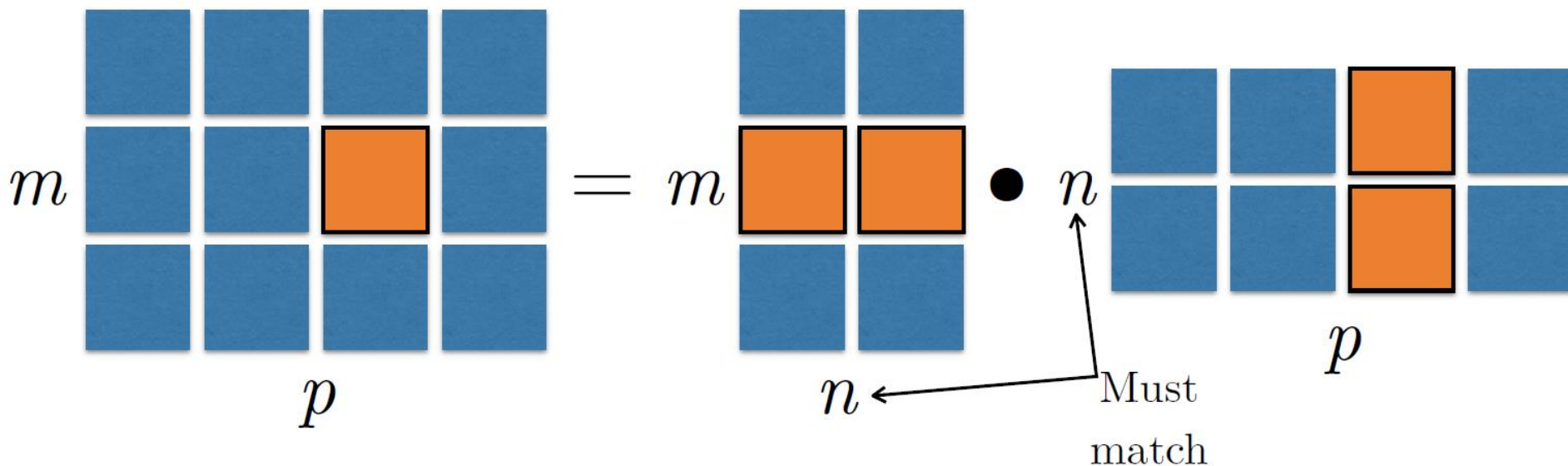
$$(A^T)^T = A \quad (AB)^T = B^T A^T \quad (A + B)^T = A^T + B^T$$

$$(cA)^T = cA^T \quad \det(A^T) = \det(A) \quad a \cdot b = a^T b$$



# 矩阵乘积 (Matrix Product)

- $C = AB$
- $C_{i,j} = \sum_k A_{i,k} B_{k,j}$



# 单位矩阵 (Identity Matrix)

---

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

单位阵示例:  $I_3$

$$\forall x \in \mathbb{R}^n, I_n x = x$$

# 方程组 (Systems of Equations)

$$Ax = b \quad \Rightarrow \quad \begin{cases} (A_1, :)x = b_1 \\ (A_2, :)x = b_2 \\ \dots \\ (A_m, :)x = b_m \end{cases}$$

其中,

$$A = \begin{bmatrix} A_1, : \\ A_2, : \\ \dots \\ A_m, : \end{bmatrix}$$

# 解方程组

---

- 一个线性方程组解的情况包括:
  - 无解
  - 包含多个解
  - 仅有一个解，这意味着线性方程组中的矩阵是一个可逆矩阵

# 矩阵求逆 (Matrix Inversion)

---

- 逆矩阵:

$$A^{-1}A = I_n$$

- 用逆矩阵求解方程组:

$$Ax = b$$

$$A^{-1}Ax = A^{-1}b$$

$$I_n x = A^{-1}b$$

- 数值上不稳定, 但对抽象分析很有用

# 可逆性 (Invertibility)

---

- 矩阵不可逆的情况包括：
  - 行数比列数多
  - 列数比行数多
  - 冗余的行数或列数，包括线性依赖或低秩等情况

# 范数 (Norms)

---

- 度量一个向量多 “大” 的函数
- 类似于向量表示的点到零点的距离
  - $f(x) = 0 \Rightarrow x = 0$
  - $f(x + y) \leq f(x) + f(y)$  (三角不等式)
  - $\forall \alpha \in \mathbb{R}, f(\alpha x) = |\alpha|f(x)$

# 范数 (Norms)

---

- $L^p$  范数

$$\|x\|_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- L2 范数,  $p=2$ , 最常用范数:
- L1 范数,  $p=1$ :  $\|x\|_1 = \sum_i |x_i|$ .
- 最大范数,  $p=\infty$ :  $\|x\|_\infty = \max_i |x_i|$ .



# 特殊矩阵和向量

---

- 单位向量:

$$\|x\|_2 = 1$$

- 对称矩阵:

$$A = A^T$$

- 正交矩阵:

$$A^T A = A A^T = I$$
$$A^{-1} = A^T$$

# 特征分解 (Eigendecomposition)

---

- 特征向量和特征值:

$$Av = \lambda v$$

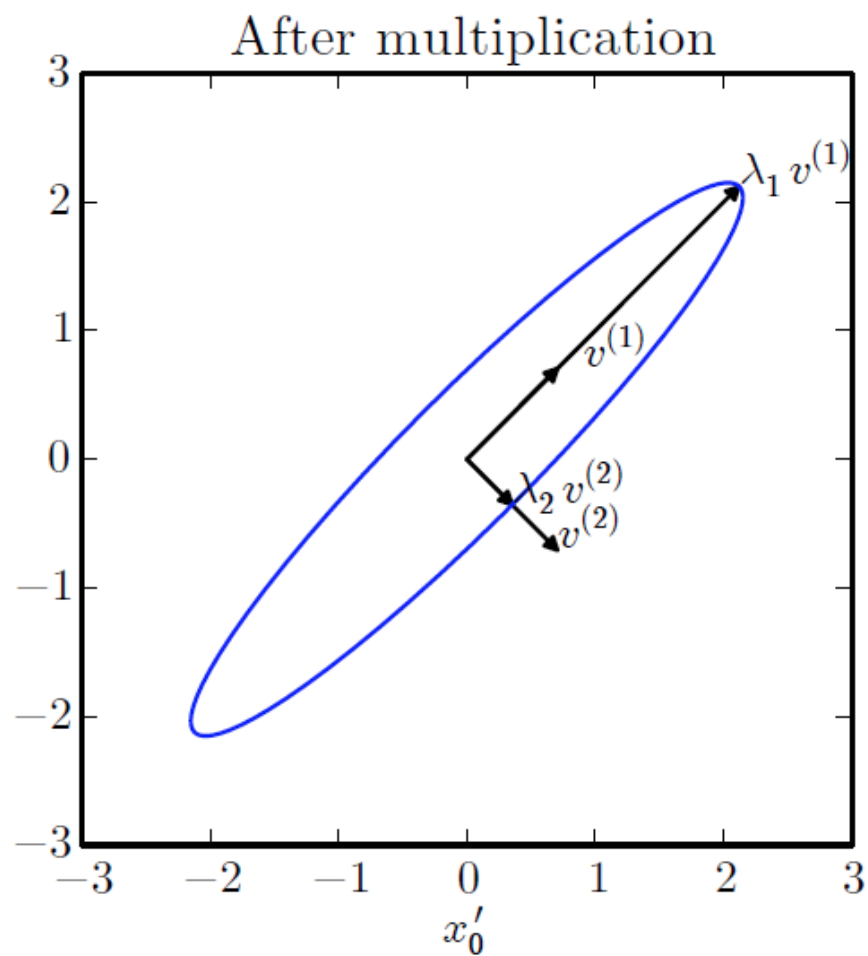
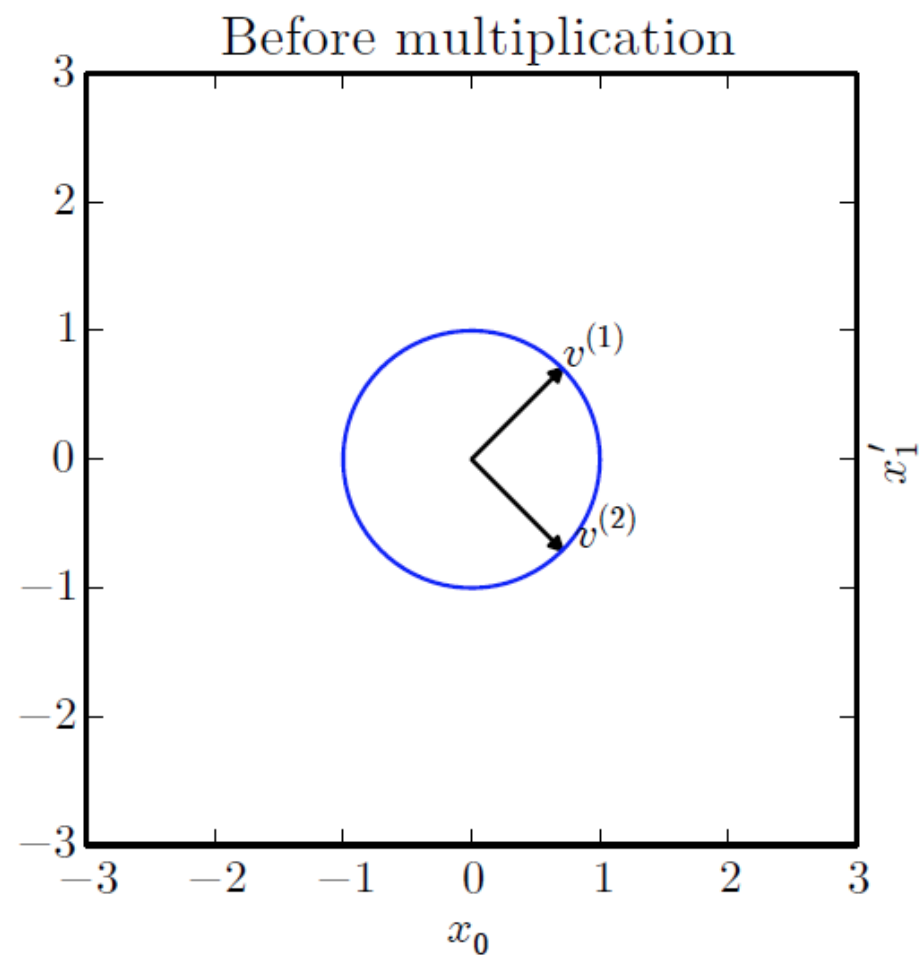
- 对角化矩阵的特征分解:

$$A = V \text{diag}(\lambda) V^{-1}$$

- 每一个实对称矩阵都有一个实值的、正交的特征分解:

$$A = Q \Lambda Q^T$$

# 特征值的影响



# 奇异值分解

---

- 类似于特征分解
- 更通用的; 矩阵不必是方形的

$$A = UDV^T$$

# 摩尔-彭若斯广义逆

$$x = A^+ y$$

- 方程组解的情况包括:
  - 仅有一个解：此时摩尔-彭若斯广义逆矩阵与逆矩阵相同
  - 无解：此时会给出解的最小误差  $\|Ax - y\|_2$
  - 多个解：此时会给出  $x$  范数最小的解

# 迹 (Trace)

---

矩阵的迹的性质:

$$Tr(A) = \sum_i A_{i,i}$$

$$Tr(A + B) = Tr(A) + Tr(B)$$

$$Tr(r \cdot A) = r \cdot Tr(A)$$

$$Tr(A) = Tr(A^T)$$

$$Tr(AB) = Tr(BA)$$

$$Tr(ABC) = Tr(CAB) = Tr(BCA)$$

# 学习线性代数

---

- 做一些相关的**练习题**
- 从大量的**求和符号**和**索引到单个条目**的练习开始
- 最后，你将能够快速而轻松地使用矩阵和向量积符号

# 目录

---

1/ 课程回顾

2/ 线性代数

3/ 概率和信息论

4/ 深度学习中的数值计算

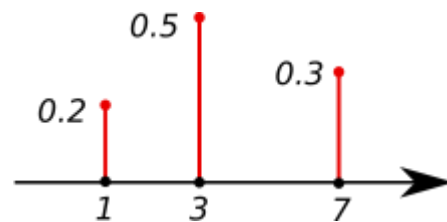
5/ 机器学习基本知识



# 概率质量函数

- 概率质量函数是**离散随机变量**在各特定取值上的概率，其中 $P$ 的域值必须为 $x$ 所有可能状态的集合
- $\forall x \in x, 0 \leq P(x) \leq 1$ 。一个**不可能事件**发生的概率为0，没有任何状态的概率小于0。同样地，一个**确定会发生的事件**的概率为1，没有其他状态比该事件发生的概率更大
- $\sum_{x \in X} P(x) = 1$ , 该性质被称为**标准化（或归一化）**。如果没有该性质，我们通过计算会发现很多事件发生的概率大于1

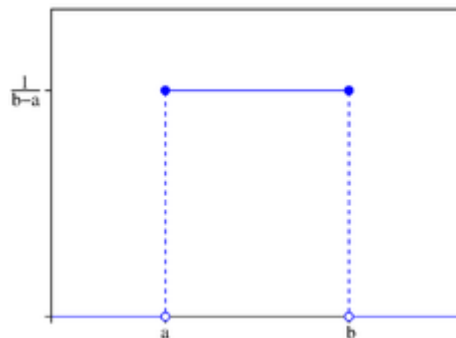
**示例:**  $P(x = x_i) = \frac{1}{k}$ , 均匀分布



# 概率密度函数

- 概率密度函数是用于描述连续型随机变量在某个确定取值点附近的可能性的函数， $P$ 的域值必须是包括所有可能状态的集合
- $\forall x \in \mathcal{X}, 0 \leq P(x)$ 。值得注意的是其不需要满足  $P(x) \leq 1$
- $\int p(x)dx = 1$

示例:  $u(x: a, b) = \frac{1}{b-a}$ , 均匀分布



# 使用求和及积分规则计算边缘概率

$$\forall x \in \mathcal{X}, P(\mathbf{x} = x) = \sum_y P(\mathbf{x} = x, \mathbf{y} = y).$$

$$p(x) = \int p(x, y) dy.$$

# 条件概率

---

$$P(y = y|x = x) = \frac{P(y = y, X=x)}{P(X=x)}$$

# 概率链式规则

---

$$P(x^{(1)}, \dots, x^{(n)}) = P(x^{(1)}) \prod_{i=2}^n P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$

# 独立性

---

$$\forall x \in \mathbf{x}, y \in \mathbf{y}, p(\mathbf{x} = x, \mathbf{y} = y) = p(\mathbf{x} = x)p(\mathbf{y} = y)$$

# 条件独立性

---

$$\begin{aligned}\forall x \in x, y \in y, z \in z, p(x = x, y = y | z = z) \\ = p(x = x | z = z)p(y = y | z = z)\end{aligned}$$

# 期望

---

$$E_{X \sim P}[f(x)] = \sum_x P(x)f(x),$$

$$E_{X \sim P}[f(x)] = \int p(x)f(x)dx,$$

期望的线性性质:

$$E_X[\alpha f(x) + \beta g(x)] = \alpha E_X[f(x)] + \beta E_X[g(x)]$$



# 方差与协方差

---

$$\text{Var}(f(x)) = E[(f(X) - E[f(x)])^2]$$

$$\begin{aligned} \text{Cov}(f(x), g(y)) \\ = E[(f(x) - E[f(x)])(g(y) - E[g(y)])] \end{aligned}$$

协方差矩阵:

$$\text{Cov}(\mathbf{x})_{i,j} = \text{Cov}(x_i, x_j)$$

# 伯努利分布

---

$$P(x = 1) = \theta$$

$$P(x = 0) = 1 - \theta$$

$$P(x = x) = \theta^x (1 - \theta)^{1-x}$$

$$E_x [x] = \theta$$

$$\text{Var}_x(x) = \theta(1 - \theta)$$

# 高斯分布

---

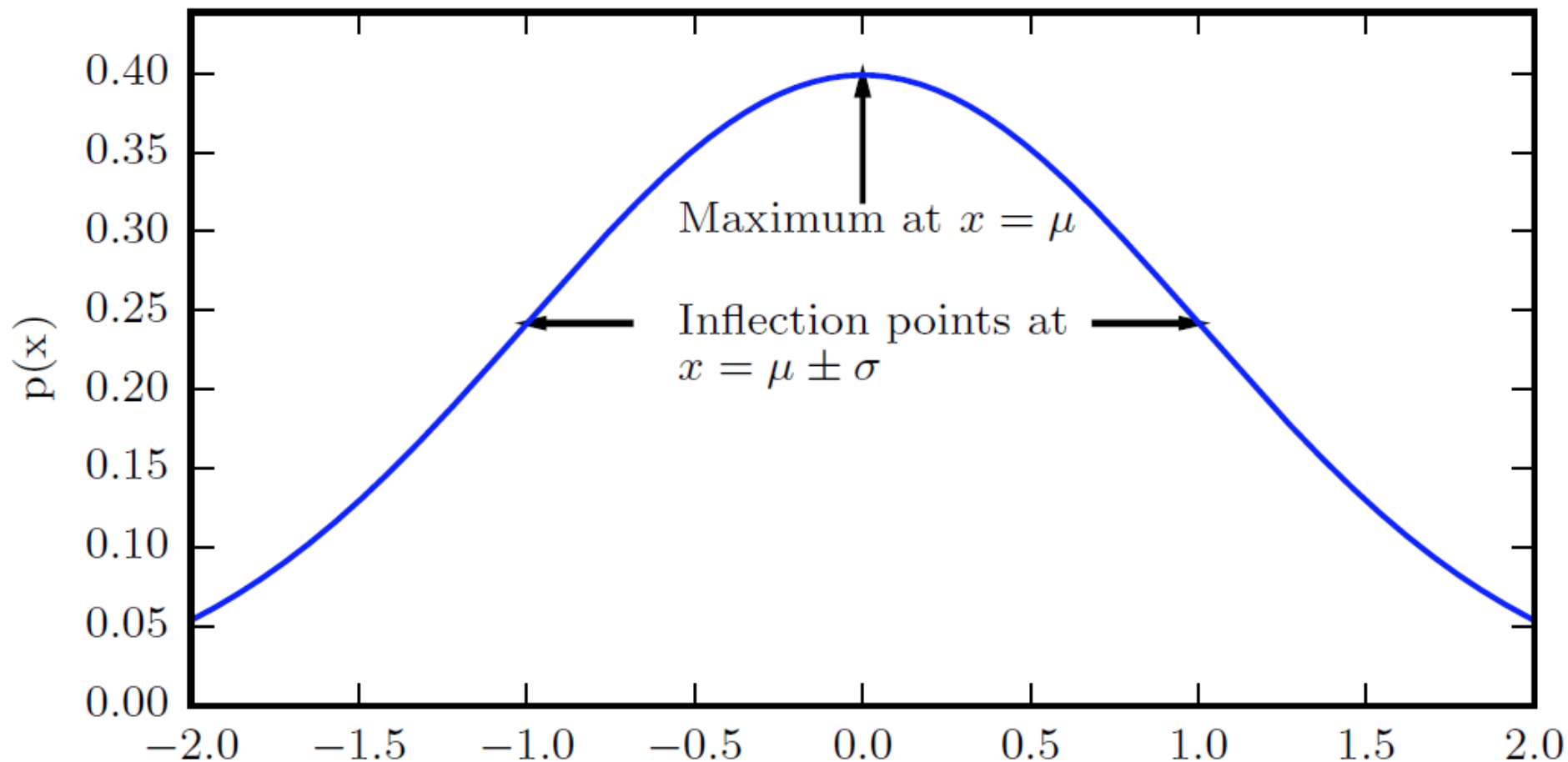
由方差参数化 (Parametrized by variance) :

$$N(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu)^2\right)$$

由精度参数化 (Parametrized by precision) :

$$N(x; \mu, \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{1}{2}\beta (x - \mu)^2\right)$$

# 高斯分布



# 多元高斯分布

由协方差矩阵参数化 (Parametrized by covariance matrix) :

$$N(x; \mu, \Sigma) = \sqrt{\frac{1}{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

由精度矩阵参数化 (Parametrized by precision matrix) :

$$N(x; \mu, \beta^{-1}) = \sqrt{\frac{\det(\beta)}{(2\pi)^n}} \exp\left(-\frac{1}{2} (x - \mu)^T \beta (x - \mu)\right)$$

# 更多概率分布

---

指数分布:

$$p(x; \lambda) = \lambda \mathbf{1}_{x \geq 0} \exp(-\lambda x)$$

拉普拉斯分布:

$$\text{Laplace}(x; \mu, \gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x-\mu|}{\gamma}\right)$$

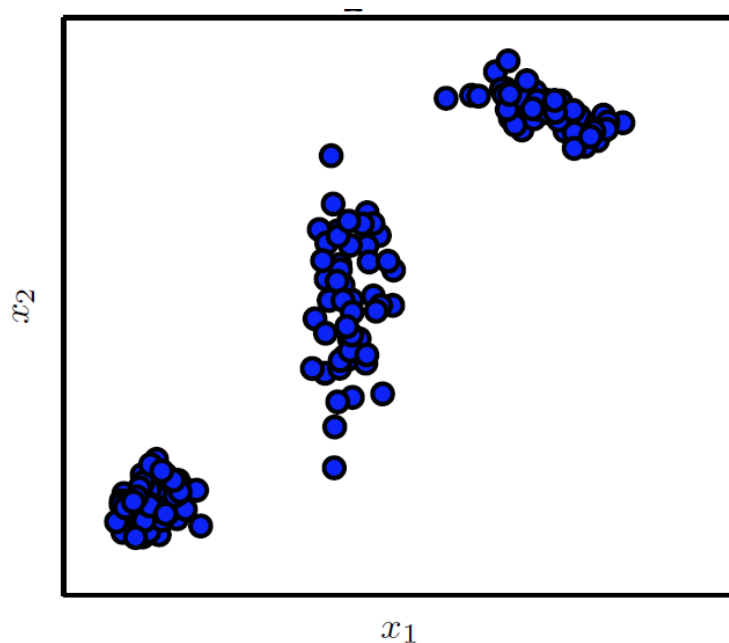
狄拉克分布:

$$p(x) = \delta(x - \mu)$$

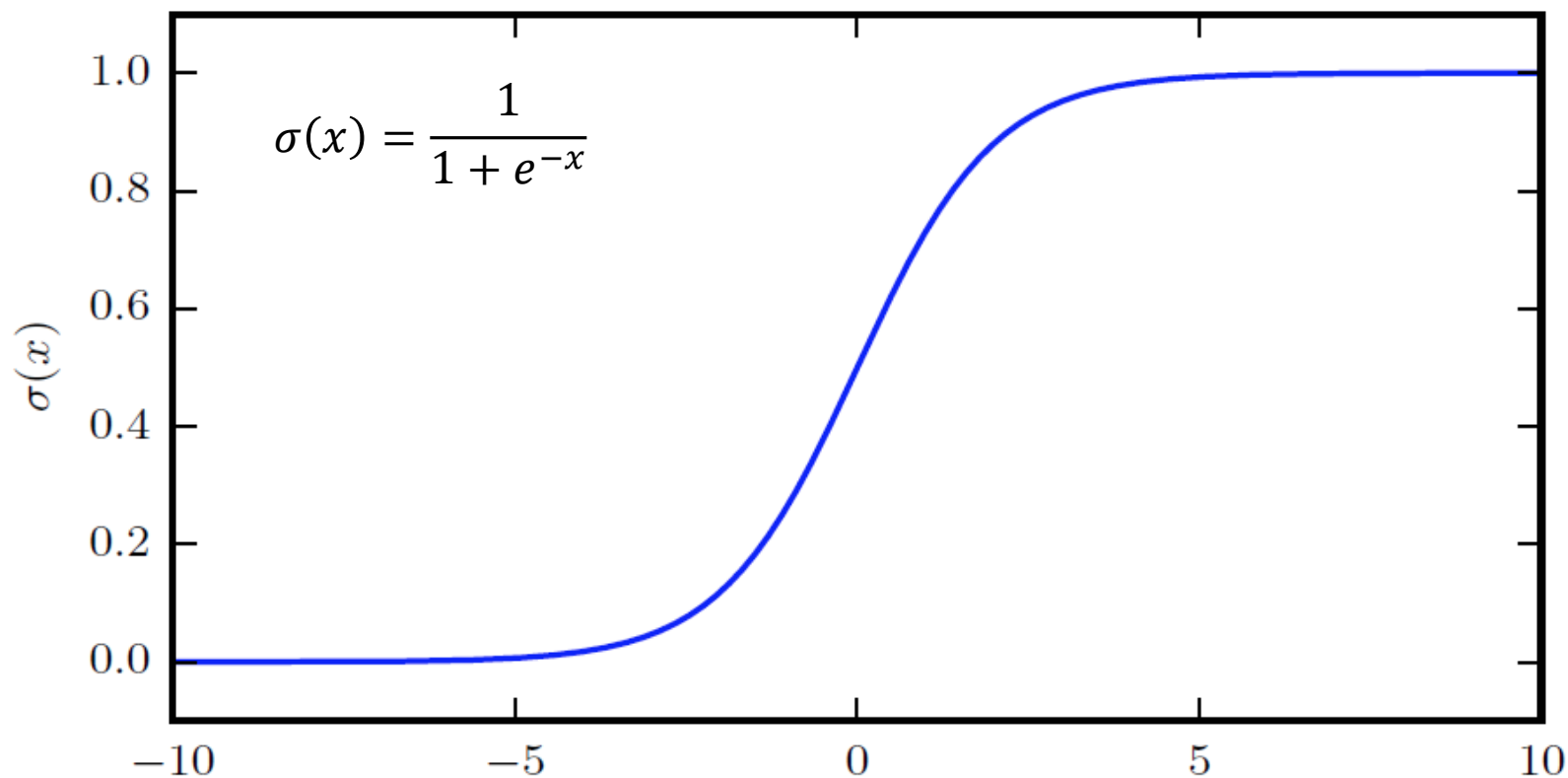
# 混合分布 (Mixture Distributions)

$$P(x) = \sum_i P(c = i)P(x|c = i)$$

由三个组件构成的高斯混合分布



# Logistic Sigmoid函数

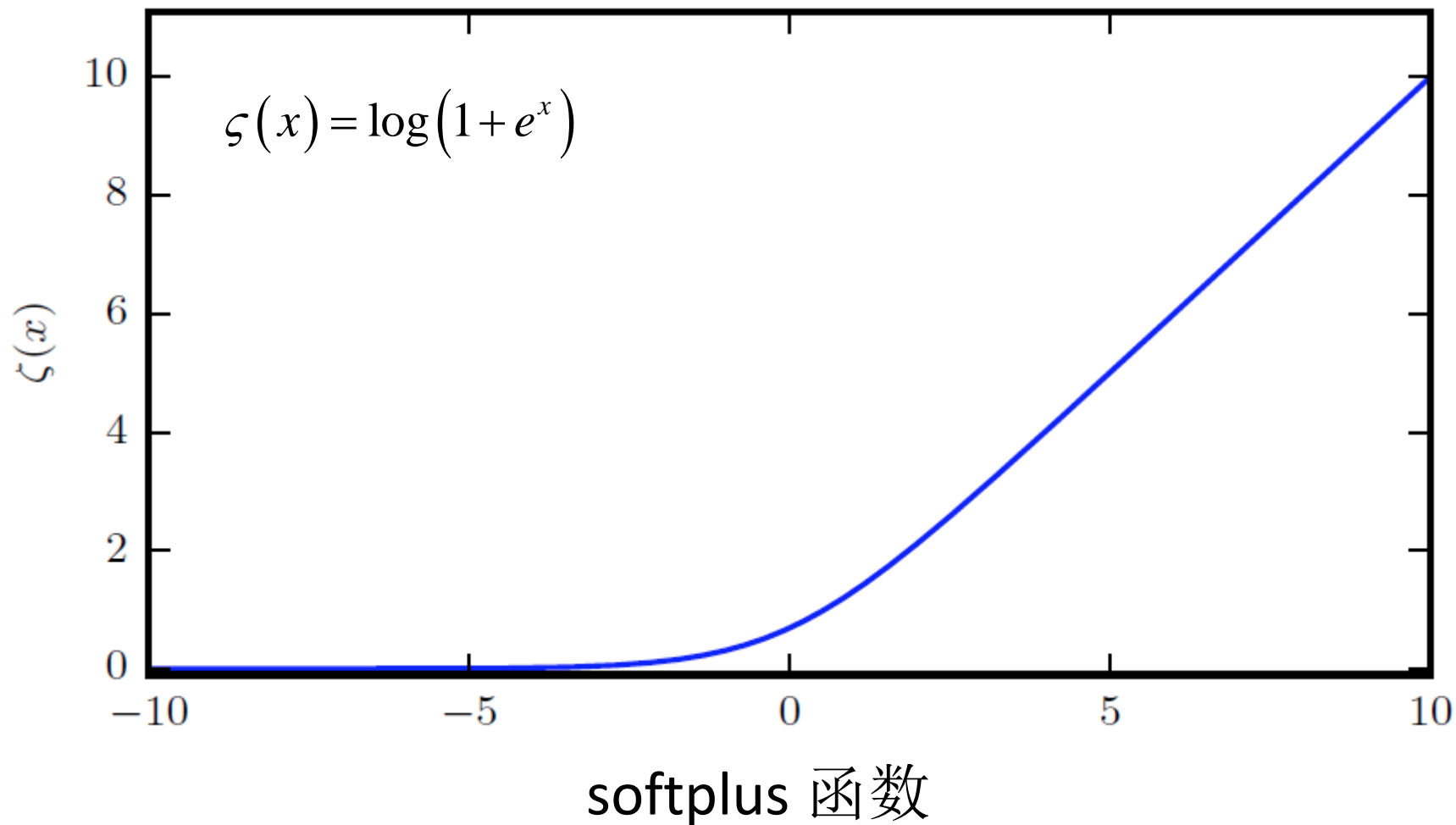


logistic sigmoid函数

通常用于产生Bernoulli分布中的参数 $\theta$



# Softplus函数



# 贝叶斯规则 (Bayes' Rule)

---

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)}$$

# 信息论 (Information Theory)

---

自信息:

$$I(x) = -\log P(x)$$

信息熵:

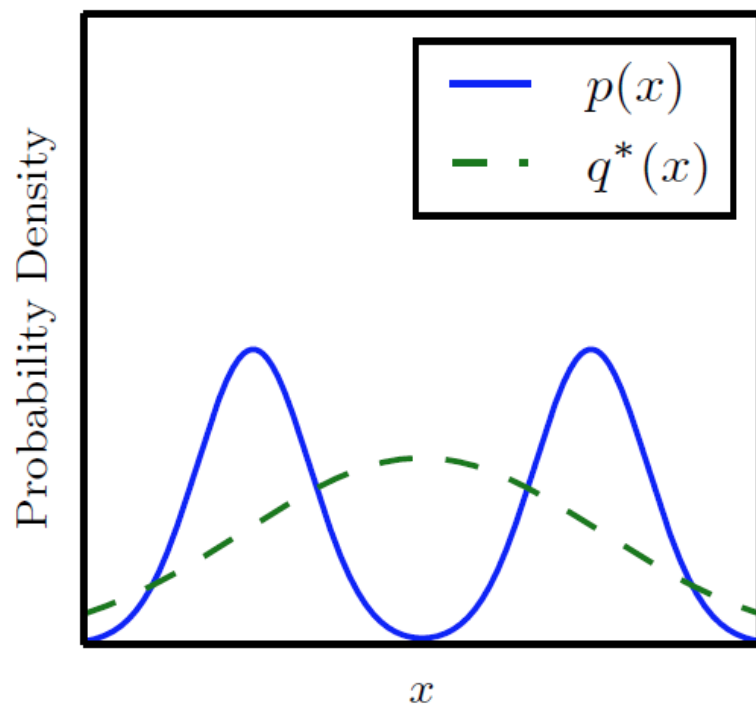
$$H(x) = E_{X \sim P}[I(x)] = E_{X \sim P}[\log P(x)]$$

KL散度:

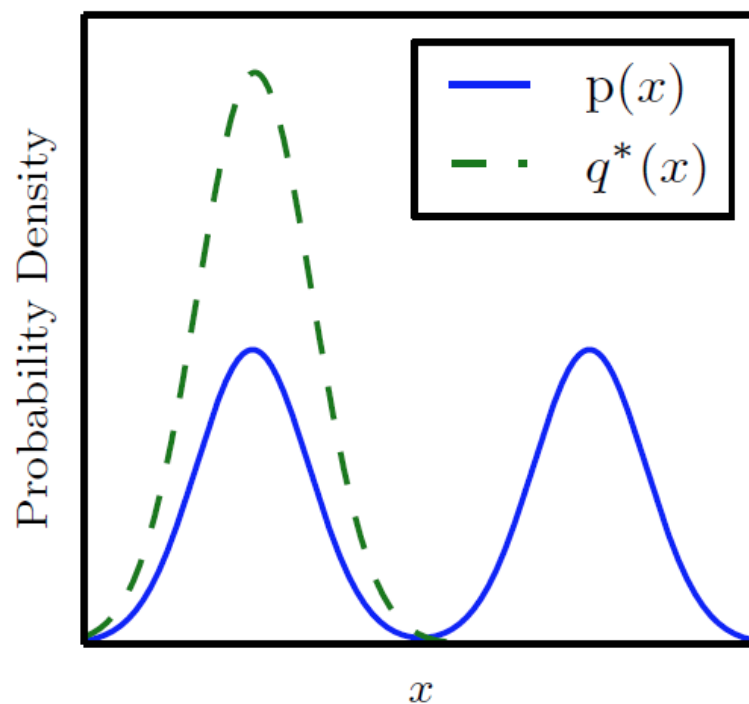
$$\begin{aligned} D_{KL}(P \parallel Q) &= E_{X \sim P} \left[ \log \frac{P(x)}{Q(x)} \right] \\ &= E_{X \sim P} [\log P(x) - \log Q(x)] \end{aligned}$$

# KL散度是不对称的

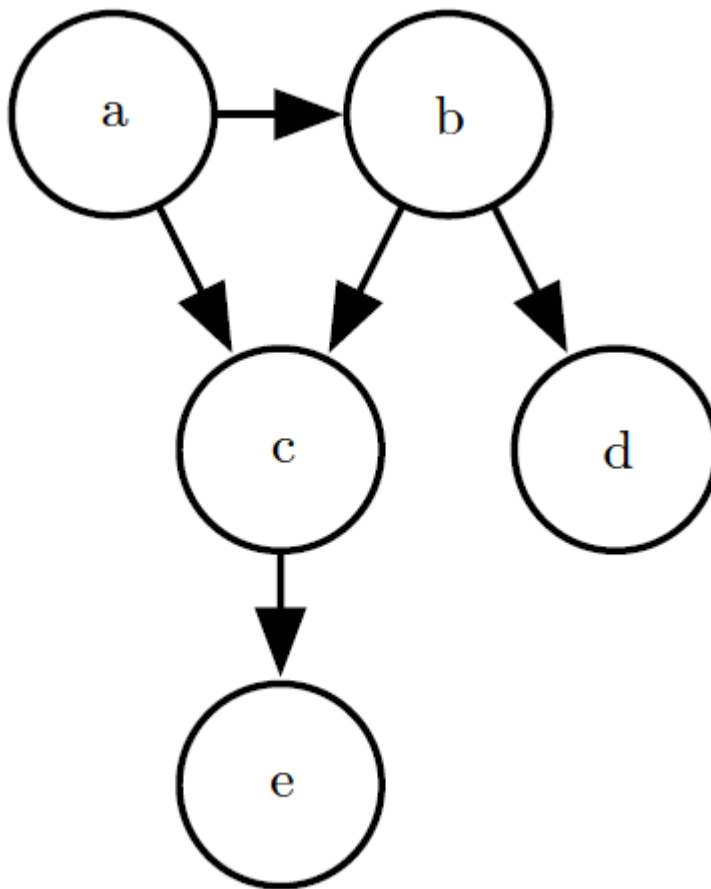
$$q^* = \operatorname{argmin}_q D_{KL}(p \parallel q)$$



$$q^* = \operatorname{argmin}_q D_{KL}(q \parallel p)$$

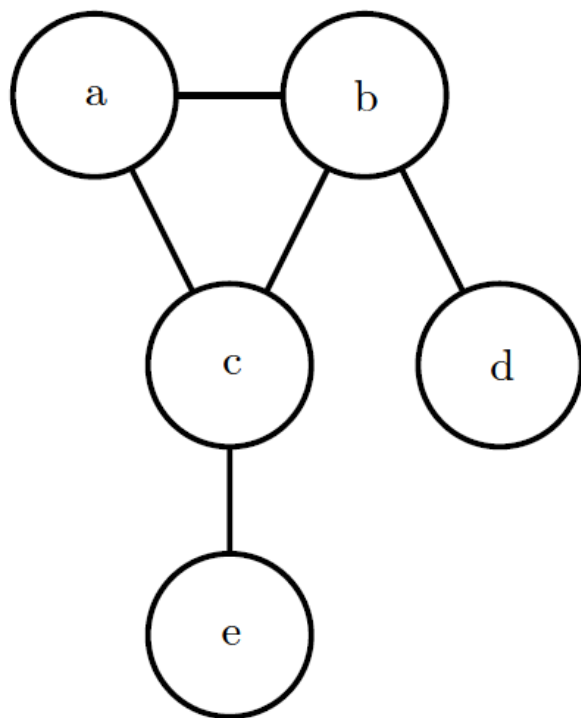


# 有向模型 (Directed Model)



$$p(a, b, c, d, e) = p(a)p(b|a)p(c|a, b)p(d|b)p(e|c)$$

# 无向模型 (Undirected Model)



$$p(a, b, c, d, e) = \frac{1}{Z} \phi^{(1)}(a, b, c) \phi^{(2)}(b, d) \phi^{(3)}(c, e)$$

# 目录

---

1/ 课程回顾

2/ 线性代数

3/ 概率和信息论

4/ 深度学习中的数值计算

5/ 机器学习基本知识

# 深度学习中的数值问题

- 算法通常用实数表示，但实数不能在有限元的计算机上实现
  - 当用有限的比特数来实现时，这个算法仍然有效吗？
- 函数输入的微小的变化，会引起函数输出的巨大变化吗？
  - 舍入误差，噪音，度量误差会引起巨大的变化
  - 迭代搜索来寻找最佳输入是困难的

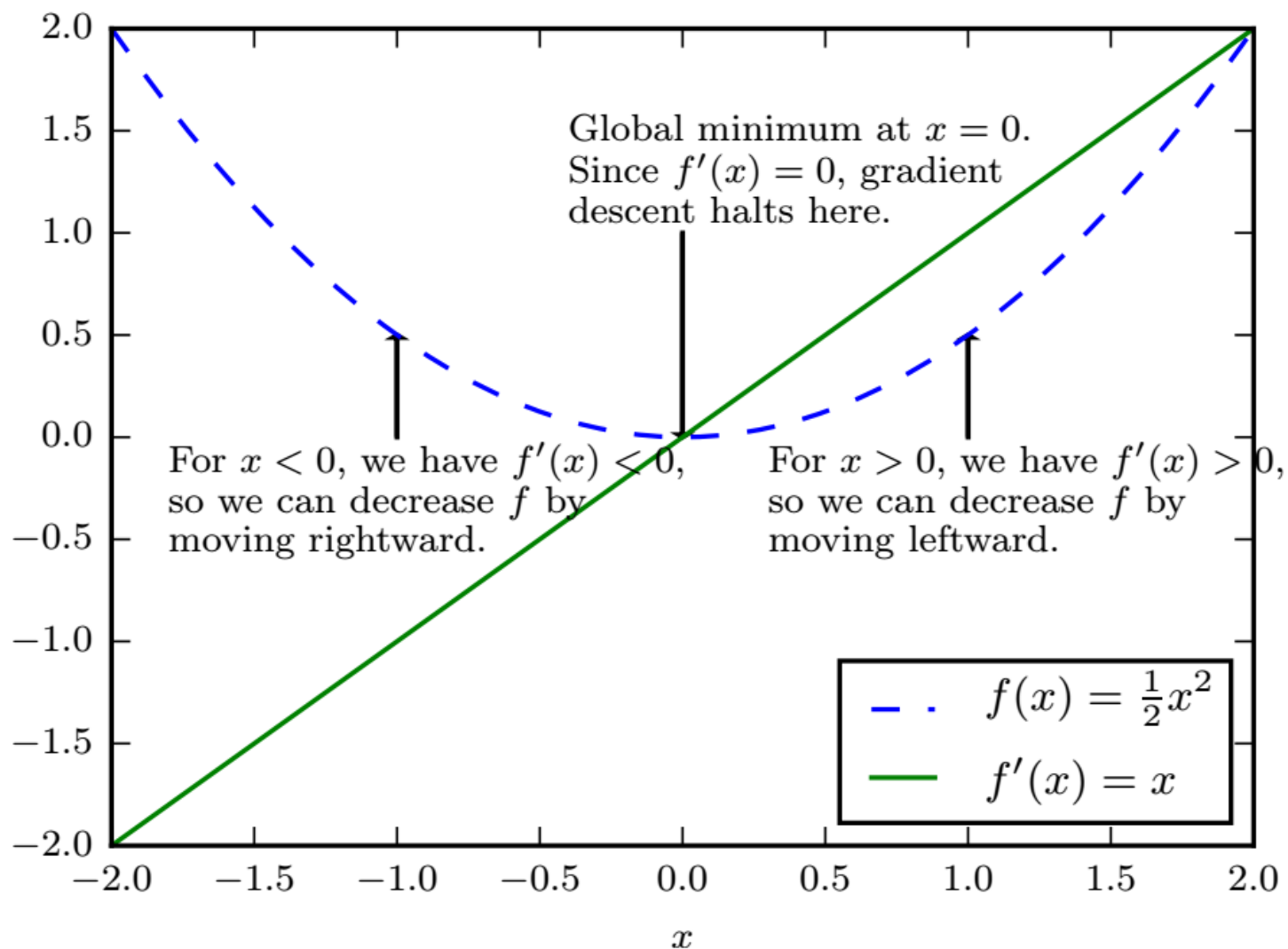


# 迭代优化

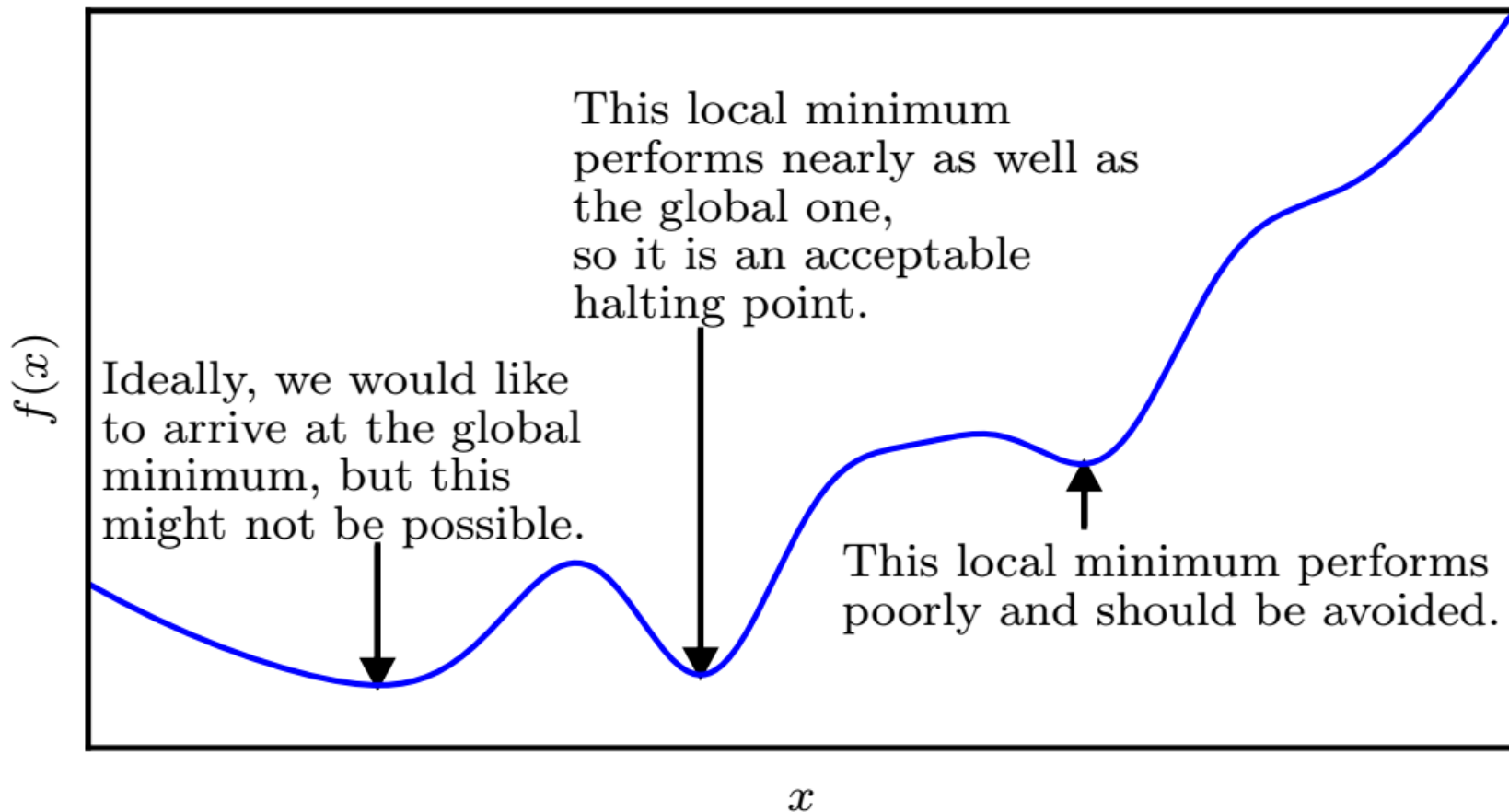
---

- 梯度下降
- 曲率

# 梯度下降

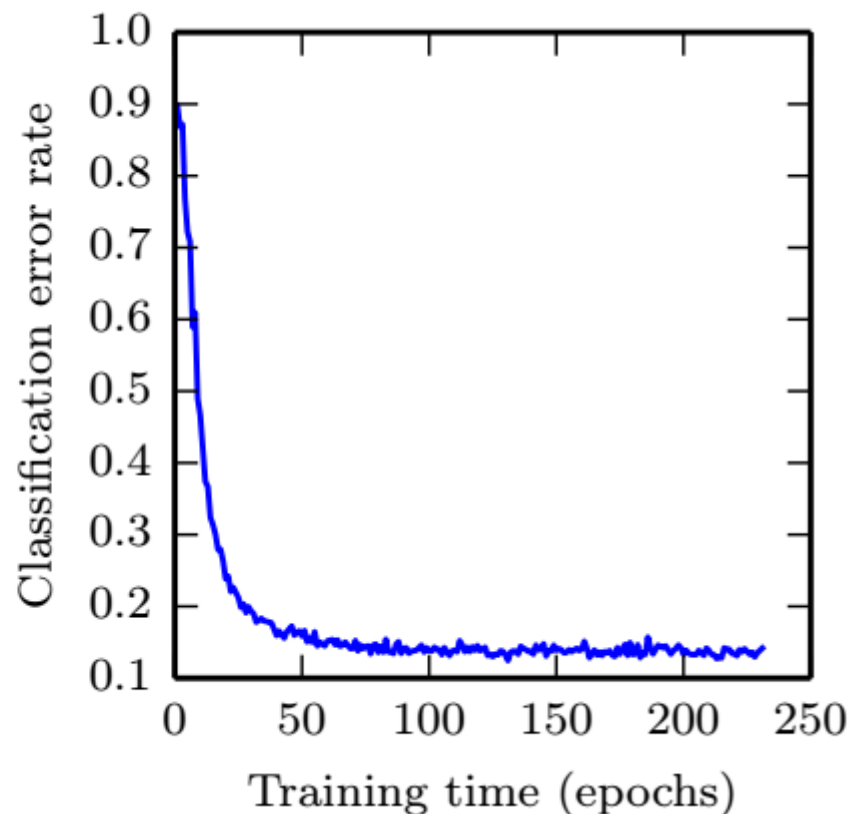
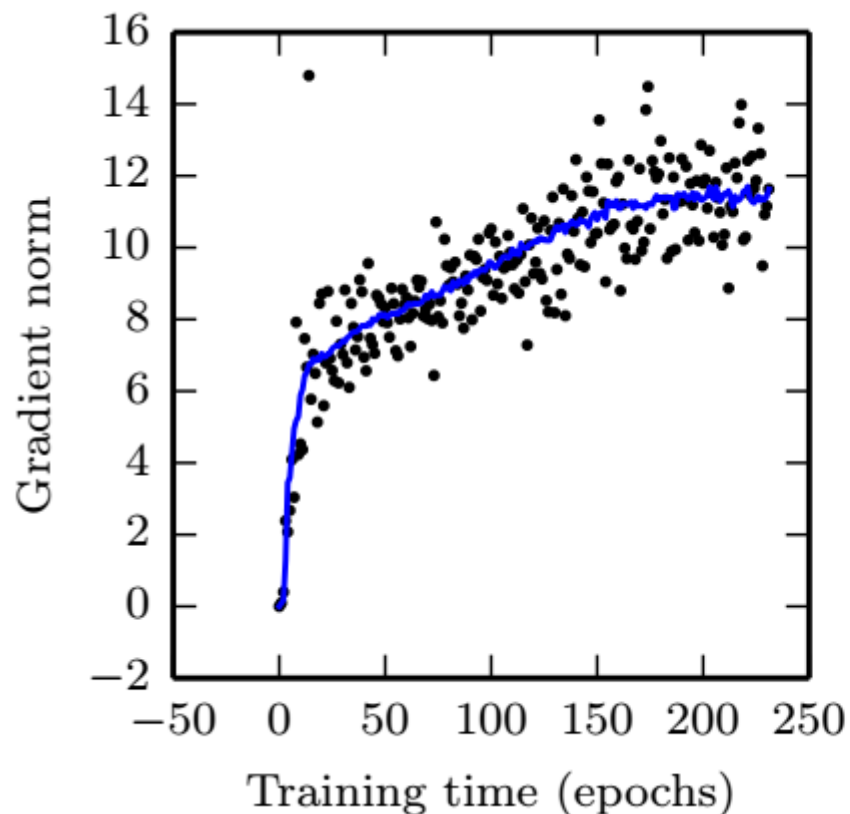


# 近似优化



# 近似优化

通常甚至不会取得局部最小值



# 深度学习的优化方法

---

- 纯数学的方法:
  - 迭代地寻找  $f(x)$  的最小值
  - 或者寻找使  $f(x)$  取得局部最小值的临界点
- 深度学习的优化方法:
  - 使  $f(x)$  的值降低很多

# 迭代优化

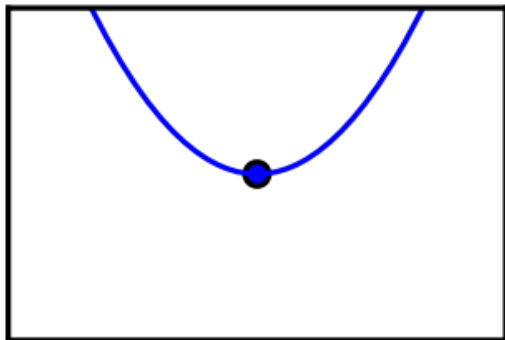
---

- 梯度下降
- 曲率

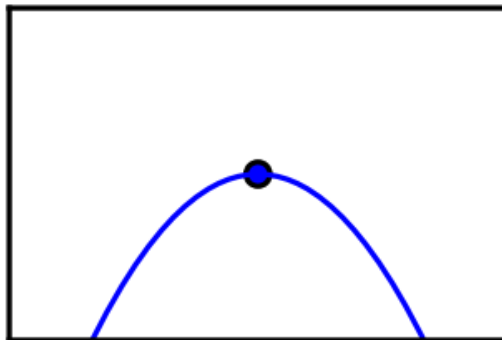
# 临界点 (Critical Points)

---

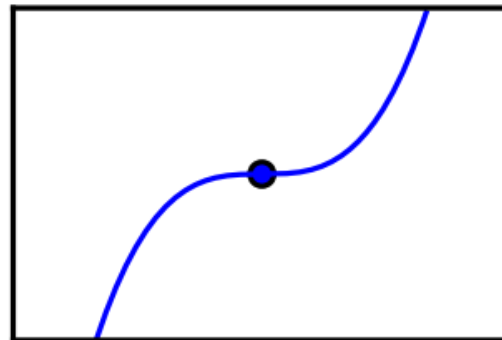
Minimum



Maximum

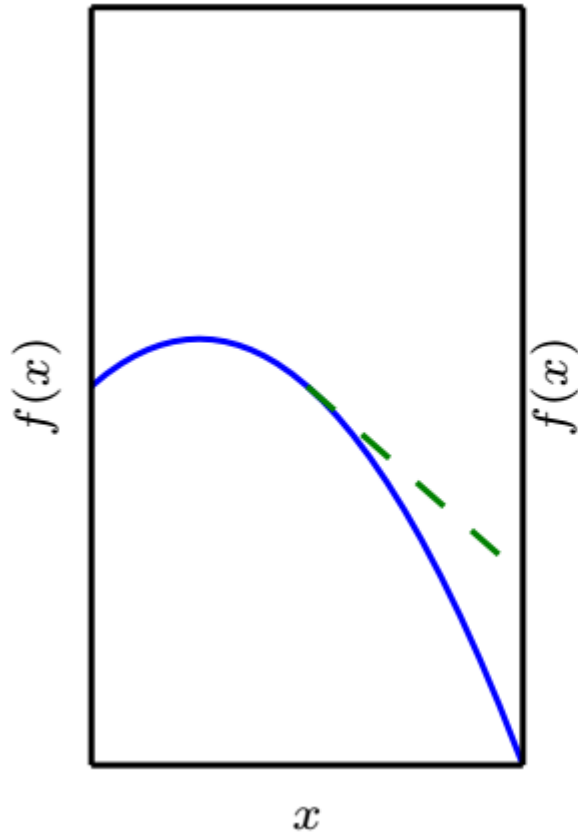


Saddle point

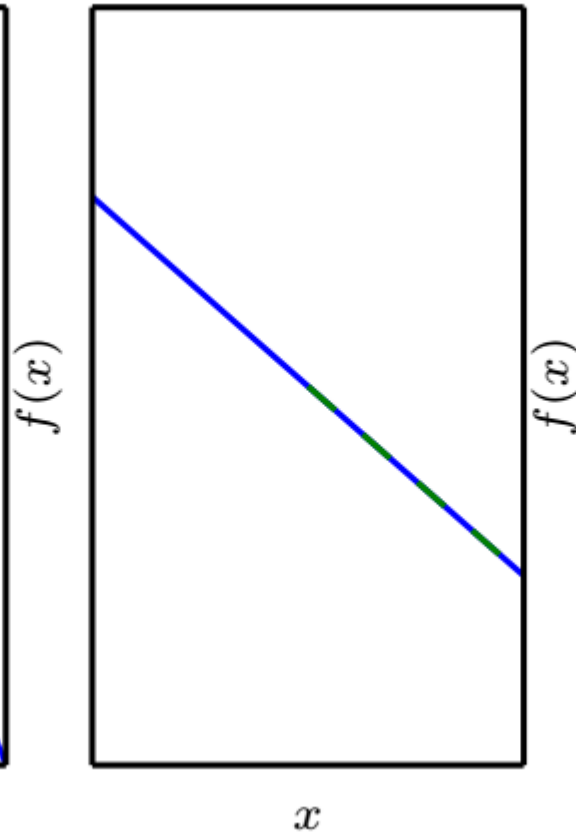


# 曲率 (Curvature)

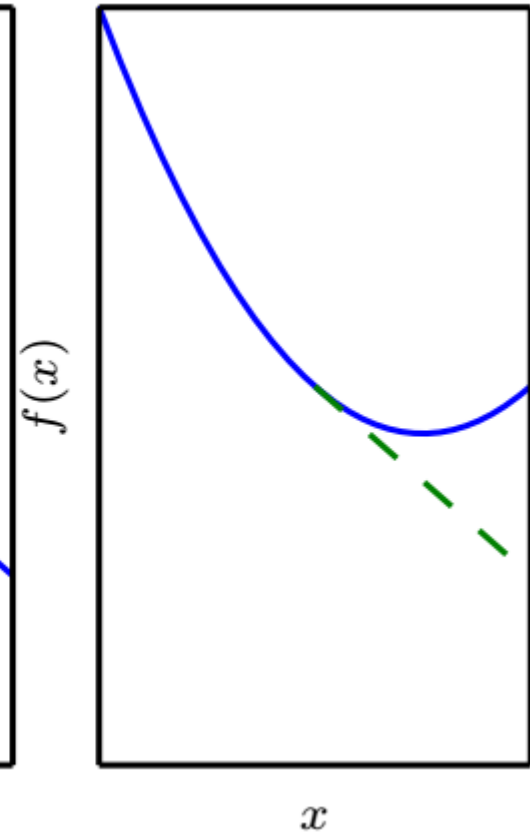
Negative curvature



No curvature

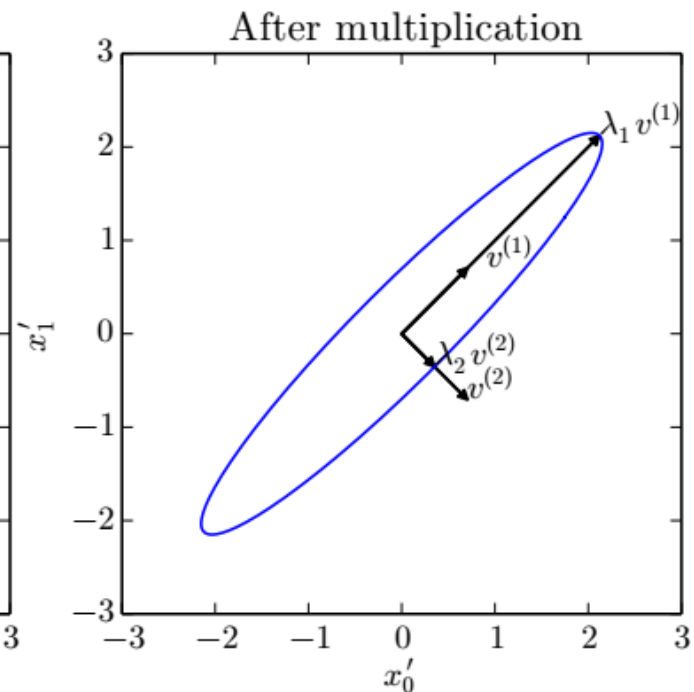
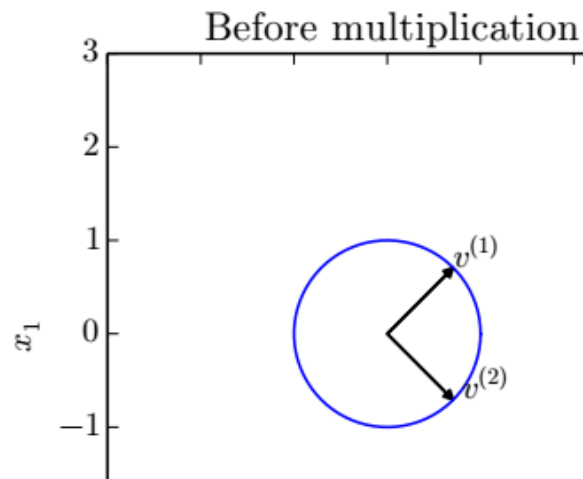


Positive curvature





# 二阶方向导数



$$Q = [v_1, \dots, v_n]$$

$$H = Q\Lambda Q^\top$$

Second derivative in direction  $d$ :

$$d^\top H d = \sum_i \lambda_i \cos^2 \angle(v_i, d)$$

# 预测最佳步长

使用泰勒级数

$$f(x^{(0)} - \epsilon g) \approx f(x^{(0)}) - \epsilon g^\top g + \frac{1}{2} \epsilon^2 g^\top H g.$$

$$\epsilon^* = \frac{g^\top g}{g^\top H g}.$$

大的梯度加速收敛

如果与它们的特征向量对齐，  
大的特征值将会减速

# 目录

---

1/ 课程回顾

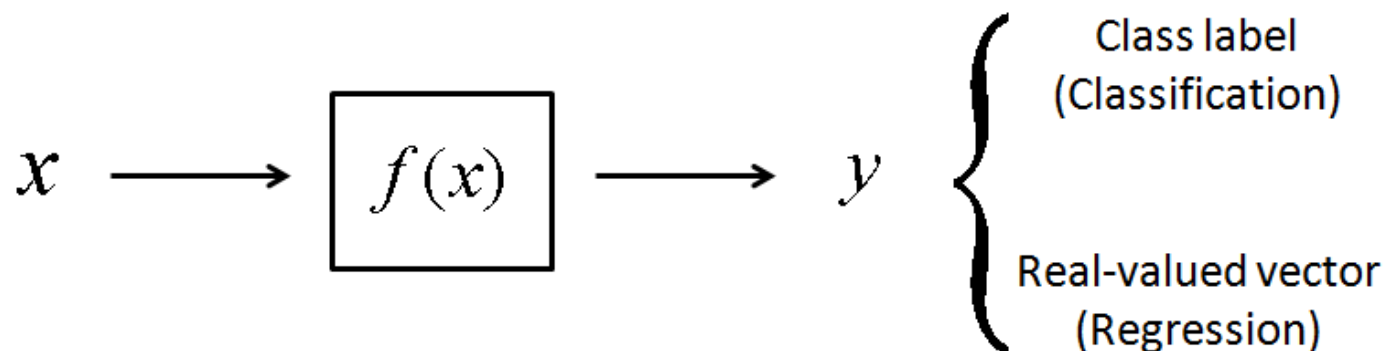
2/ 线性代数

3/ 概率和信息论

4/ 深度学习中的数值计算

5/ 机器学习基本知识

# 机器学习 (Machine Learning)



Object recognition

{dog, cat, horse, flower, ...}



Low-resolution image

Super resolution



High-resolution image

# 监督学习和无监督学习

- 监督学习: 学习目标是用输入-标签对  $(x; y)$  学习一个在给定输入时可以预测标签（或标签的概率分布）的函数  $f$ , 即  $\hat{y} = f(x)$
- 无监督学习: 不提供标签或其他目标。数据由样本集  $x$  组成, 目标是学习  $x$  自身的统计结构
- 弱监督学习: 与监督学习一样, 训练集包括输入-标签对  $(x; y)$ , 但是标签  $y$  要么不可靠的出现（有缺失值）, 要么有噪声（给定的标签不是正确的标签）

# 无监督学习

- 找出保留 $x$ 信息尽可能多的，同时又比  $x$  “简单” 的“最佳” 数据表示。以线性情况为例

$$\tilde{\mathbf{x}} = a_0 + \sum_{i=1}^{d'} a_i \mathbf{e}_i$$

- 更低维度的表示:  $d' < d$
  - 稀疏表示: 非零 $a_i$  的数目很少
  - 独立的表达: 解开潜在数据分布的变量来源，使表达的各个维度之间是统计独立的，即如果  $i \neq j$ ，则 $a_i$  和  $a_j$  是统计独立的
- 深度学习是以非线性和层次化的方式学习数据表达

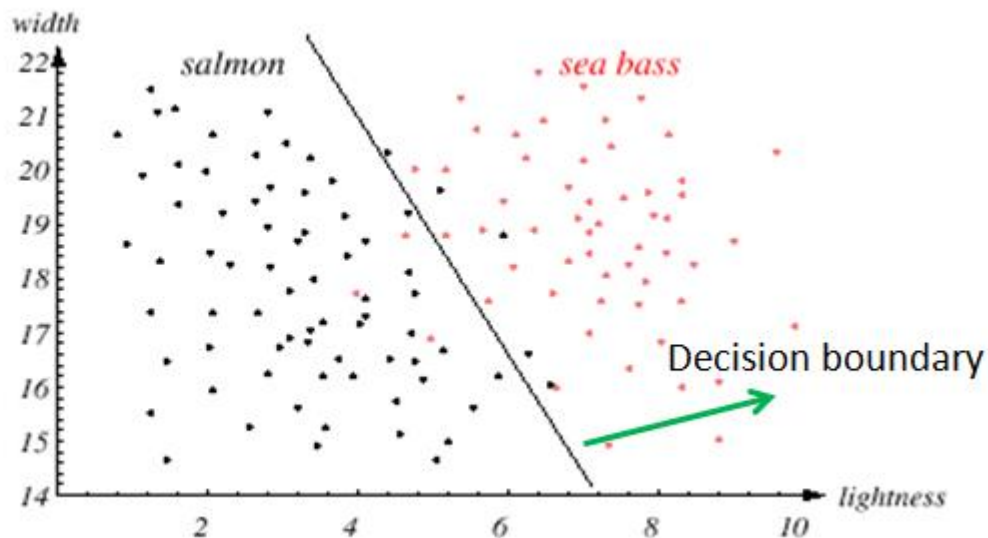
# 分类 (Classification)

- $f(x)$  预测  $x$  所属的类别

$$f : \mathcal{R}^D \rightarrow \{1, \dots, K\}$$

- $f(x)$  由决策边界决定
- 作为一个变体, 在给定  $x$  时,  $f$  可以预测类别的概率分布,  $f(x) = P(y|x)$ 。预测的类别为:

$$y^* = \arg \max_k P(y = k | \mathbf{x})$$



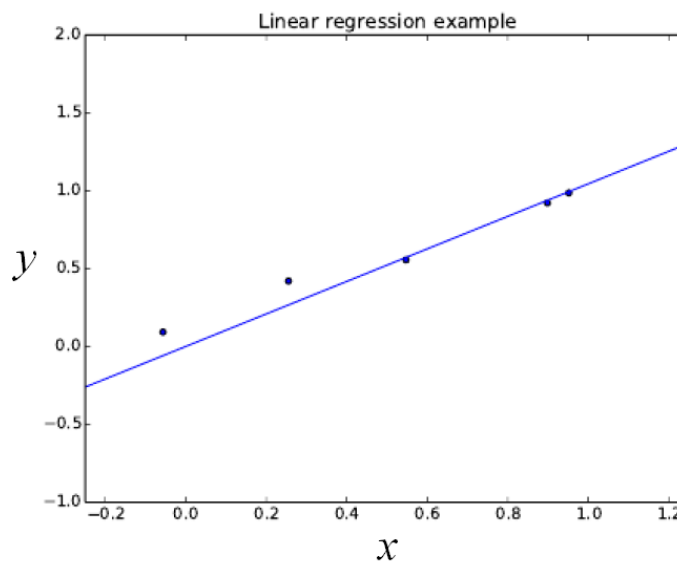
# 回归 (Regression)

- 预测实值输出

$$f : \mathcal{R}^D \rightarrow \mathcal{R}^M$$

- 示例: 线性回归

$$y = \mathbf{w}^t \mathbf{x} = \sum_{d=1}^D w_d x_d + w_0$$





# 判别模型 (Discriminative model)

- 判别模型  $P(y|x)$  和决策边界
- 学习判别函数  $g_k(x)$

$$y = \arg \max_k g_k(\mathbf{x})$$

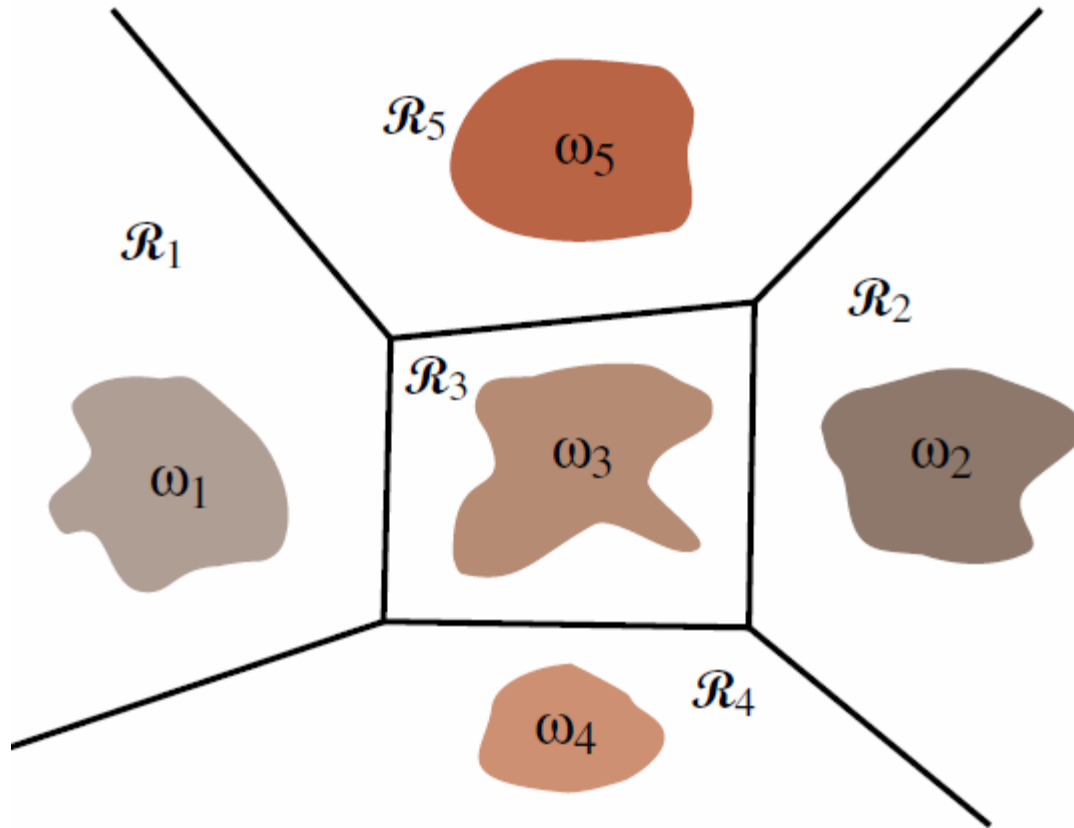
- 在线性情况下,  $g_k(x) = w_k^t x$
- $P(y|x)$  可以由线性判别函数来估计, 该函数也被称为 softmax 函数

$$P(y = j|\mathbf{x}) = \frac{e^{w_j^t \mathbf{x}}}{\sum_{k=1}^K e^{w_k^t \mathbf{x}}}$$

- 示例: SVM, boosting, 神经网络 (NN)

# 判别模型 (Discriminative model)

- 判别模型更容易拟合数据



# 判别模型 (Discriminative model)

- 参数  $\theta = \{w_k\}$  可以通过最大化数据似然函数来进行估计

$$\hat{\theta} = \arg \max_{\theta} P(\mathcal{D}_n | \theta) = \arg \max_{\theta} \prod_{i=1}^n P(y_n^{(\text{train})} | \mathbf{x}_n^{(\text{train})}, \theta)$$

- 最大后验估计 (MAP)

$$\theta = \arg \max_{\theta} p(\theta | \mathcal{D}_n) = \arg \max_{\theta} \log P(\mathcal{D}_n | \theta) + \log p(\theta)$$

- 根据贝叶斯规则, 即  $p(\theta | \mathcal{D}_n) = P(\mathcal{D}_n | \theta)p(\theta) / P(\mathcal{D}_n)$

$$\theta = \arg \max_{\theta} \log P(\mathcal{D}_n | \theta) + \log p(\theta)$$

$$\theta = \arg \max_{\theta} \sum_{i=1}^n \log P(y_n^{(\text{train})} | \mathbf{x}_n^{(\text{train})}, \theta) + \log p(\theta)$$

- 先验概率  $p(\theta)$  对应一个正则项, 例如,

$$p(\theta) = e^{-\lambda \|\theta\|^2}$$

# 生成模型 (Generative model)

- 估计潜在类的条件概率密度  $p(x|y = k)$ , 然后用贝叶斯决策理论构建分类器

$$P(y = k|\mathbf{x}) = \frac{p(\mathbf{x}|y = k)P(y = k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y = k)P(y = k)}{\sum_{k'=1}^K p(\mathbf{x}|y = k')P(y = k')}$$

- $p(x|y)$  和  $P(y)$  可以由  $\theta$  参数化
- 先验概率  $P(y)$  可以建模预测之间的依赖关系, 比如像素的分割标签或语音序列的预测
- 建模类的条件概率密度是更困难的。然而它为模型拟合添加了更强的正则项, 因为所学模型不仅需要预测类别标签, 而且需要生成输入数据。
- 在设计模型  $p(x|y)$  时, 更容易添加域知识

# 训练 (Training)

- 训练: 从训练集 $\{(x_i^{(train)}, y_i^{(train)})\}$ 中估计 $f$ 的参数
  - 决策边界,  $P(y|x)$ 的参数, 线性回归中的 $w$
- 在训练集上优化目标函数。在训练集上的性能度量可能不同于测试集上的性能度量, 比如, 在分类任务中, 训练时的性能度量为交叉熵损失, 而测试的性能度量为错误预测率
  - 线性回归中的均方误差 (MSE)

$$\text{MSE}_{\text{train}} = \frac{1}{N} \sum_i \|\mathbf{w}^t \mathbf{x}_i^{(\text{train})} - y_i^{(\text{train})}\|_2^2$$

- 分类中的交叉熵损失(CE)

$$\text{CE}_{\text{train}} = \frac{1}{N} \sum_i \log P(y = y_i^{(\text{train})} | \mathbf{x}_i^{(\text{train})})$$

# 优化 (Optimization)

- 目标函数的选择应该有利于优化
- 以线性回归为例

$$\nabla_{\mathbf{w}} \text{MSE}_{\text{train}} = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\mathbf{w} = (\mathbf{X}^{(\text{train})t} \mathbf{X}^{(\text{train})})^{-1} \mathbf{X}^{(\text{train})t} \mathbf{y}^{(\text{train})}$$

where  $\mathbf{X}^{(\text{train})} = [\mathbf{x}_1^{(\text{train})}, \dots, \mathbf{x}_N^{(\text{train})}]$  and  $\mathbf{y}^{(\text{train})} = [y_1^{(\text{train})}, \dots, y_N^{(\text{train})}]$ .

# 函数估计 (Function estimation)

- 我们感兴趣的是从输入  $x$  预测输出  $y$ , 假设存在一个函数来描述  $y$  和  $x$  之间的关系, 例如,  $y = f(x) + \epsilon$ , 其中  $\epsilon$  是服从某种分布的随机噪声
- 预测函数  $f$  可以被参数向量  $\theta$  参数化
- 从训练集  $D_n = \{(x_1^{(train)}, y_1^{(train)}), \dots, (x_n^{(train)}, y_n^{(train)})\}$  中估算  $\hat{f}_n$ , 相当于从  $D_n$  中估算  $\hat{\theta}_n$
- 由于  $D_n$  随机生成于一种潜在的概率分布, 所以  $\hat{\theta}$  和  $\hat{f}$  都是服从某种概率分布的随机变量 (或者为向量, 或者为函数)
- 相比于 “真实” 的参数向量  $\theta$  或函数  $\hat{f}$ , 估算的质量可以用偏差和方差来度量
- 如果有更好的函数参数形式的设计, 学习器尽管有比较小的容量, 但仍能取得较低的泛化误差
- 这个设计过程涉及到域知识

# 偏差 (Bias)

$$\text{bias}(\hat{\theta}) = E(\hat{\theta}) - \theta$$

期望是在潜在概率分布上采样的、大小为n的全部训练集上求得的

- 如果  $E(\hat{\theta}) = \theta$  , 则该估计称为无偏估计
- 示例: 高斯分布。若高斯分布为  $p(x_i; \theta) = N(\theta, \Sigma)$  , 则  $\theta$  估计子为  $\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i^{(\text{train})}$

$$E(\hat{\theta}) = E \left[ \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i^{(\text{train})} \right] = \frac{1}{n} \sum_{i=1}^n E \left[ \mathbf{x}_i^{(\text{train})} \right] = \frac{1}{n} \sum_{i=1}^n \theta = \theta$$



# 方差 (Variance)

$$\text{Var}[\hat{\theta}] = E[(\hat{\theta} - E[\hat{\theta}])^2] = E[\hat{\theta}^2] - E[\hat{\theta}]^2$$

- 方差通常随着数据集的增大而减小
- 偏差和方差都是估计误差的来源

$$\text{MSE} = E[(\hat{\theta} - \theta)^2] = \text{Bias}(\hat{\theta})^2 + \text{Var}[\hat{\theta}]$$

- 增加学习器的容量可能也会增加方差，尽管有更好的机会去包括真实的函数

# 泛化性 (Generalization)

- 我们更关心模型在**先前未见过**的新样本上的性能
- 训练样本通常不可能包括所有可能的输入配置，所以学习器必须要具备从训练样本泛化到新样本的能力
- 泛化误差: **所有样本**的期望误差
- 为获取机器学习算法泛化时的理论保证，我们假定所有的样本**均服从**  $p(x, y)$  分布，然后在该分布上以取**期望**的方式计算预测函数  $f$  的泛化误差

$$GE_f = \int_{\mathbf{x}, y} p(\mathbf{x}, y) \mathbf{Error}(f(\mathbf{x}), y)$$

# 泛化性 (Generalization)

- 然而, 实际上  $p(x, y)$  是未知的, 我们用测试集  $\{x_i^{(test)}, y_i^{(test)}\}$  来评估泛化性能

$$\text{Performance}_{\text{test}} = \frac{1}{M} \sum_{i=1}^M \text{Error}(f(\mathbf{x}_i^{(test)}), y_i^{(test)})$$

- 我们希望测试样本和训练样本都服从相同的概率分布  $p(x, y)$ , 尽管测试样本是未知的

# 容量 (Capacity)

- 学习器/模型从一系列函数中发现一个函数的能力。通俗来说，就是模型拟合样本数目（或各种函数）的能力。示例：

- 线性预测器

$$y = wx + b$$

- 二次多项式预测器

$$y = w_2 x^2 + w_1 x + b$$

- 十次多项式预测器

$$y = b + \sum_{i=1}^{10} w_i x^i$$

- 后面的系列更丰富，能拟合更复杂的函数

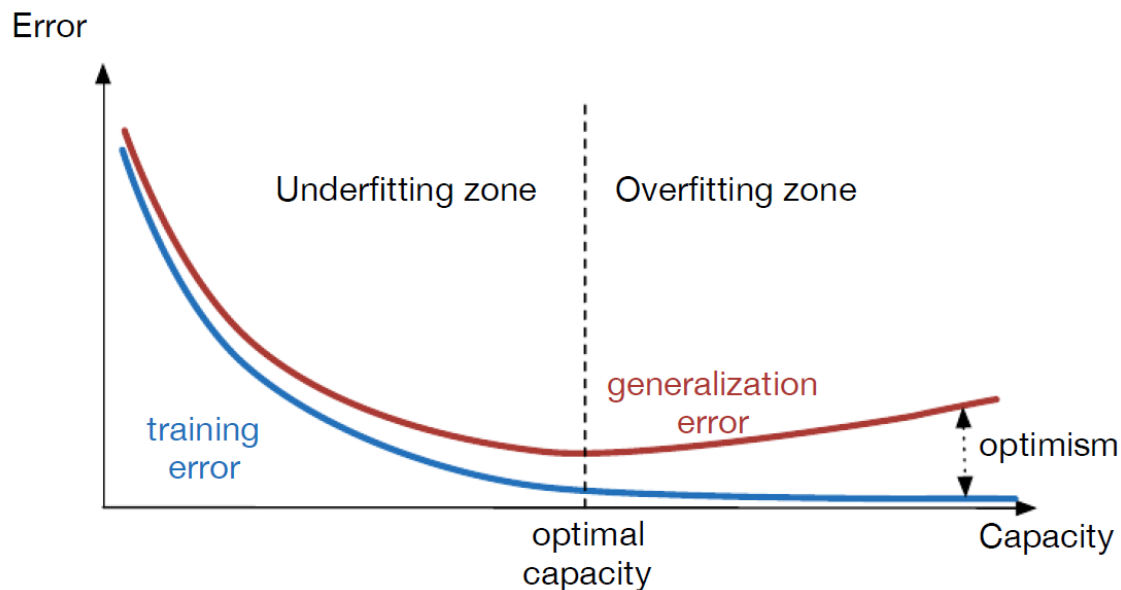
- 容量可以用学习器能拟合训练样本  $\{x_i^{(train)}, y_i^{(train)}\}$  的数目来度量，而且不管如何改变  $x_i^{(train)}$  和  $y_i^{(train)}$  的值

# 最佳容量 (Optimal capacity)

---

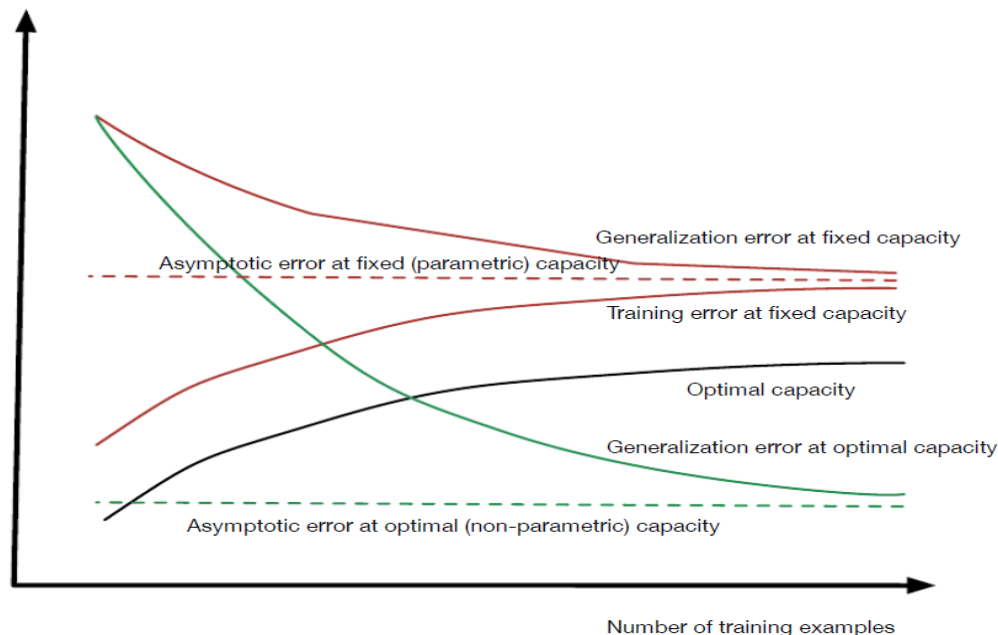
- 训练误差和泛化误差之间的差异会随着学习器容量的扩大而增加
- 泛化误差是容量的U形函数
- 优化容量与从欠拟合到过拟合的过渡相关联
  - 可以使用验证集经验地监控泛化误差
- 优化容量应该随着训练样本数目的增加而增加

# 最佳容量 (Optimal capacity)



上图展示了**训练误差**及**泛化误差**（或测试误差）与**容量**之间的典型关系。随着容量的增加，训练误差在减小，但 optimism 在增大（optimism是指训练误差和泛化误差之间的差值）。在某些情况下，optimism的增加要比训练误差的降低大，特别是当训练误差比较低，以至于不能再减少的情况。模型进入**过拟合状态**时，容量会很大，并高于最佳容量。当模型的容量低于最佳容量时，模型处于**欠拟合状态**。

# 最佳容量 (Optimal capacity)



随着训练样本的增加，模型的**最佳容量**（黑实曲线）也随之增加（可以提供更大、更灵活的模型），最佳容量模型对应的**泛化误差**（绿实曲线）也随之降低，最终到达（非参数的）**渐近误差值**（绿虚线）。

如果模型的**容量固定**（参数设置），增加训练样本的数目将会降低其对应的**泛化误差**（上部红实曲线），但不会像之前那么快，其对应的**训练误差**（下部红实曲线）将会缓慢增加，最终两者都会到达容量固定模型的**渐近误差值**（红虚线），该渐进误差值对应某类已知函数的最优解。

# 问题

---

- 欠拟合
- 过拟合
- 维度灾难



# 欠拟合 (Underfitting)

- 学习器找不到很好地拟合训练样本的解
  - 例如, 当  $y_i^{(train)}$  是  $x_i^{(train)}$  的二次函数, 而用线性回归来拟合训练样本  $\{x_i^{(train)}, y_i^{(train)}\}$
- 欠拟合意味着学习器不能抓住数据的一些重要的方面
- 欠拟合发生的原因
  - 模型不够丰富
  - 在训练集上很难找到目标函数的全局最优点, 而且容易陷入局部最优点
  - 计算资源的限制 (在迭代优化的过程中没有足够的训练迭代次数)
- 欠拟合通常发生于具有大规模训练数据的深度学习中, 在某些情况下比过拟合的问题更加严重

# 问题

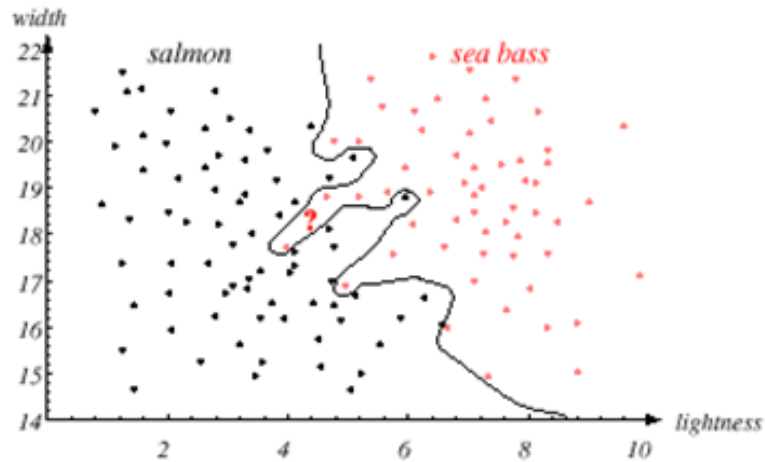
---

- 欠拟合
- 过拟合
- 维度灾难

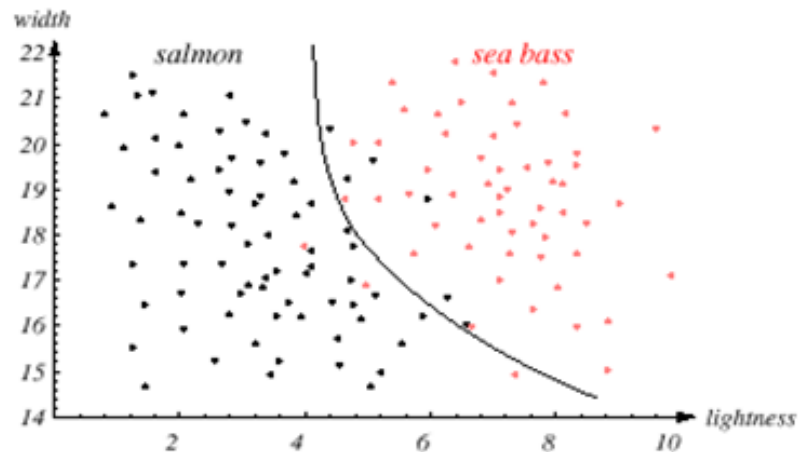
# 过拟合 (Overfitting)

- 学习器可以很好地拟合训练数据，但是泛化能力会变差。即有很小的训练误差，但是有很大的泛化误差
- 容量 (capacity) 大的学习器往往容易过拟合
  - 与训练集数据的规模相比，函数族的规模太过庞大，而且其中有很多函数都能很好地拟合训练数据
  - 没有足够的数据，学习器无法区分哪一个模型是最合适的，并且会在那些看起来不错的模型中做出任意选择
  - 单独的验证集有助于选择更合适的模型
  - 在大多数情况下，数据会被噪音污染。大容量的学习器会倾向于描述随机误差和噪音，而不是数据的潜在模型和类

# 过拟合 (Overfitting)



Overly complex models lead to complicated decision boundaries. It leads to perfect classification on the training examples, but would lead to poor performance on new examples.



The decision boundary might represent the optimal tradeoff between performance on the training set and simplicity of classifier, therefore giving highest accuracy on new examples.

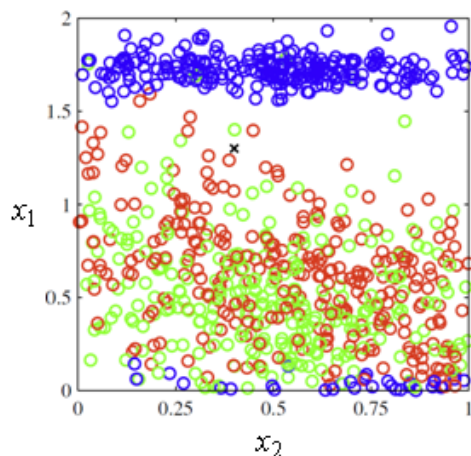
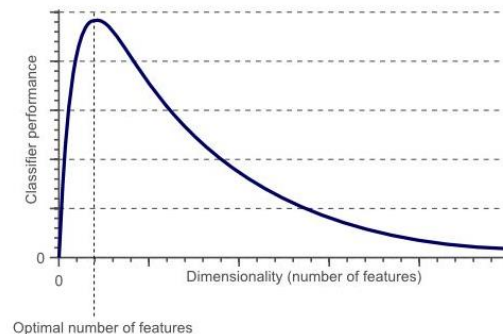
# 问题

---

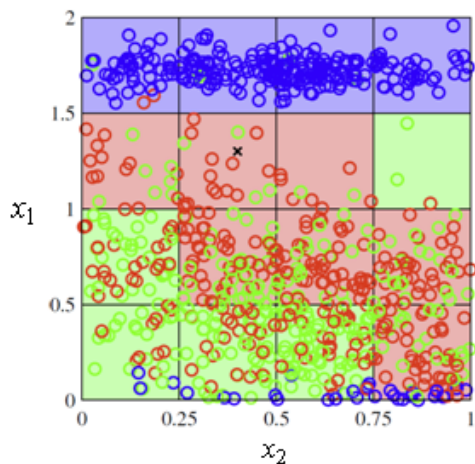
- 欠拟合
- 过拟合
- 维度灾难

# 维度灾难

- 为什么需要降低特征空间的维度？



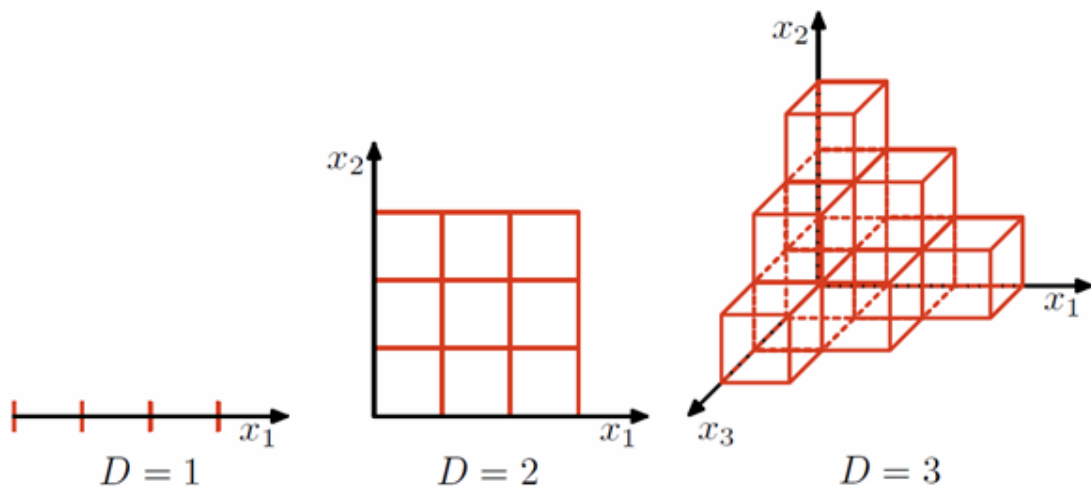
Scatter plot of the training data of three classes. Two features are used. The goal is to classify the new testing point denoted by 'x'.



The feature space is uniformly divided into cells. A cell is labeled as a class, if the majority of training examples in that cell are from that class. The testing point is classified according to the label of the cell where it falls in.

# 维度灾难

- 每个单元 (cell) 中的训练样本越多时，分类器就越鲁棒
- 特征空间中单元的数目随着特征空间维度的增加而指数级增长。如果每个维度被分为3个区间，则单元的总数目为  $N = 3^D$
- 当单元的数目很大时，有些单元是空的



# 解决方案

---

- 如何降低容量
- 如何避免维度灾难



# 如何降低容量

---

- 降低特征的数目
- 降低独立参数的数目
- 降低深度模型的网络规模
- 减少训练的迭代次数
- 向学习器添加正则化约束
- ...

# 降低容量的具体方法

---

- 正则化
- 奥卡姆剃刀

# 正则化 (Regularization)

- 相当于在可以作为学习器解决方案的函数集上施加优先级
- 在贝叶斯学习中，它反映为学习器可以评估的函数空间（相当于它们的参数）的先验概率分布
- 正则化通过对复杂模型添加惩罚项来防止过拟合
- 训练一个分类器/回归器相当于最小化

训练集上的预测误差 + 正则化项

- 示例

— 线性回归的目标函数变为了

$$\text{MSE}_{\text{train}} + \text{regularization} = \frac{1}{N} \sum_i (\mathbf{w}^t \mathbf{x}_i^{(\text{train})} - y_i^{(\text{train})})^2 + \lambda \|\mathbf{w}\|_2^2$$

— 多任务学习，迁移学习，dropout，稀疏性，预训练

# 降低容量的具体方法

---

- 正则化
- 奥卡姆剃刀

# 奥卡姆剃刀 (Occam's Razor)

---

- 机器学习的基本要素是容量和泛化性之间的权衡
- 奥卡姆剃刀指出在能解释训练数据的诸多竞争函数中，人们应该选择更简单的那个。但简单通常是容量的对立面
- 奥卡姆剃刀建议我们选择足够大的函数族，只选择其中的一个来拟合数据

# 解决方案

---

- 如何降低容量
- 如何避免维度灾难

# 如何避免维度灾难

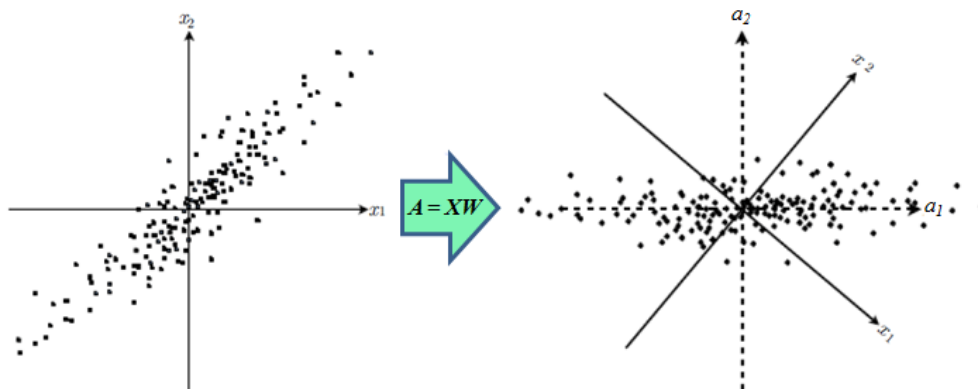
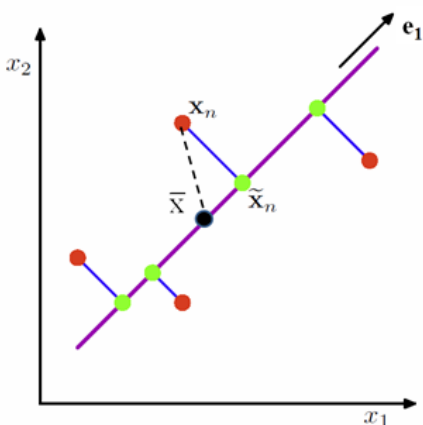
---

- 降低特征的维度
  - 主成分分析法 (PCA) 是尽可能地忠实地再现原始数据的所有信息的降维方法

# 主成分分析法(PCA)

- 有n个d维样本  $\mathbf{x}_1, \dots, \mathbf{x}_n$
- PCA寻找一个由  $d'$  ( $d' < d$ ) 个单位正交向量组  $\mathbf{e}_1, \dots, \mathbf{e}_{d'}$  等所张成的主子空间, 使其
  - 投影到该子空间上的样本 ( $\tilde{\mathbf{x}}_k = a_0 + \sum_{i=1}^{d'} a_{ki} \mathbf{e}_i$ ) 有最大的方差
  - 样本和投影之间的均方差是最小的
  - 投影  $\{a_{ki}\}$  之间是不相关的 (即如果数据分布被假定为高斯分布, 则投影之间相互独立)
- PCA公式化

$$\arg \max_{\{\mathbf{e}_i\}} \sum_{k=1}^n \|\tilde{\mathbf{x}}_k - \bar{\mathbf{x}}\|^2 \quad \text{或} \quad \arg \min_{\{\mathbf{e}_i\}} \sum_{k=1}^n \|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|^2$$
$$(\|\mathbf{x}_k - \bar{\mathbf{x}}\|^2 = \|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|^2 + \|\tilde{\mathbf{x}}_k - \bar{\mathbf{x}}\|^2)$$





# PCA的总结

- 主子空间由 $d'$  个单位正交向量  $\mathbf{e}_1, \dots, \mathbf{e}_{d'}$  所张成, 可以被计算为散布矩阵  $\mathbf{S}$  中对应于最大特征值  $\lambda_1, \dots, \lambda_{d'}$  的 $d'$  个特征向量
- 样本的主成分  $a_{ki}$  可以被计算为
$$a_{ki} = \mathbf{e}_i^t (\mathbf{x}_k - \bar{\mathbf{x}})$$
- 映射到主子空间上的样本方差为  $\sum_{i=1}^{d'} = \lambda_i$
- 样本与样本投影之间的均方距离为  $\sum_{i=d'+1}^d = \lambda_i$
- 假设变量服从高斯分布, 则PCA可以把潜藏于数据中的变量因素分解开
- 我们致力于学习能使更复杂的特征依赖形式分解开的表达

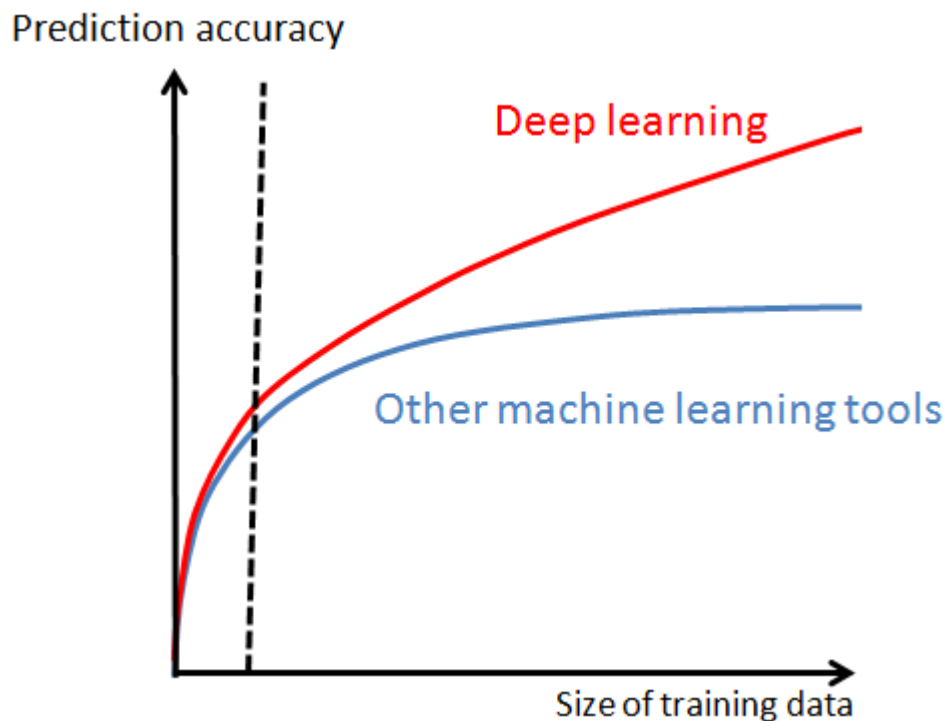
# 机器学习中需要关注的问题

---

- 如何用有效的优化方法和模型来解决欠拟合的问题
- 如何平衡容量和泛化性
- 如何在不增加很多偏差的情况下有效地降低模型容量  
(这意味着减少估计方差)
- 对具有大训练数据的机器学习，如何有效地增加模型容量来覆盖或更接近要估计的真实函数

# 开放性讨论 (Open discussion)

- 与其他大规模训练的机器学习方法相比，为什么深度学习有不同的表现？



# 下一讲

---

- 深度前馈网络

# 致谢

---

本课件中部分材料借鉴以下课程:

- Ian Goodfellow, MIT University, Deep Learning Slides
- Hung-yi Lee, National Taiwan University, Machine Learning and having it Deep and Structured course
- Xiaogang Wang, The Chinese University of Hong Kong, Deep Learning Course
- Fei-Fei Li, Standord University, CS231n Convolutional Neural Networks for Visual Recognition course

---

# Thank You !

