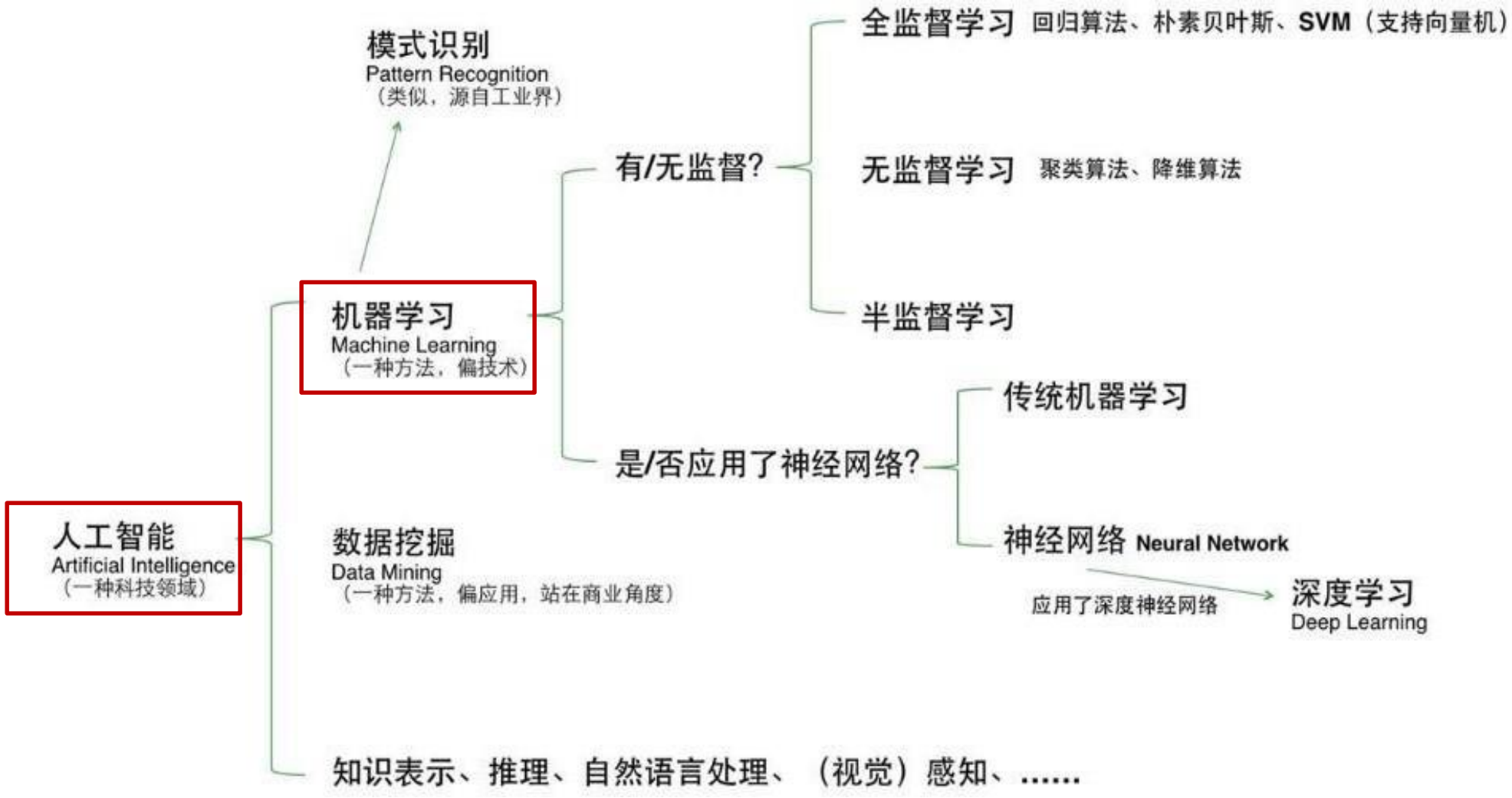
第二讲-机器学习基础

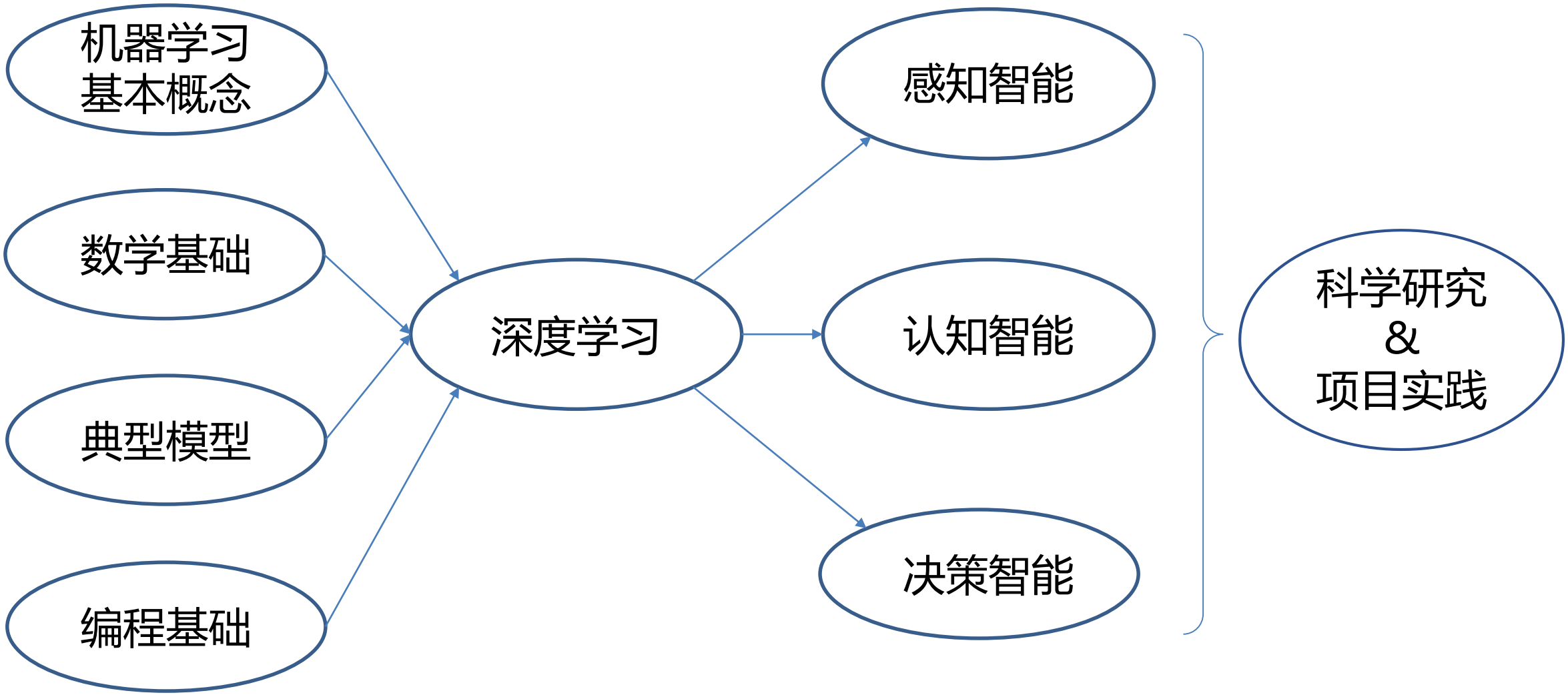
崔志勇

2023-09

人工智能>机器学习>深度学习



机器学习基础 & 数据基础



机器学习基本概念

1. 数学基础
2. 机器学习基础
3. 常用模型
   - 线性回归
   - Logistic回归
   - Softmax回归
   - 感知机

机器学习基本概念

1. 数学基础
2. 机器学习基础
3. 常用模型
   - 线性回归
   - Logistic回归
   - Softmax回归
   - 感知机

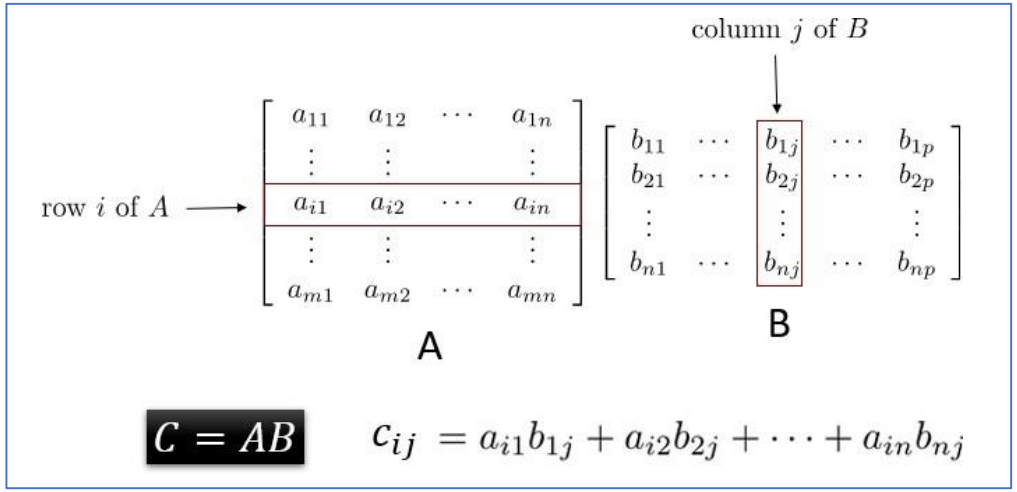
数学基础

- 线性代数
  - 向量、矩阵、矩阵运算
- 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

数学基础 – 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

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



数学基础 – 线性代数

- 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，

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

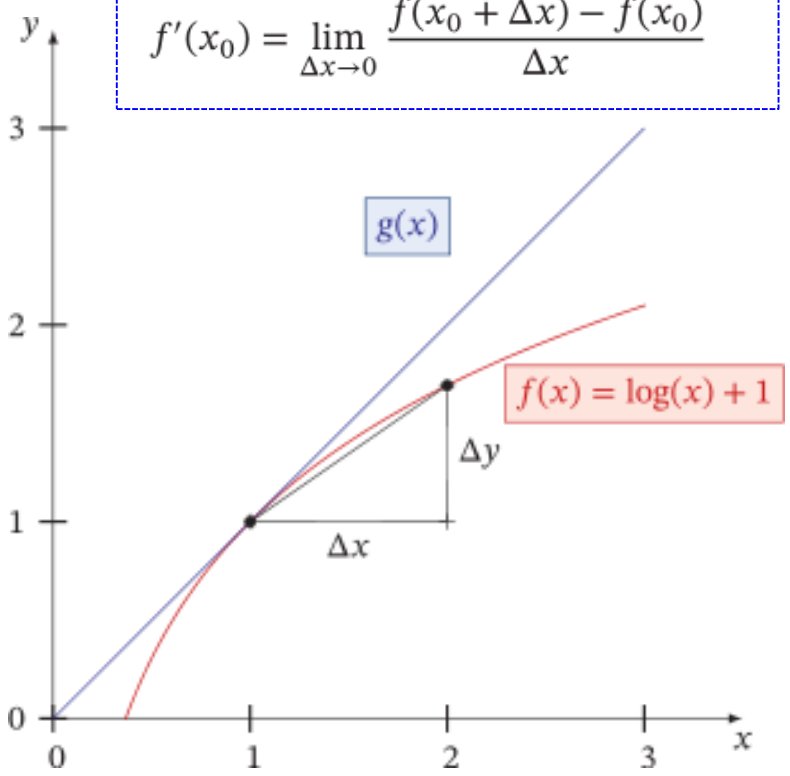
- 矩阵的范数，常用的ℓ_p范数定义

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

数学基础 – 微积分

- **导数**
  - 曲线的斜率，反映曲线变化的快慢

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



- **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

数学基础 – 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

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

数学基础 – 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。
  - 分母布局

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

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

数学基础 – 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

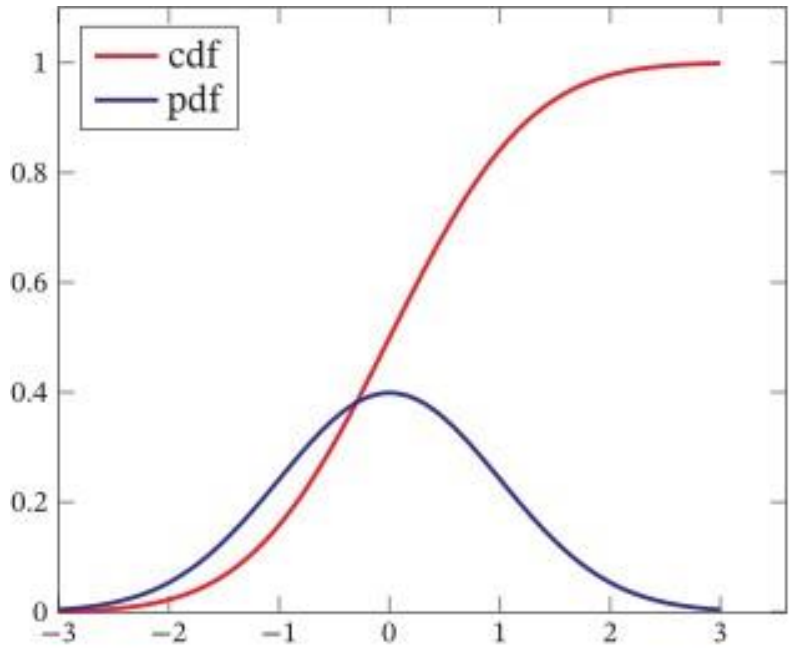
$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \quad \in \mathbb{R}^{M \times N}.$$

（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

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

数学基础 – 概率和统计

- **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
  - 连续随机变量的概率分布有：
    - 均匀分布、正态分布



标准正态分布的概率密度函数和累计分布函数

数学基础 – 概率和统计

- **Jensen不等式**

如果X是随机变量，g是凸函数，则

$$g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)].$$

等式当且仅当 X是一个常数或g是线性时成立

- **大数定理**
- **随机过程**
- **信息熵**

…

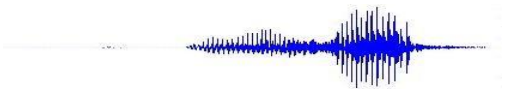
统计学习 ?== 机器学习

机器学习基本概念

1. 数学基础
2. **机器学习基础**
3. 常用模型
   - 线性回归
   - Logistic回归
   - Softmax回归
   - 感知机

机器学习基础

➢ 机器学习 ≈ 构建一个映射函数

- 语音识别 $f($  $)=$ “你好”
- 图像识别 $f($  $)=$ “猫”
- 围棋 $f($  $)=$ “5-5” (落子位置)
- 对话系统 $f($ “你好” $)=$ “今天天气真不错”

用户输入　　机器回应

机器学习的三要素

模型

- 线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$
- 广义线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$
  - 如果 $\phi(\mathrm{x})$ 为可学习的非线性基函数，$f(\mathrm{x},\theta)$ 就等价于神经网络。

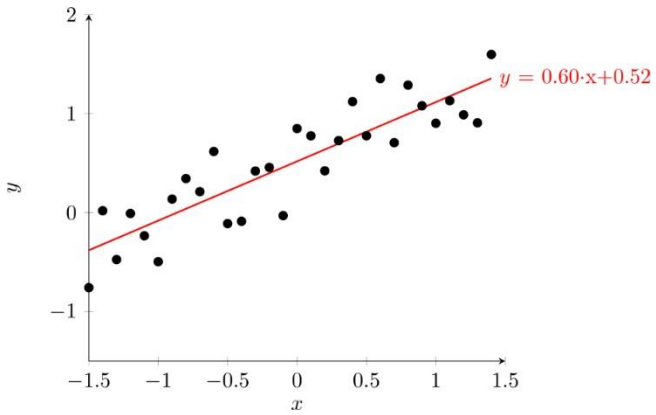
学习准则

- 期望风险

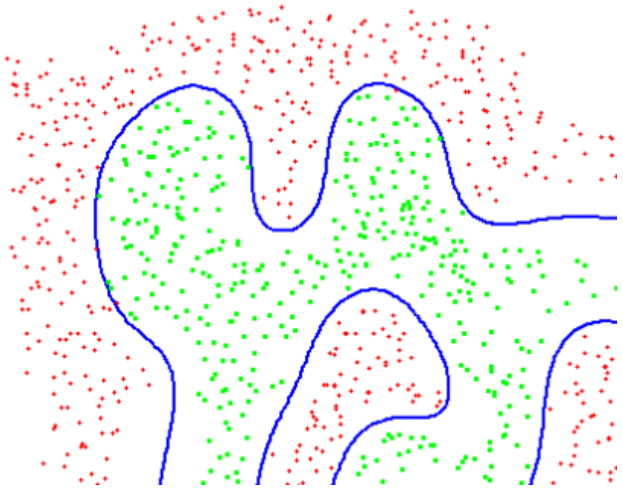
优化算法

- 梯度下降

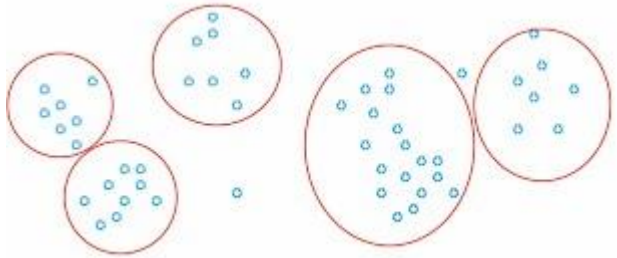
常见的机器学习问题



回归



分类



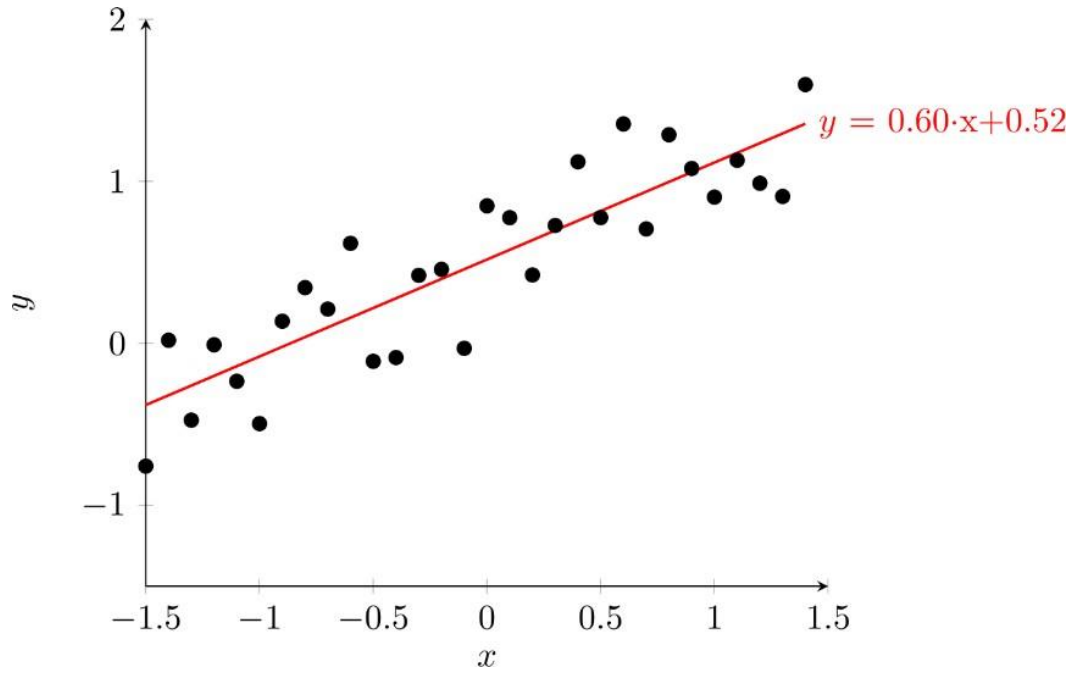
聚类

机器学习的简单示例

- **以线性回归（Linear Regression）为例**

  - 模型：

$$f(\mathbf{x};\mathbf{w},b)=\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$$



如何确定 w？

机器学习的三要素

模型

- 线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$
- 广义线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$
  - 如果 $\phi(\mathrm{x})$ 为可学习的非线性基函数，$f(\mathrm{x},\theta)$ 就等价于神经网络。

x：输入数据
y：输出标签
$f(x,\theta)$：模型预测值
$\theta=\{w^T, b\}$：模型参数

学习准则

- 期望风险 $\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x}), y)],$

⇒ 损失函数

⇒ 经验风险最小化

优化算法

- 梯度下降

机器学习的简单示例

- **以线性回归（Linear Regression）为例**
  - **学习准则**：损失函数
    - 0-1损失函数

$$\mathcal{L}(y, f(\boldsymbol{x};\theta)) = \begin{cases} 0 & \text{if } y = f(\boldsymbol{x};\theta) \\ 1 & \text{if } y \neq f(\boldsymbol{x};\theta) \end{cases} = I(y \neq f(\boldsymbol{x};\theta))$$

    - 平方损失函数

$$\mathcal{L}(y, f(\boldsymbol{x};\theta)) = \frac{1}{2}\left(y - f(\boldsymbol{x};\theta)\right)^2$$

机器学习的简单示例

以线性回归（Linear Regression）为例

- **学习准则**：风险最小化
  - 一个好的模型，应当有一个比较小的期望错误，当真实数据分布和映射函数未知时，通过参数寻找，使得经验风险最小化。
  - 期望风险（expected risk）：对**所有样本（包含未知样本和已知的训练样本）**的预测能力， 是全局概念。
  - 经验风险（empirical risk）：对**所有训练样本**都求一次损失函数，再累加求平均。即模型 f(x)对训练样本中所有样本的预测能力。经验风险则是局部概念，仅仅 表示决策函数对训练数据集里的样本的预测能力。

机器学习的简单示例

以线性回归（Linear Regression）为例

- **学习准则**：风险最小化
  - 期望风险未知，通过经验风险近似
    - 训练数据：$\mathcal{D} = \{\mathrm{x}^{(\mathrm{n})}, y^{(\mathrm{n})}\}, \mathrm{n} \in [1, N]$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$

  - 经验风险最小化
    - 选择合适的风险函数，寻找一个参数 θ^*，使经验风险函数最小化。

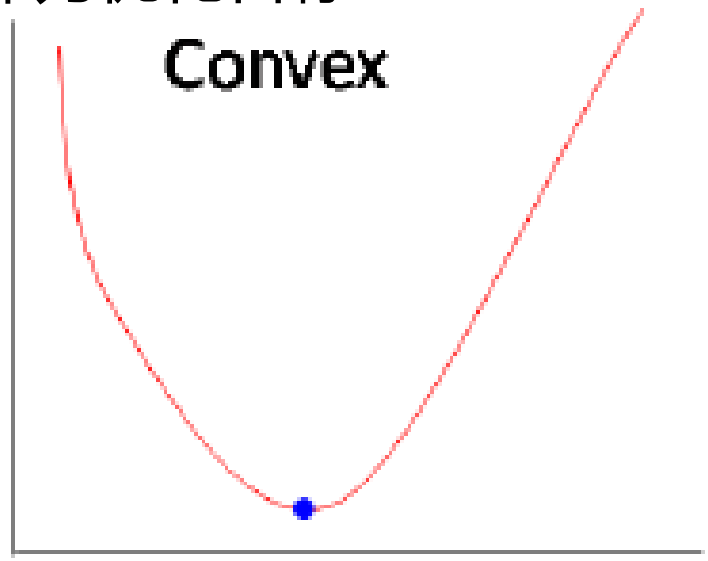
$$\theta^* = \arg\min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

  - **机器学习问题转化成为一个最优化问题**

最优化问题

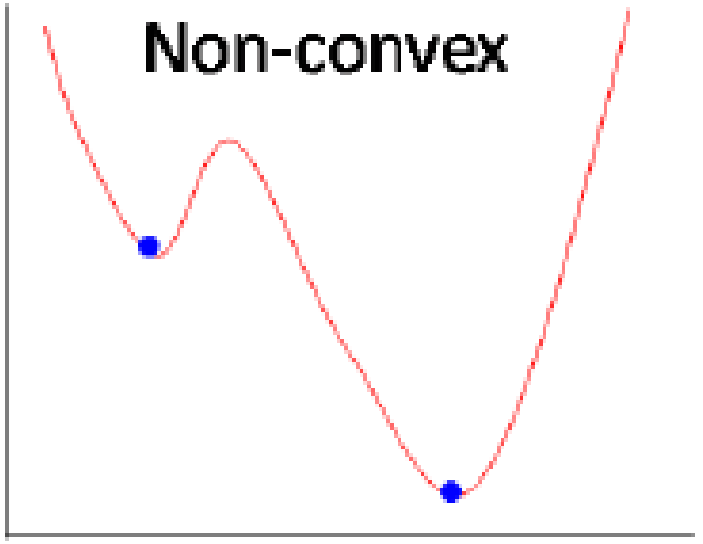
- 一般函数构造凸函数

  作为优化目标



$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$

- 部分模型（例如神经网络）的优化目标是非凸的



局部最优解

机器学习的三要素

模型

- 线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$
- 广义线性方法：$f(\mathbf{x}, \theta) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$
  - 如果 $\phi(\mathrm{x})$ 为可学习的非线性基函数，$f(\mathrm{x},\theta)$ 就等价于神经网络。

学习准则

- 期望风险 $\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x}), y)],$

优化算法

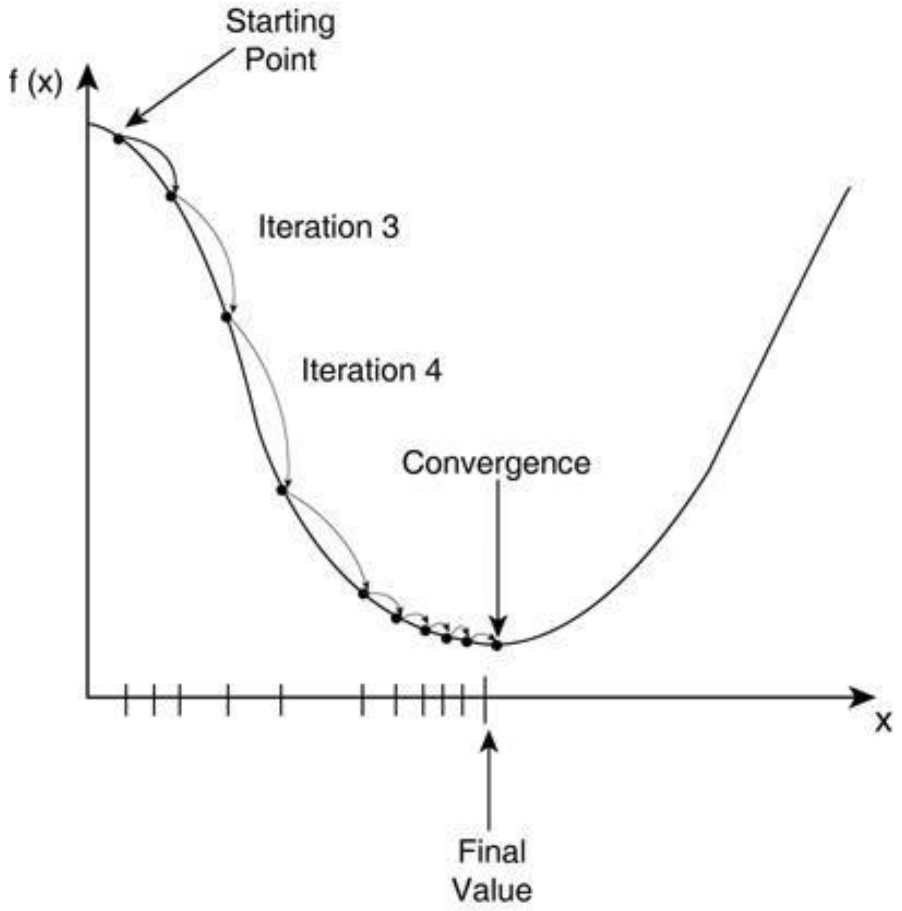
- 梯度下降

梯度下降法（Gradient Descent）

- 假设一个人需要从山的某处开始下山，尽快到达山底。在下山之前他需要确认两件事：下山的方向和下山的距离。因为下山的路有很多，他必须利用一些信息，找到从该处开始最陡峭的方向下山，这样可以保证他尽快到达山底。此外，这座山最陡峭的方向并不是一成不变的，每当走过一段规定的距离，他必须停下来，重新利用现有信息找到新的最陡峭的方向。通过反复进行该过程，最终抵达山底。

- **梯度下降法用于求解无约束最优化问题**：山代表了需要优化的函数表达式；山的最低点就是该函数的最优值；每次下山的距离代表后面要解释的学习率；寻找方向利用的信息即为样本数据；最陡峭的下山方向则与函数表达式**梯度的方向**有关，之所以要寻找最陡峭的方向，是为了满足最快到达山底的限制条件；某处——代表了我们给优化函数设置的初始值，算法后面正是利用这个初始值进行不断的迭代求出最优解。

梯度下降法（Gradient Descent）



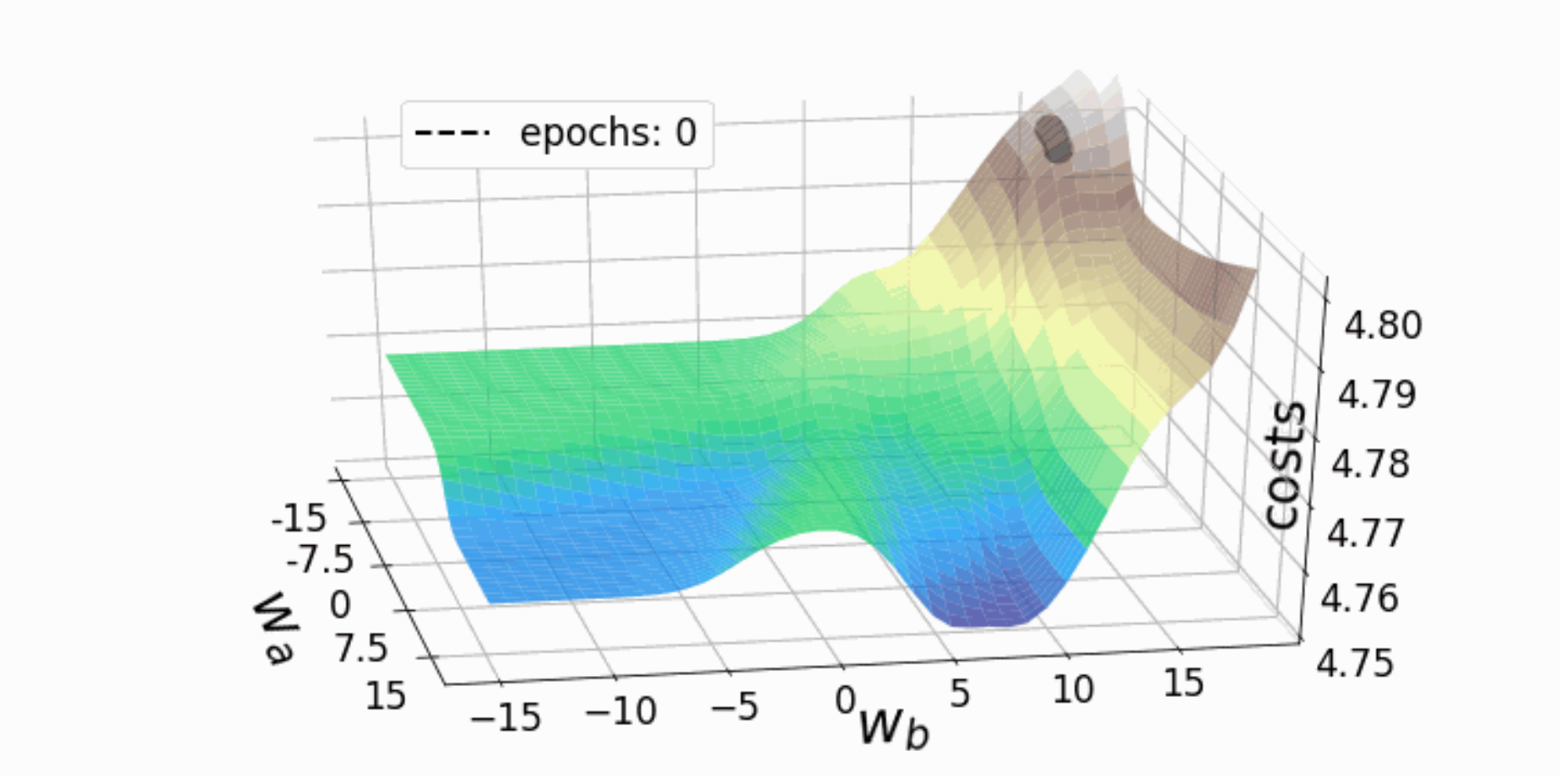
1. 给定待优化连续可微函数 $J(\Theta)$、学习率 α 以及一组初始值 $\Theta_0 = (\theta_{01}, \theta_{02}, \cdots, \theta_{0l},)$
2. 计算待优化函数梯度：$\nabla J(\Theta_0)$
3. 更新迭代公式：$\Theta^{0+1} = \Theta_0 - \alpha \nabla J(\Theta_0)$
4. 计算 Θ^{0+1} 处函数梯度 $\nabla J(\Theta_{0+1})$
5. 计算梯度向量的模来判断算法是否收敛：$\|\nabla J(\Theta)\| \leqslant \varepsilon$
6. 若收敛，算法停止，否则根据迭代公式继续迭代

- 经过迭代计算风险函数的最小值

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

搜索步长 α 中也叫作学习率（Learning Rate）

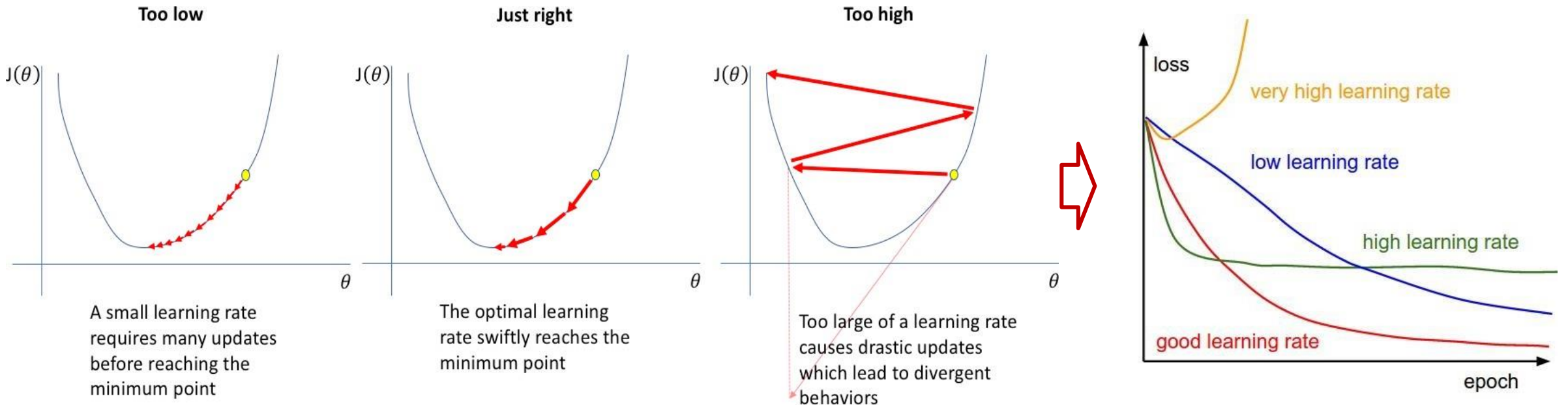
梯度下降法（Gradient Descent）



https://towardsdatascience.com/animations-of-gradient-descent-and-loss-landscapes-of-neural-networks-in-python-e757f3584057

梯度下降法（Gradient Descent）

- 学习率是十分重要的超参数!



随机梯度下降法（Stochastic Gradient Descent，SGD）

（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。

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

速度慢！大数据内存不足！

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta},$$

方差大！损失函数震荡严重！

算法 2.1: 随机梯度下降法

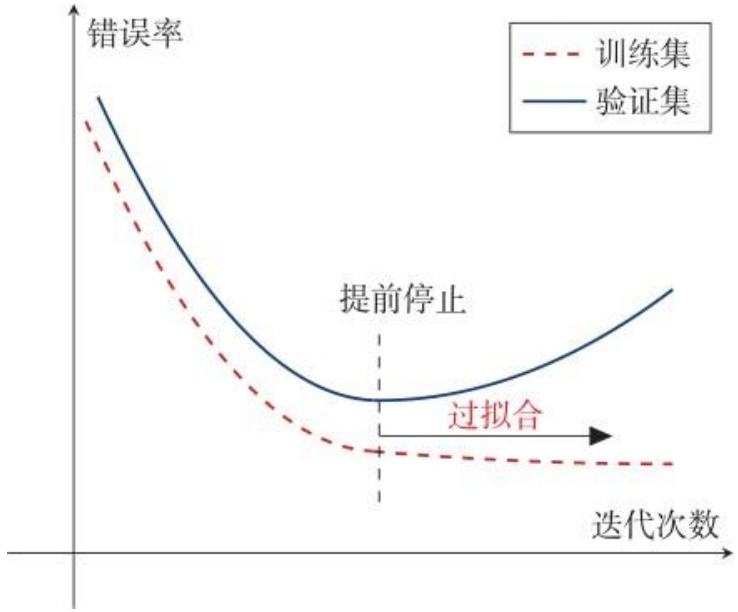
输入: 训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

随机初始化 θ;
repeat
  对训练集 $\mathcal{D}$ 中的样本随机重排序;
  for $n = 1 \cdots N$ **do**
    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)}, y^{(n)})$;
    // 更新参数
    $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta; x^{(n)}, y^{(n)})}{\partial \theta}$;
  end
until 模型 $f(\mathbf{x}; \theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;
输出: θ

其他优化方法

- **小批量（Mini-Batch）随机梯度下降法**
  - 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。

- **提前停止法**
  - 验证集上错误率不再下降，就停止迭代



多选题 1分

设置

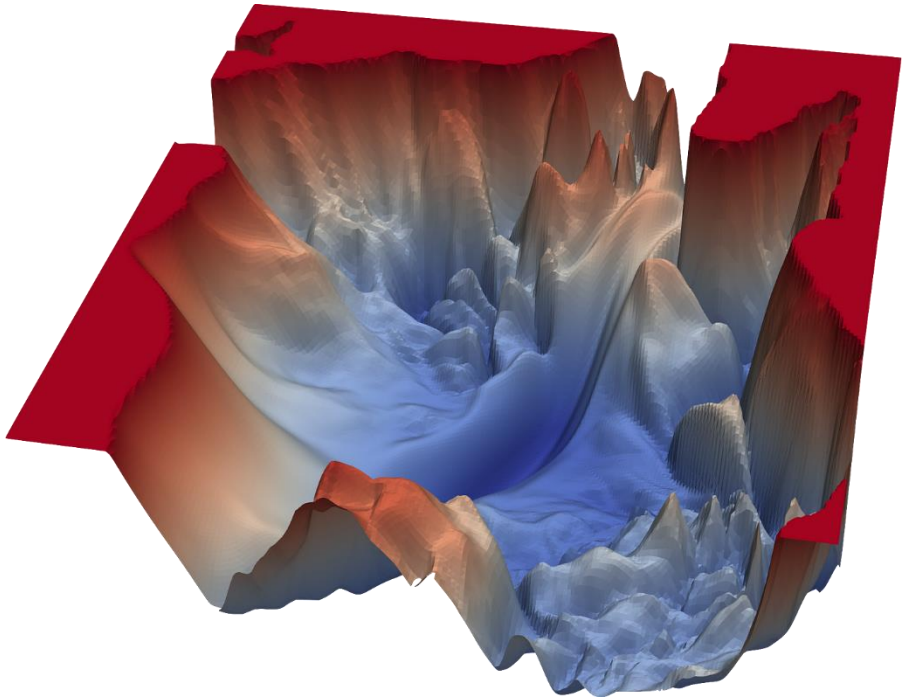
右图是一个一个56层卷积神经网络(VGG-56)的损失函数图景，
下列因素影响模型（神经网络）是否能够收敛到到最优的是 （）

A 学习率

B 优化算法

C 模型初始化参数

D 模型训练轮次



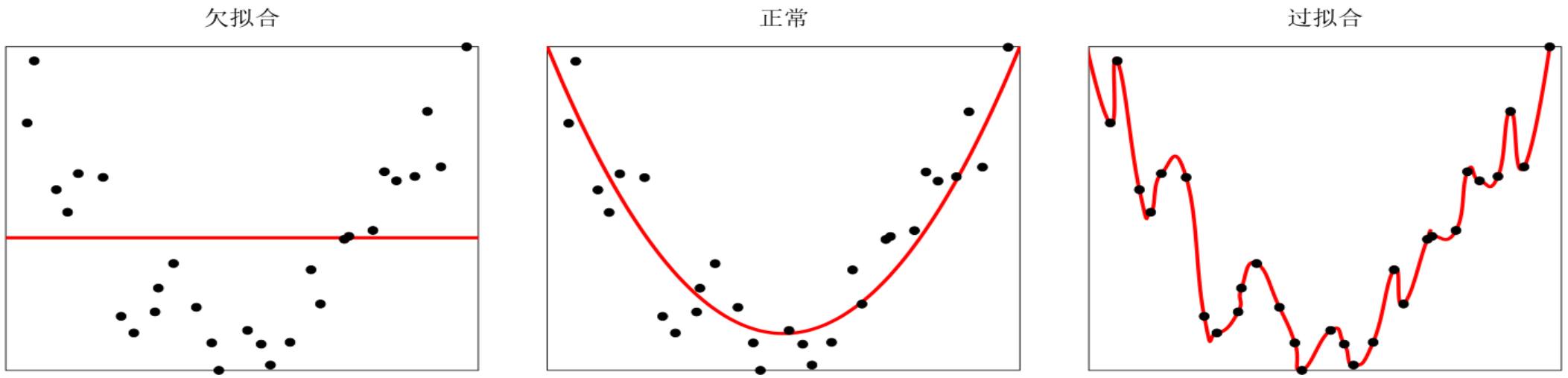
一个56层卷积神经网络(VGG-56)的
损失函数图景

提交

https://arxiv.org/abs/1712.09913

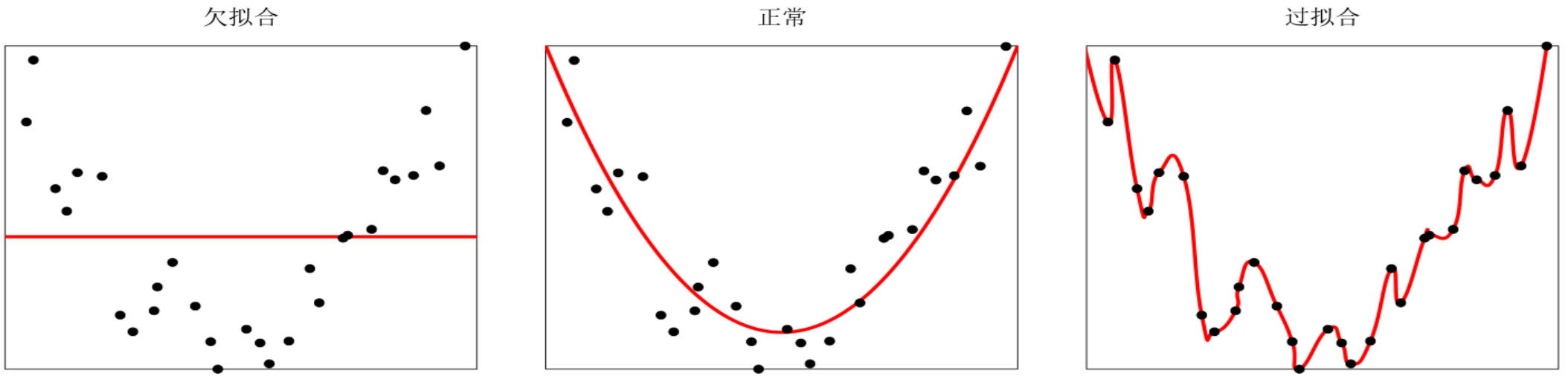
过拟合和欠拟合

- **欠拟合（Underfitting）：**
  - 即模型不能很好地拟合训练数据， 在训练集上的错误率比较高。
  - 欠拟合一般是由于**模型能力不足**造成的，说明其对训练样本的一般性质尚未学好。



过拟合和欠拟合

- **过拟合（Overfitting）：**
  - 学习器把训练样本学习得“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。
  - 过拟合问题往往是**由于训练数据少和噪声以及模型能力强**等原因造成的。



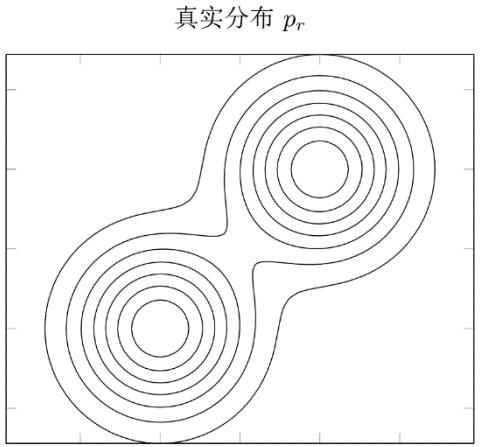
泛化错误

期望风险

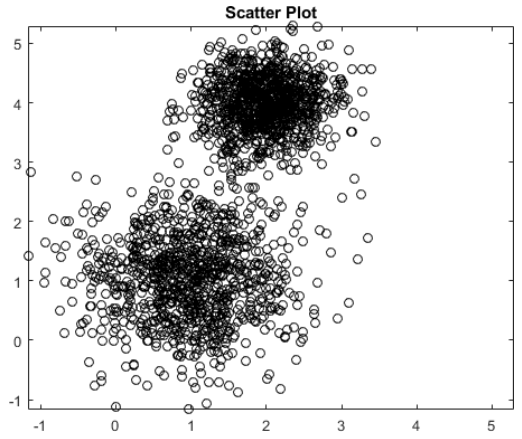
$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x}), y)],$$

经验风险

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$



≠



⇨ **正则化**

限制模型能力，
使其不要过度地
最小化经验风险

泛化错误 $\mathcal{G}_{\mathcal{D}}(f) = \mathcal{R}(f) - \mathcal{R}_{\mathcal{D}}^{emp}(f)$

- ✓ 泛化错误可以衡量一个机器学习模型是否可以很好地泛化到未知数据。
- ✓ 泛化错误一般表现为一个模型在训练集和测试集上的错误率。
- ✓ 机器学习的目标是减少泛化错误。

正则化（regularization）

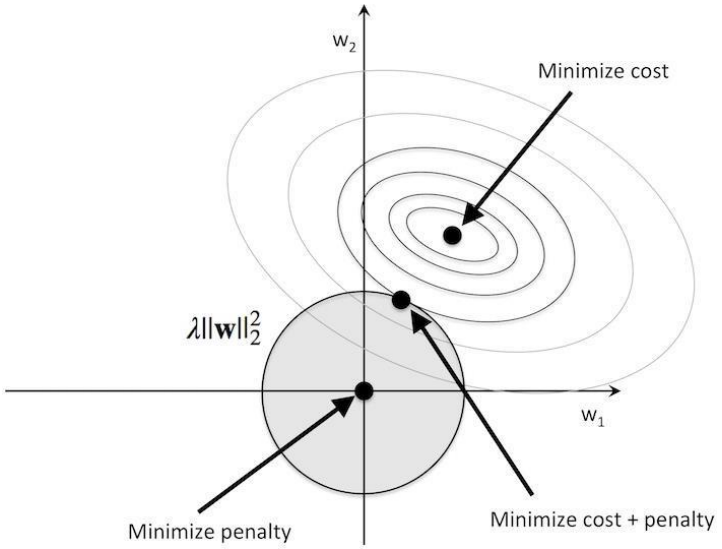
所有损害优化的方法都是正则化

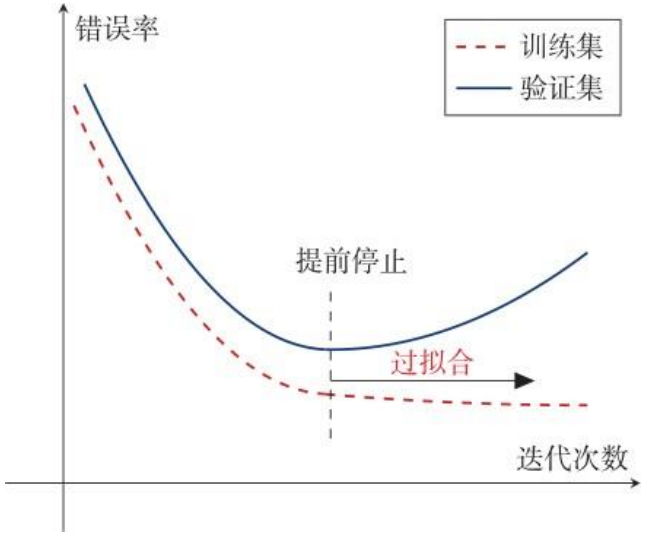
增加优化约束

L1/L2约束、数据增强

干扰优化过程

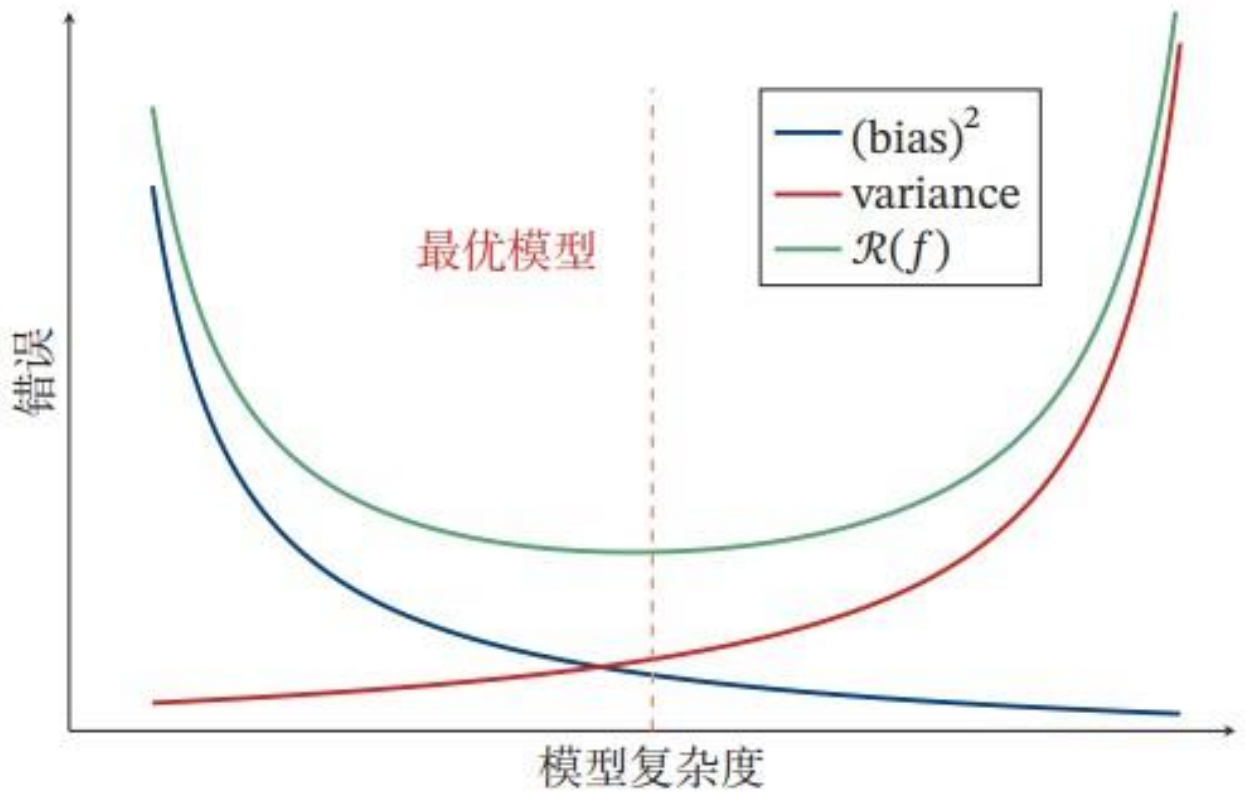
权重衰减、随机梯度下降、提前停止

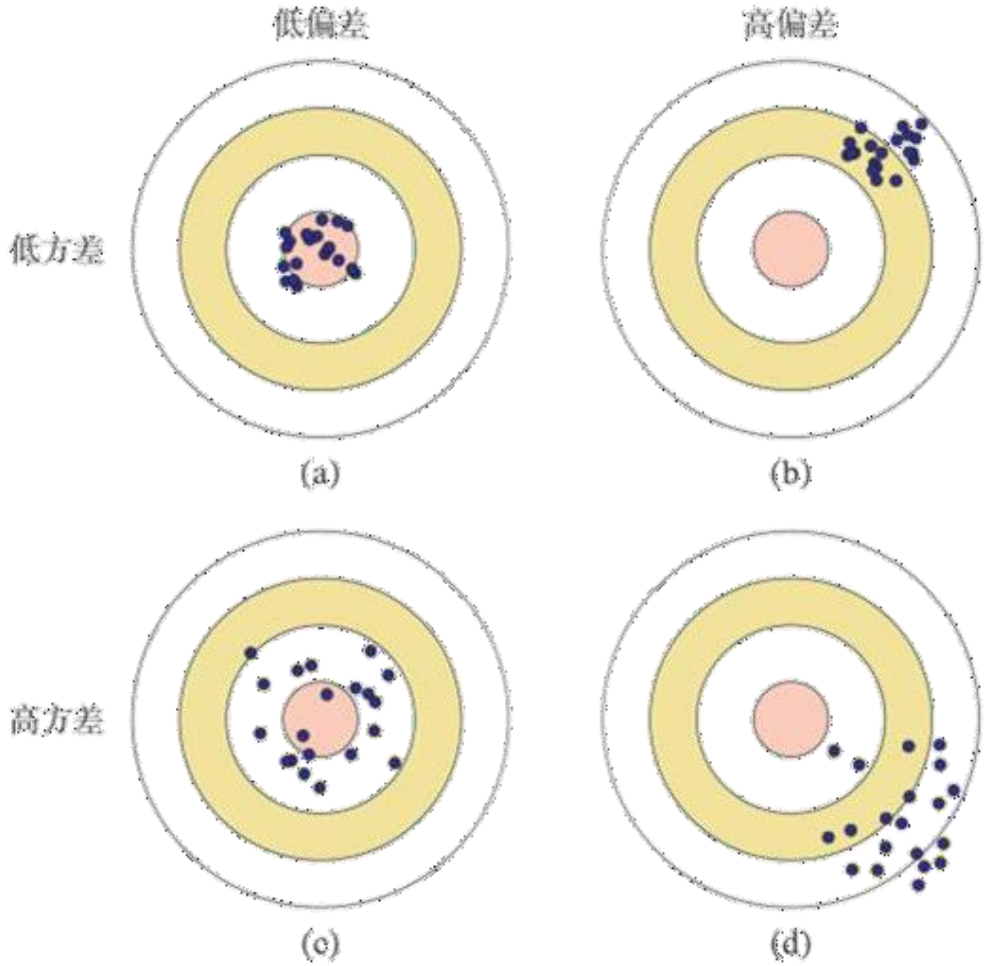




偏差-方差分解（Bias-Variance Decomposition）

- 最小化期望错误等价于最小化偏差和方差之和





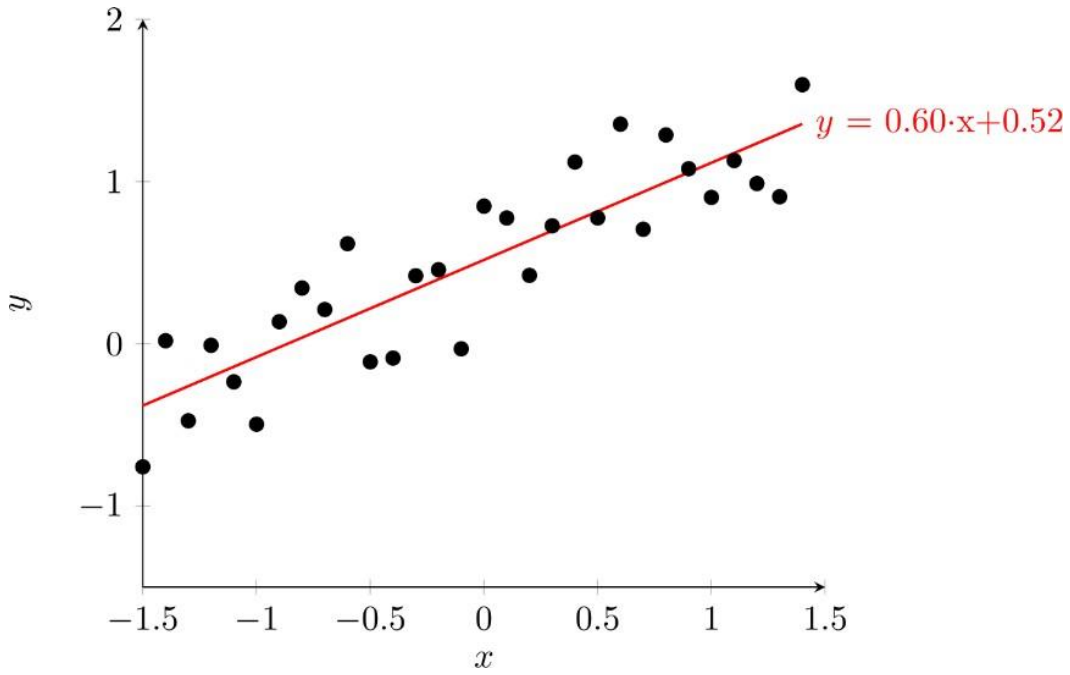
机器学习基本概念

1. 数学基础
2. 机器学习基础
3. **常用模型**
   - 线性回归
   - Logistic回归
   - Softmax回归
   - 感知机

线性模型（Linear Model）

- 线性模型是通过样本特征的线性组合来进行预测的模型

$$f(\mathbf{x};\mathbf{w},b)=\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$$



- 分类问题中，由于输出目标 y 是一些离散的标签，而 f 值域为实数，因此无法直接进行预测，需要引入一个非线性的决策函数（Decision Function）$g(\cdot)$来预测输出目标

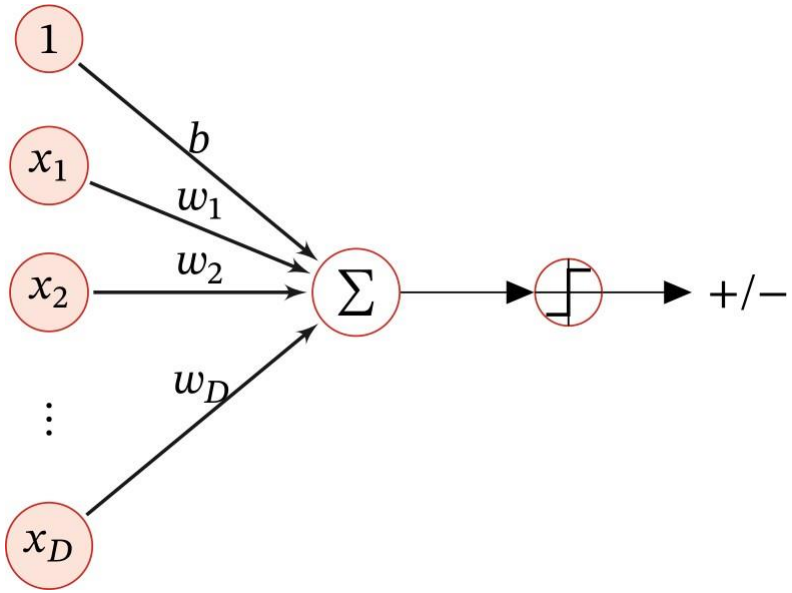
$$y=g(f(\boldsymbol{x};\boldsymbol{w})),$$

线性模型（Linear Model）

二分类问题中，$g(\cdot)$ 可以是符号函数（Sign Function）

$$\mathrm{g}\big(f(\boldsymbol{x};\boldsymbol{w})\big) = \mathrm{sgn}\big(f(\boldsymbol{x};\boldsymbol{w})\big)$$

$$\triangleq \begin{cases} +1 & \text{if} \quad f(\boldsymbol{x};\boldsymbol{w}) > 0, \\ -1 & \text{if} \quad f(\boldsymbol{x};\boldsymbol{w}) < 0. \end{cases}$$



损失函数？

Logistic 回归

➢将分类问题看作条件概率估计问题

- 为了解决连续的线性函数不适合进行分类的问题，引入非线性函数 g 来预测类别标签的条件概率 $p(y=c|x)$。
- 以二分类为例，

$$p(y=1|\mathbf{x})=g(f(\mathbf{x};\mathbf{w}))$$

- 函数 f：线性函数
- 函数 g：把线性函数的值域从实数区间“挤压”到了(0,1)之间，可以用来表示概率。

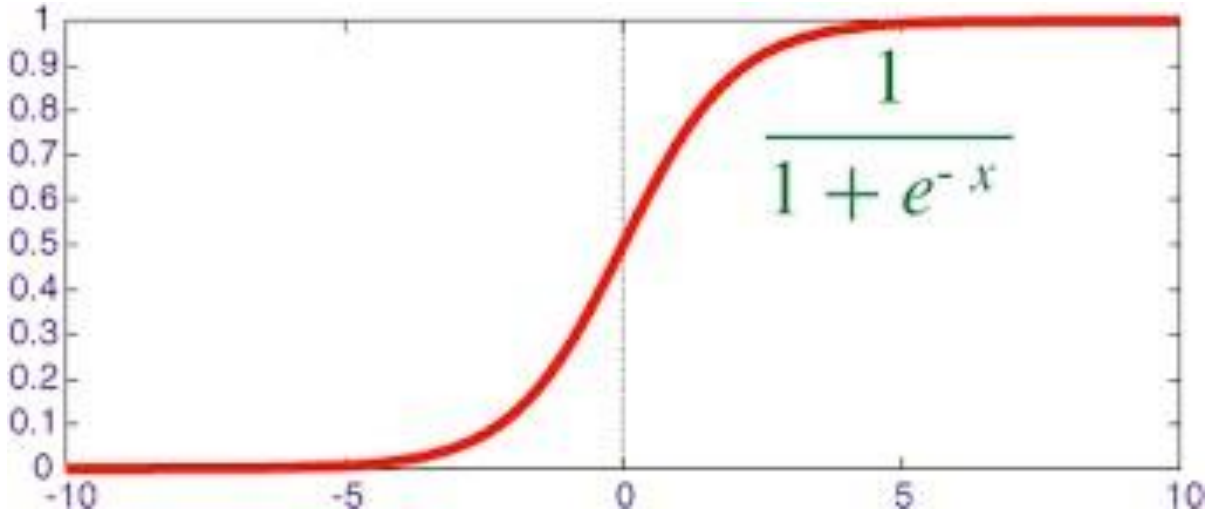
如何构建函数 g？

Logistic 回归

➢Logistic 函数

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

除了logisitc，还有什么函数$f: R \to (0,1)$?



➢Logistic 回归

$$p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\top}\boldsymbol{x})$$

$$\triangleq \frac{1}{1 + \exp(-\boldsymbol{w}^{\top}\boldsymbol{x})}$$

Logistic 回归

➢学习准则

- 模型预测条件概率$p_\theta(y|\boldsymbol{x})$

$$p_\theta(y=1|\boldsymbol{x})=\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

- 真实条件概率 $p_r(y|\boldsymbol{x})$
  - 对于一个样本(x,y^*)，其真实条件概率为

$$p_r(y=1|\boldsymbol{x})=y^*$$
$$p_r(y=0|\boldsymbol{x})=1-y^*$$

如何衡量两个条件分布的差异?

Logistic 回归

➢学习准则 -- 熵（Entropy）

- 在信息论中，熵用来衡量一个随机事件的不确定性。
  - 自信息（Self Information）　　$I(x) = -\log(p(x))$
  - 熵

$$
\begin{aligned}
H(X) &= \mathbb{E}_X[I(x)] \\
&= \mathbb{E}_X[-\log p(x)] \\
&= -\sum_{x \in X} p(x) \log p(x)
\end{aligned}
$$

  - 熵越高，则随机变量的信息越多；熵越低，则随机变量的信息越少 。
  - 在对分布 $q(y)$ 的符号进行编码时，熵 $I(q)$ 也是理论上最优的平均编码长度，这种编码方式称为熵编码（ Entropy Encoding ） 。

Logistic 回归

➢**学习准则 --** 交叉熵（Cross Entropy）

- 交叉熵是按照概率分布 q 的最优编码对真实分布为 p 的信息进行编码的长度。

$$H(p,q) = \mathbb{E}_p[-\log q(x)]$$
$$= -\sum_x p(x) \log q(x)$$

- 在给定q 的情况下，如果 p 和 q 越接近，交叉熵越小。
- 如果 p 和 q 越远，交叉熵就越大。

Logistic 回归

➢参数学习：**Logistic回归采用交叉熵作为损失函数，并使用梯度下 降法来对参数进行优化**

给定N 个训练样本$\{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$， 用Logistic回归模型对每个样本$\boldsymbol{x}^{(n)}$ 进行预测，输出其标签为1的后验概率， 记为模型预测条件概率$\hat{y}^{(n)}$

$$\hat{y}^{(n)} = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}^{(n)}), \qquad 1 \leq n \leq N.$$

由于$y^{(n)} \in \{0, 1\}$， 样本$(\boldsymbol{x}^{(n)}, y^{(n)})$的真实条件概率可以表示为:

$$p_r(y^{(n)} = 1|\boldsymbol{x}^{(n)}) = y^{(n)},$$

$$p_r(y^{(n)} = 0|\boldsymbol{x}^{(n)}) = 1 - y^{(n)}.$$

Logistic 回归

- 使用交叉熵损失函数，模型在经验风险函数为

$$\mathcal{R}(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left(p_r(y^{(n)}=1|\boldsymbol{x}^{(n)})\log\hat{y}^{(n)} + p_r(y^{(n)}=0|\boldsymbol{x}^{(n)})\log(1-\hat{y}^{(n)})\right)$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\log\hat{y}^{(n)} + (1-y^{(n)})\log(1-\hat{y}^{(n)})\right).$$

$$\hat{y}^{(n)} = \sigma(\boldsymbol{w}^{\top}\boldsymbol{x}^{(n)}), \qquad 1 \le n \le N.$$

- 梯度为

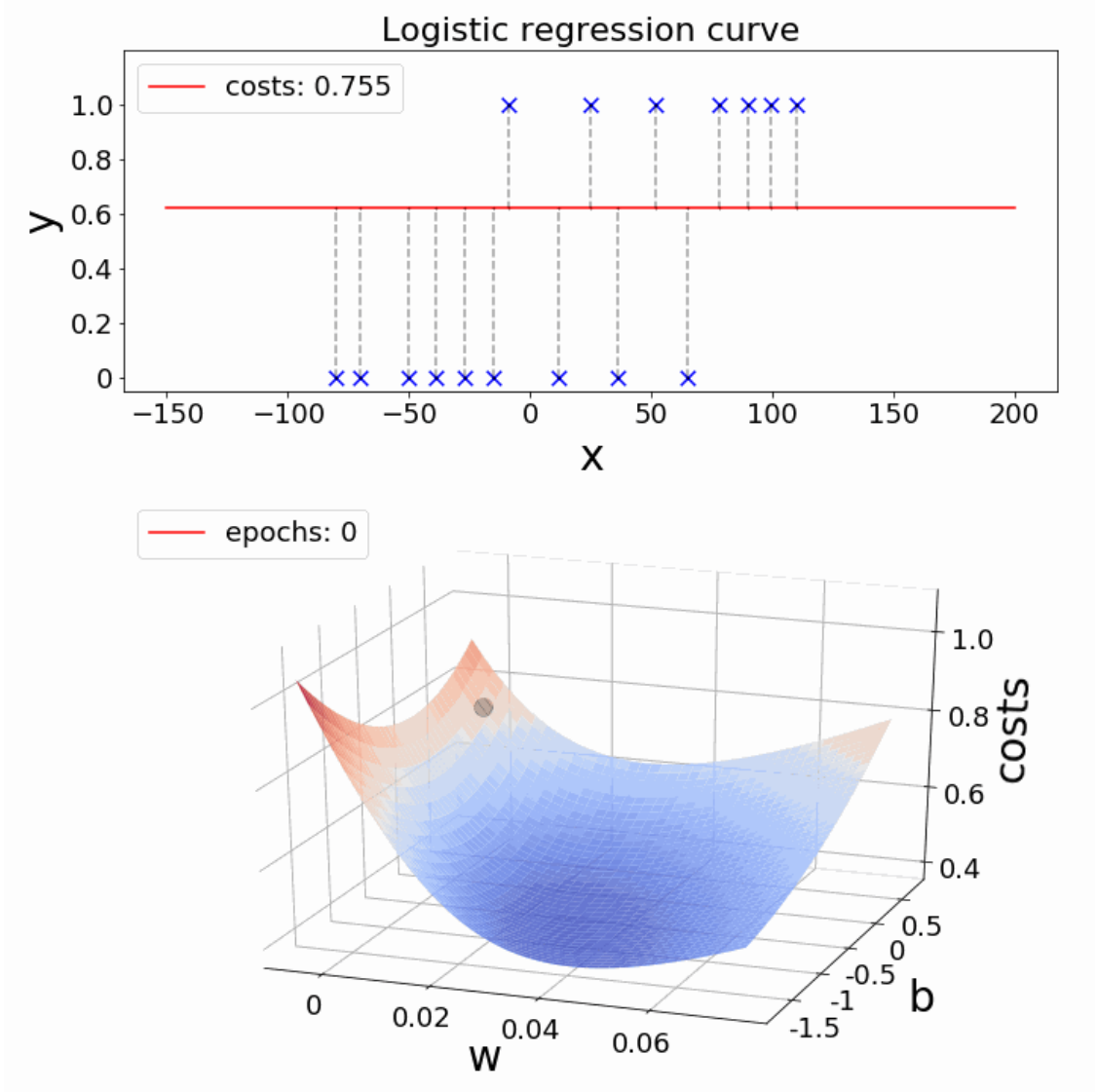
$$\frac{\partial\mathcal{R}(\boldsymbol{w})}{\partial\boldsymbol{w}} = -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\frac{\hat{y}^{(n)}(1-\hat{y}^{(n)})}{\hat{y}^{(n)}}\boldsymbol{x}^{(n)} - (1-y^{(n)})\frac{\hat{y}^{(n)}(1-\hat{y}^{(n)})}{1-\hat{y}^{(n)}}\boldsymbol{x}^{(n)}\right)$$

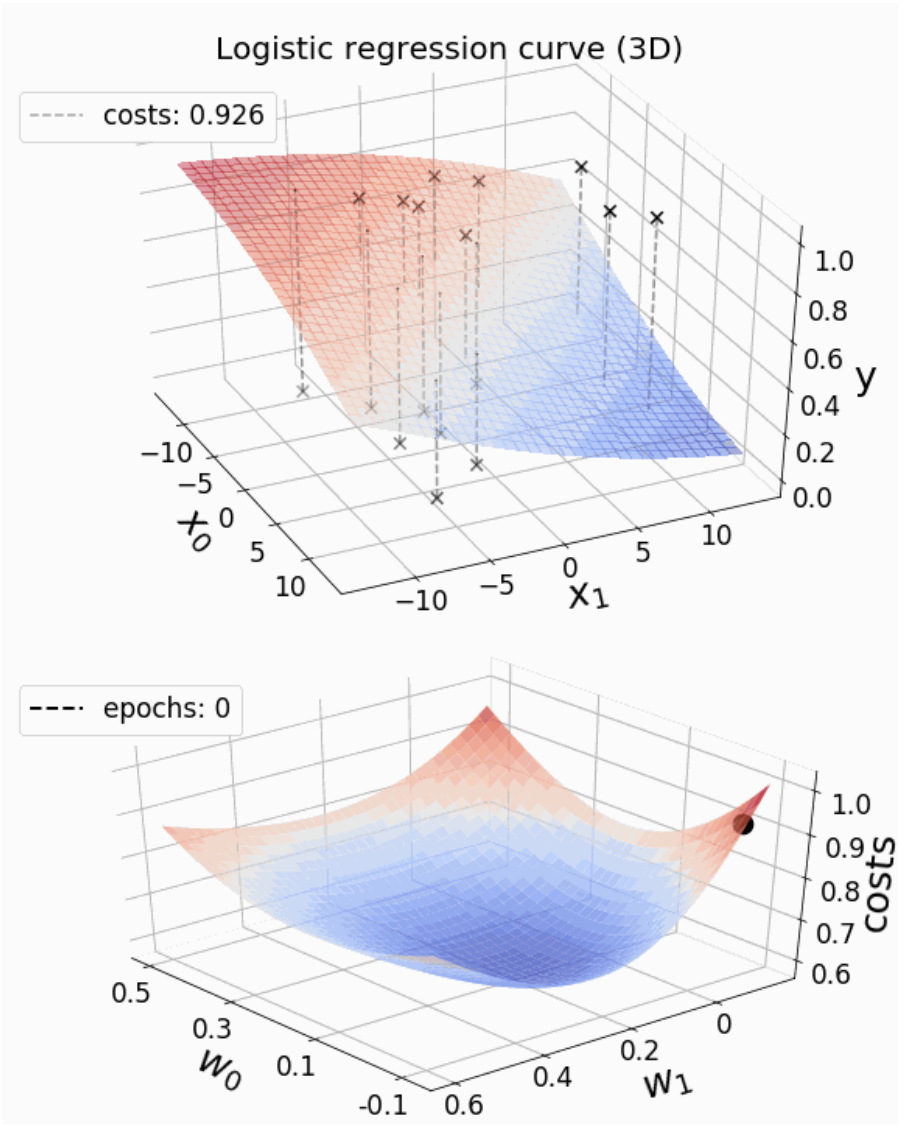
$$= -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}(1-\hat{y}^{(n)})\boldsymbol{x}^{(n)} - (1-y^{(n)})\hat{y}^{(n)}\boldsymbol{x}^{(n)}\right)$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}(y^{(n)}-\hat{y}^{(n)}).$$

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}\left(y^{(n)} - \hat{y}_{\boldsymbol{w}_t}^{(n)}\right),$$

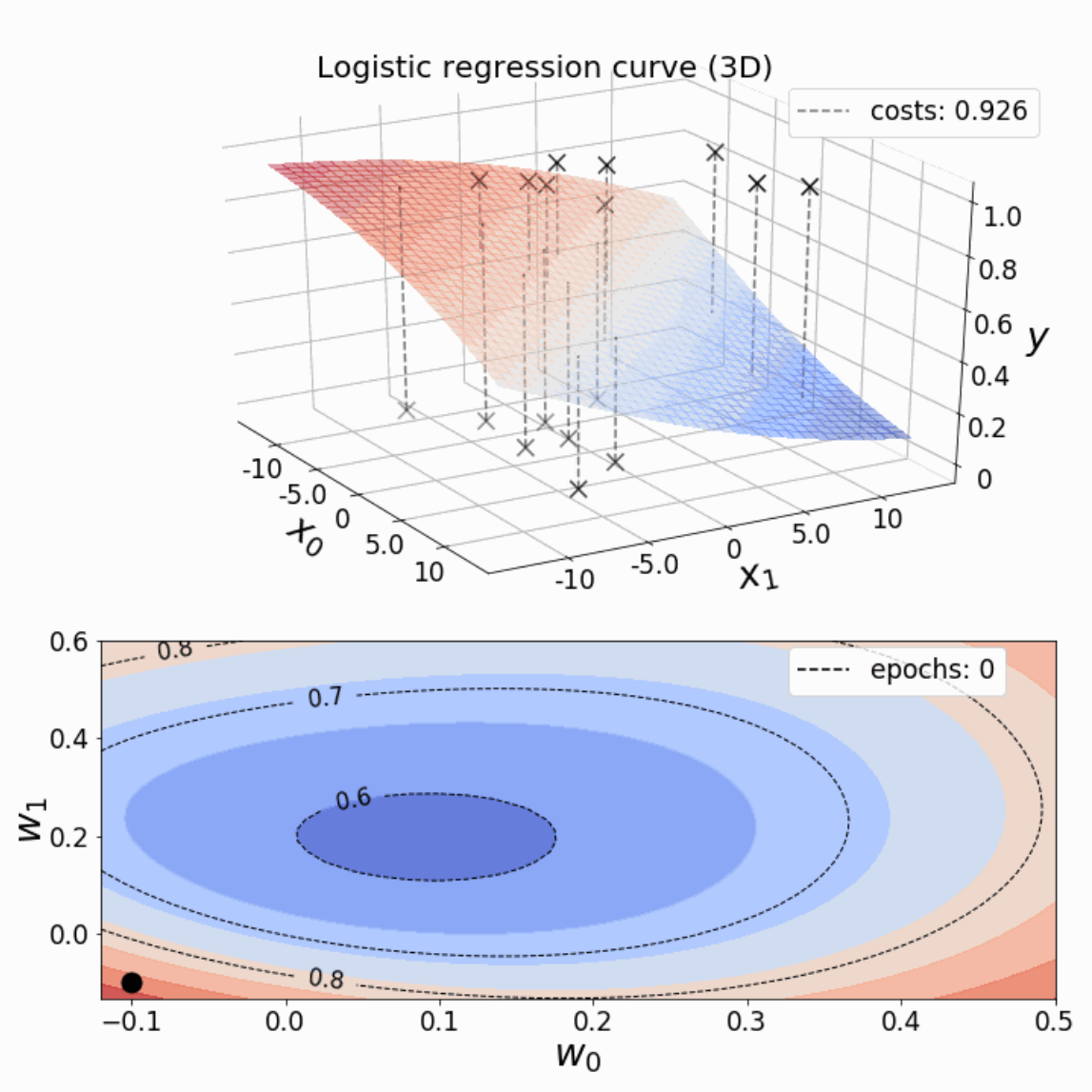
Logistic 回归





https://towardsdatascience.com/animations-of-logistic-regression-with-python-31f8c9cb420

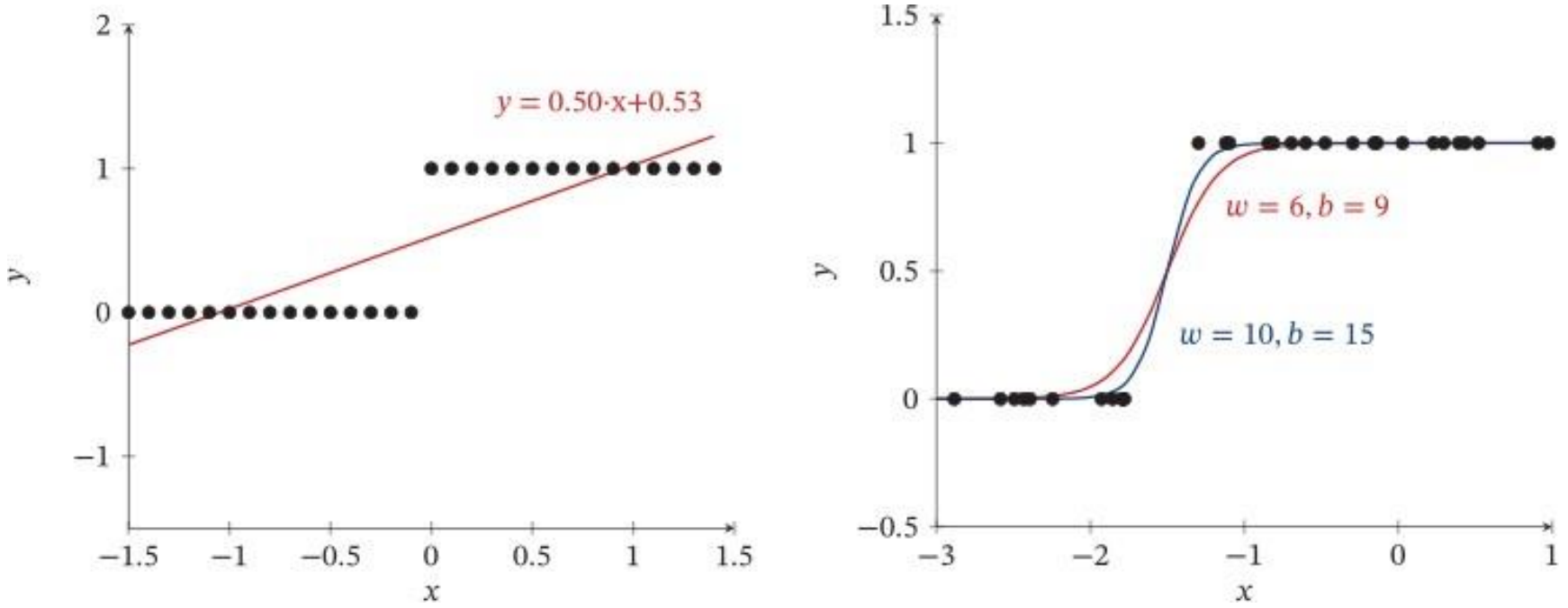
Logistic 回归



https://towardsdatascience.com/animations-of-logistic-regression-with-python-31f8c9cb420

Logistic 回归

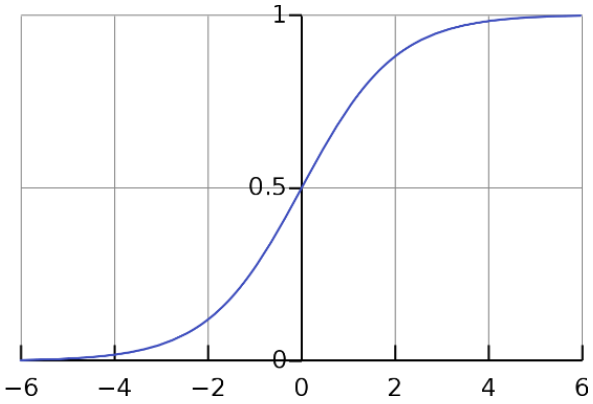
➢使用线性回归和Logistic回归来解决一维数据的二分类问题的示例。



Logistic 回归-小结

- **模型**

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \triangleq \frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}$$



- **学习准则：**交叉熵

$$\mathcal{R}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\log\hat{y}^{(n)} + (1-y^{(n)})\log(1-\hat{y}^{(n)})\right)$$

- **优化算法：**梯度下降

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}\left(y^{(n)} - \hat{y}_{\boldsymbol{w}_t}^{(n)}\right),$$

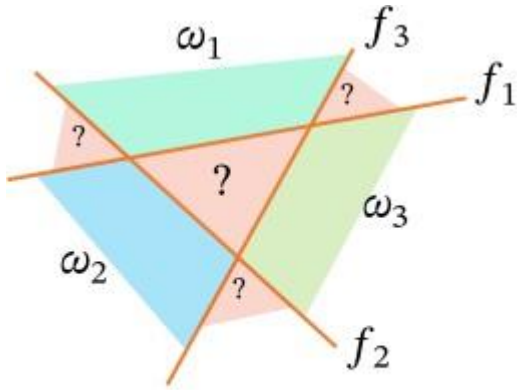
多分类问题（Multi-class Classification）

多分类问题是指分类的类别数大于2。 多分类一般需要多个线性判别函数， 但设计这些判别函数有很多种方式。

假设一个多分类问题的类别为$\{1, 2, \cdots, C\}$， 常用的方式有以下三种：

- **“一对其余” 方式**： 把多分类问题转换为 C 个 “一对其余” 的二分类问题。

  这种方式共需要 C 个判别函数， 其中第 c 个判别函数 f_c 是将类别 c 的样本和不属于类别 c 的样本分开。



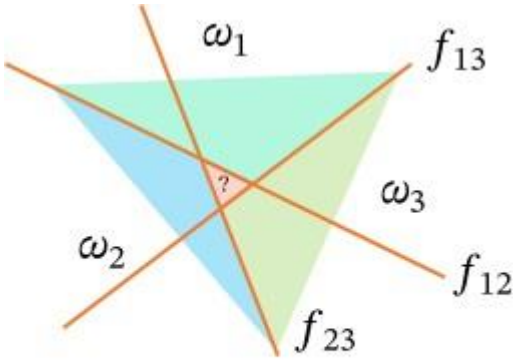
(a) “一对其余”方式

多分类问题（Multi-class Classification）

多分类问题是指分类的类别数大于2。 多分类一般需要多个线性判别函数， 但设计这些判别函数有很多种方式。

假设一个多分类问题的类别为$\{1, 2, \cdots, C\}$， 常用的方式有以下三种：

- **“一对一” 方式**： 把多分类问题转换为 $C(C-1)/2$个 “一对一” 的二分类问题。 这种方式共需要 $C(C-1)/2$个判别函数， 其中第(i, j) 个判别函数是把类别 i 和类别 j 的样本分开。



(b) “一对一”方式

“一对其余”方式和 “一对一” 方式都存在一个缺陷：特征 空间中会存在一些难以确定 类别的区域。

多分类问题（Multi-class Classification）

多分类问题是指分类的类别数大于2。 多分类一般需要多个线性判别函数， 但设计这些判别函数有很多种方式。

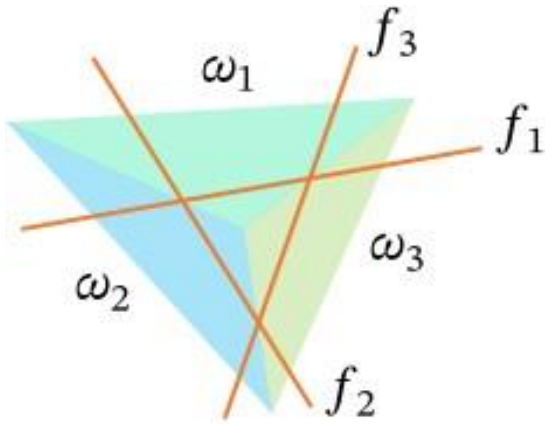
假设一个多分类问题的类别为$\{1, 2, \cdots, C\}$， 常用的方式有以下三种：

- **“argmax”方式**：这是一种改进的“一对其余”方式，共需要C个判别函数

$$f_c(\boldsymbol{x}; \boldsymbol{w}_c) = \boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x} + b_c, \qquad c \in \{1, \cdots, C\}$$

对于样本$\boldsymbol{x}$，如果存在一个类别c，相对于所有的其他类别$c\text{‖}(c\text{‖} \neq c)$有$f_c(\boldsymbol{x}; \boldsymbol{w}_c) > f_{c\text{‖}}(\boldsymbol{x}, \boldsymbol{w}_{c\text{‖}})$，那么$\boldsymbol{x}$属于类别$c$。“argmax”方式的预测函数为：

$$y = \arg\max_{c=1}^{C} f_c(\boldsymbol{x}; \boldsymbol{w}_c)$$



(c) “argmax”方式

多分类问题（Multi-class Classification）

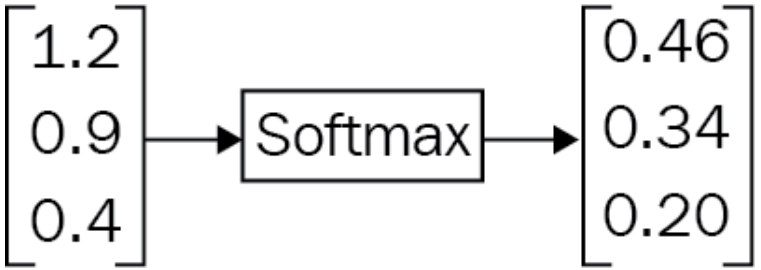
定义 3.2 – 多类线性可分：对于训练集 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，如果存在 C 个权重向量 $\boldsymbol{w}_1^*, \cdots, \boldsymbol{w}_C^*$，使得第 $c(1 \leq c \leq C)$ 类的所有样本都满足 $f_c(\boldsymbol{x}; \boldsymbol{w}_c^*) > f_{\tilde{c}}(\boldsymbol{x}, \boldsymbol{w}_{\tilde{c}}^*), \forall \tilde{c} \neq c$，那么训练集 $\mathcal{D}$ 是线性可分的.

从上面定义可知， 如果数据集是多类线性可分的， 那么一定存在一个“argmax”方式的线性分类器可以将它们正确分开。

Softmax回归

Softmax 回归（Softmax Regression），也称为多项（Multinomial）或多类（Multi-Class）的Logistic回归

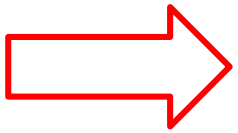
$$\text{softmax}(x_k) = \frac{\exp(x_k)}{\sum_{i=1}^{K} \exp(x_i)}$$



Softmax回归

- 给定一个样本$\boldsymbol{x}$，Softmax 回归预测的属于类别c的条件概率为：

$$p(y=c|\boldsymbol{x}) = \text{softmax}(\boldsymbol{w}_c^{\mathsf{T}}\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c'=1}^{C}\exp(\boldsymbol{w}_{c'}^{\mathsf{T}}\boldsymbol{x})},$$



$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{x}) = \frac{\exp(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{x})}{\boldsymbol{1}_C^{\mathsf{T}}\exp(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{x})},$$

其中$\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_C]$是由$C$个类的权重向量组成的矩阵，$\boldsymbol{1}_C$为$C$维的全1向量，$\hat{\boldsymbol{y}} \in \mathbb{R}^C$为所有类别的预测条件概率组成的向量，第$c$维的值是第$c$类的预测条件概率.

$\boldsymbol{w}_c$是第c类的权重向量

- Softmax 回归的决策函数可以表示为：

$$\hat{y} = \arg\max_{c=1}^{C} p(y=c|\boldsymbol{x}) = \arg\max_{c=1}^{C} \boldsymbol{w}_c^{\mathsf{T}}\boldsymbol{x}.$$

Softmax回归

➢学习准则 --交叉熵损失

➢风险函数

$$\mathcal{R}(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\boldsymbol{y}_c^{(n)}\log\hat{\boldsymbol{y}}_c^{(n)}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{y}^{(n)})^{\mathrm{T}}\log\hat{\boldsymbol{y}}^{(n)}, \qquad \hat{\boldsymbol{y}}^{(n)} = \mathrm{softmax}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}^{(n)})$$

➢ 风险函数的梯度

$$\frac{\partial\mathcal{R}(\boldsymbol{W})}{\partial\boldsymbol{W}} = -\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}\left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}^{(n)}\right)^{\mathrm{T}}$$

推导参见教材P61

Softmax回归-小结

- **模型**

$$P(y=c|\mathbf{x}) = \text{softmax}(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}) = \frac{\exp(\mathbf{w}_c^{\top}\mathbf{x})}{\sum_{i=1}^{C}\exp(\mathbf{w}_i^{\top}\mathbf{x})}.$$

- **学习准则：交叉熵**

$$\mathcal{R}(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{y}^{(n)})^{\mathrm{T}}\log\hat{\boldsymbol{y}}^{(n)},$$

- **优化算法：**梯度下降

$$\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + \alpha\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}\left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}_{\boldsymbol{W}_t}^{(n)}\right)^{\mathrm{T}}\right)$$

$\hat{\boldsymbol{y}}_{\boldsymbol{W}_t}^{(n)}$ 是当参数为 $\boldsymbol{W}_t$ 时，Softmax 回归模型的输出.

感知机（Perceptron）

感知机由 Frank Roseblatt 于1957年提出， 是一种广泛使用的线性分类器。感知器可谓是最简单的人工神经网络，只有一个神经元。

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

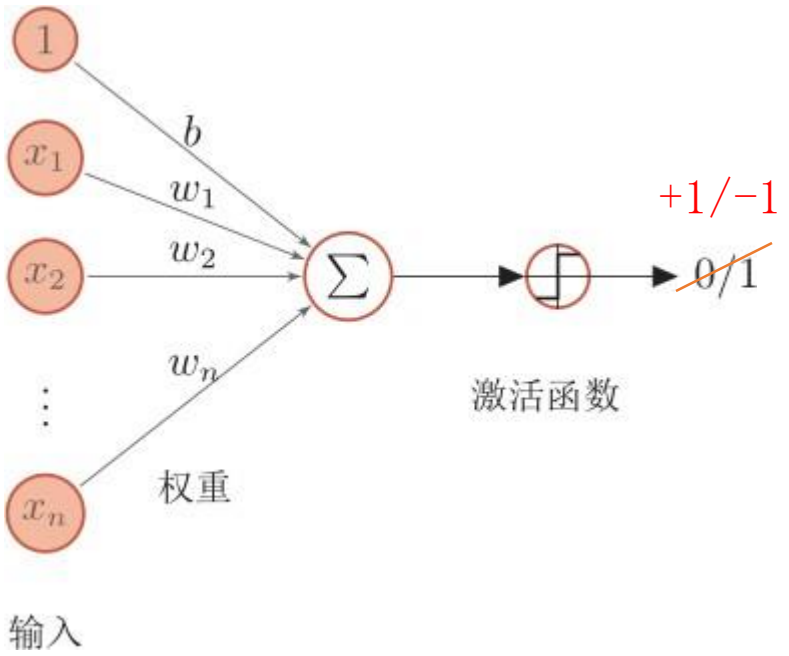
The IBM 704 computer

感知机（Perceptron）

模拟生物神经元行为的机器，有与生物神经元相对应的部件，如权重（突触）、偏置（阈值）及激活函数（细胞体），输出为+1或-1。

$$\hat{y} = \mathrm{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$$

$$= \begin{cases} +1 & \text{当} \mathbf{w}^{\mathrm{T}}\mathbf{x} > 0 \\ -1 & \text{当} \mathbf{w}^{\mathrm{T}}\mathbf{x} \leq 0 \end{cases},$$



感知机（Perceptron）

➢学习算法

- 一种错误驱动的在线学习算法：
- 先初始化一个权重向量$\boldsymbol{w} \leftarrow 0$（通常是全零向量）；
- 每次分错一个样本$(\boldsymbol{x}, y)$时，即

$$y\boldsymbol{w}^T\boldsymbol{x} < \mathbf{0}$$

- 用这个样本来更新权重

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$$

根据感知器的学习策略，可以反推出感知器的损失函数和梯度为：

$$\mathcal{L}(\boldsymbol{w}; \boldsymbol{x}, y) = \max(0, -y\boldsymbol{w}^\top\boldsymbol{x})$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}; \boldsymbol{x}, y)}{\partial \boldsymbol{w}} = \begin{cases} 0 & \text{if} \quad y\boldsymbol{w}^\top\boldsymbol{x} > 0, \\ -y\boldsymbol{x} & \text{if} \quad y\boldsymbol{w}^\top\boldsymbol{x} < 0. \end{cases}$$

感知机（Perceptron）

➢感知器的学习过程

随机梯度下降

算法 3.1: 两类感知器的参数学习算法

输入: 训练集 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，最大迭代次数 T

初始化：$\boldsymbol{w}_0 \leftarrow 0, k \leftarrow 0, t \leftarrow 0$；

repeat

  对训练集 $\mathcal{D}$ 中的样本随机排序;

  for $n = 1 \cdots N$ **do**

    选取一个样本 $(\boldsymbol{x}^{(n)}, y^{(n)})$;

    if $\boldsymbol{w}_k^{\mathrm{T}}(y^{(n)}\boldsymbol{x}^{(n)}) \leq 0$ **then**

      $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + y^{(n)}\boldsymbol{x}^{(n)}$;

      $k \leftarrow k + 1$;

    end

    $t \leftarrow t + 1$;

  end

until $t = T$;

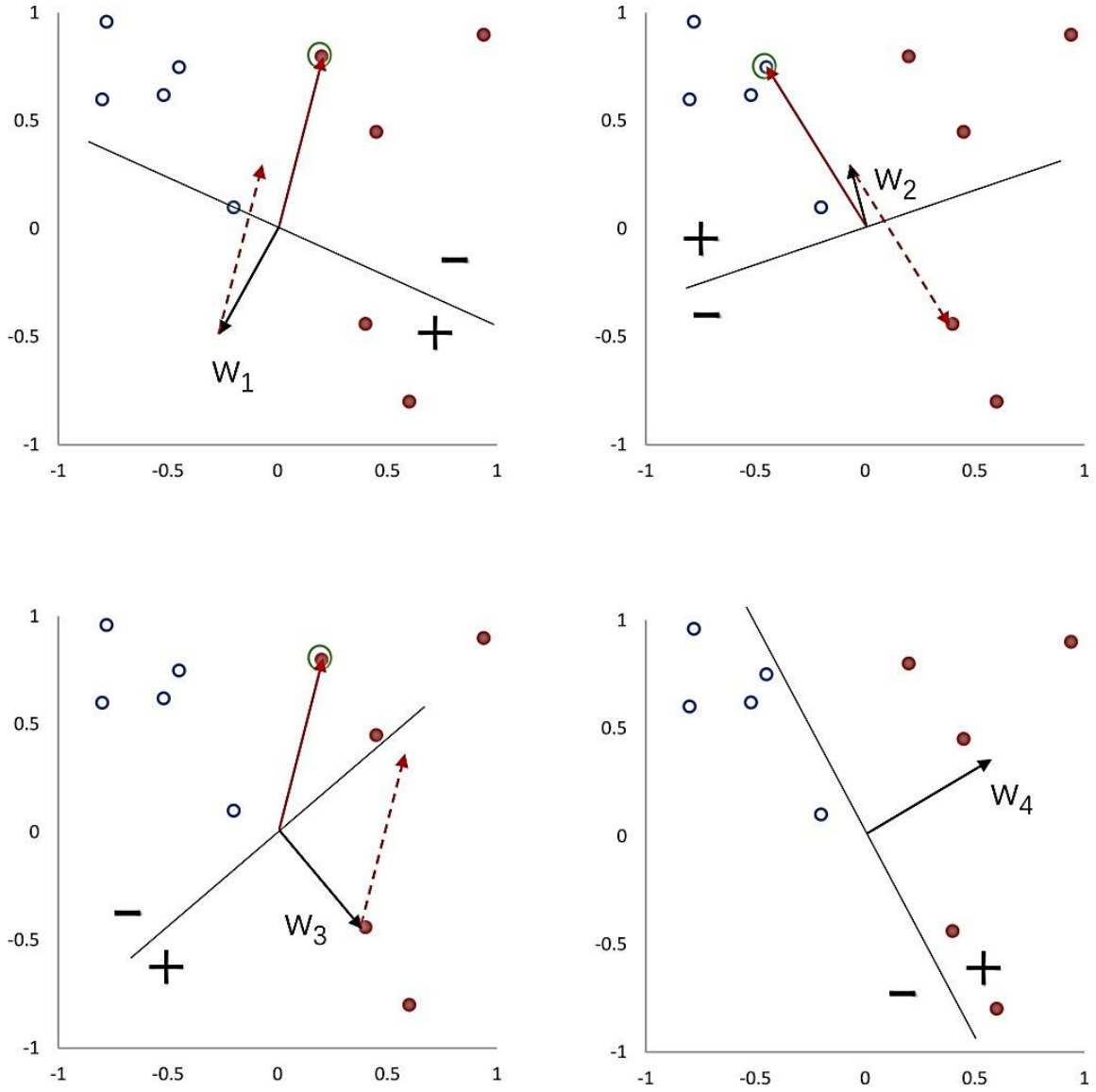
输出: $\boldsymbol{w}_k$

表示分错

感知机（Perceptron）

➢感知器参数学习的更新过程

- ✓ 其中红色实心点为正例， 蓝色空心点为负例。黑色箭头表示当前的权重向量，红色虚线箭头表示权重的更新方向。
- ✓ 假设训练数据集线性可分，感知机学习的目标是求得一个能够将训练集正实例点和负实例点完全正确分开的分离超平面。



感知机（Perceptron）

- **收敛性**：对于两类问题，如果训练集是线性可分的，那么感知器算法可以在有限次迭代后收敛。然而，如果训练集不是线性可分的，那么这个算法则不能确保会收敛。

具体证明，参考书P79

但其存在以下不足：

- ✓ 在数据集线性可分时，感知器虽然可以找到一个超平面把两类数据分开，但并不能保证其泛化能力。
- ✓ 感知器对样本顺序比较敏感。每次迭代的顺序不一致时，找到的分割超平面也往往不一致。
- ✓ 如果训练集不是线性可分的，就永远不会收敛。

感知机（Perceptron）-小结

- **模型**

$$g(\mathbf{x}, \mathbf{w}) = \begin{cases} +1 & \text{当} \quad \mathbf{w}^{\mathrm{T}}\mathbf{x} > 0, \\ -1 & \text{当} \quad \mathbf{w}^{\mathrm{T}}\mathbf{x} < 0. \end{cases}$$

- **学习准则：**

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \max(0, -y\mathbf{w}^{\mathrm{T}}\mathbf{x}).$$

- **优化算法：**随机梯度下降

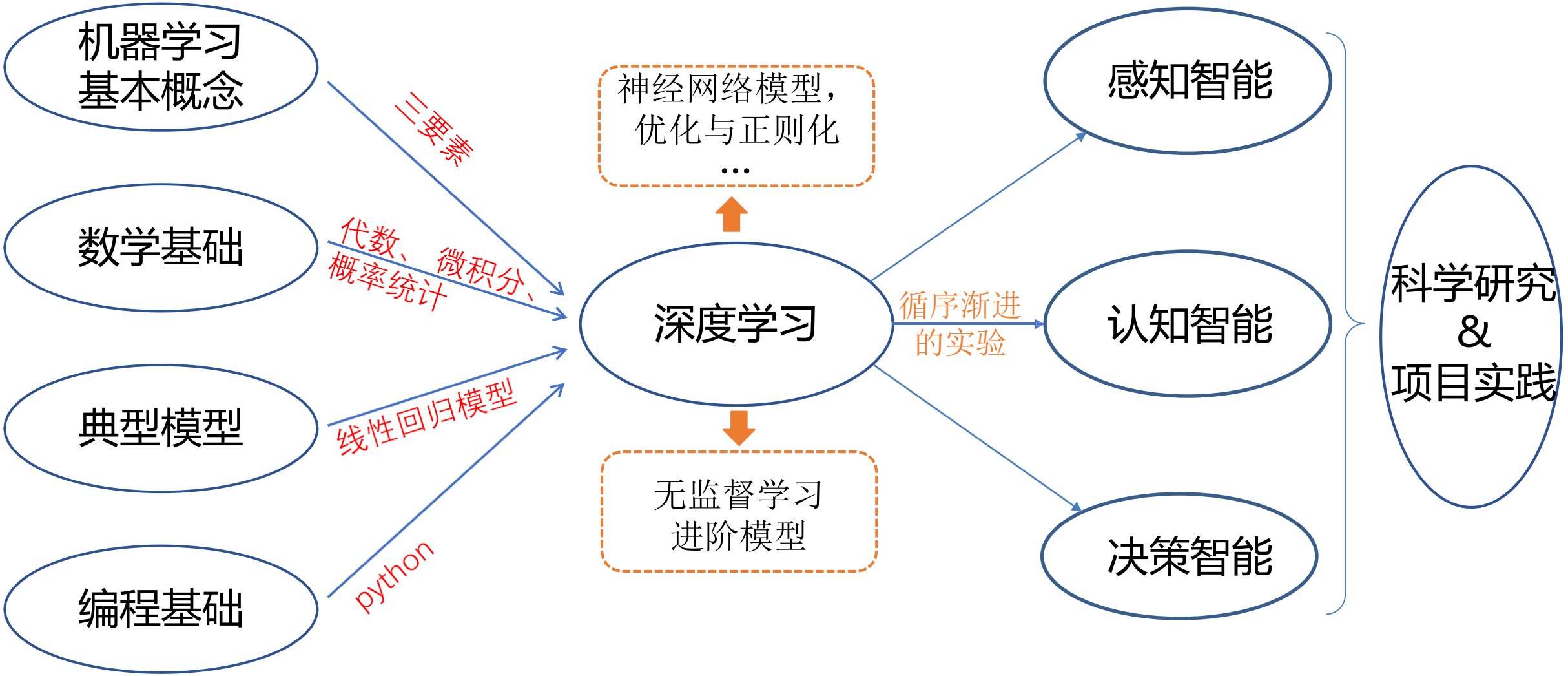
$$\frac{\partial \mathcal{L}(\mathbf{w}; \mathbf{x}, y)}{\partial \mathbf{w}} = \begin{cases} 0 & \text{当} \quad y\mathbf{w}^{\mathrm{T}}\mathbf{x} > 0, \\ -y\mathbf{x} & \text{当} \quad y\mathbf{w}^{\mathrm{T}}\mathbf{x} < 0. \end{cases}$$

线性分类模型小结

线性模型	激活函数	损失函数	优化方法
线性回归	-	$(y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})^2$	最小二乘、梯度下降
Logistic 回归	$\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$	$\boldsymbol{y}\log\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$	梯度下降
Softmax 回归	$\mathrm{softmax}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x})$	$\boldsymbol{y}\log\mathrm{softmax}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x})$	梯度下降
感知器	$\mathrm{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$	$\max(0, -y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$	随机梯度下降

- 在logistic回归和softmax回归中，y为类别的one-hot向量表示；
- 在感知器中，y为$\{+1,-1\}$

基础知识是后续深度学习的基石



参考书与资料

参考书

- 《神经网络与深度学习》，邱锡鹏，复旦大学

参考课程

- **https://rmcong.github.io/proj_deep_learning_ProfessionalCourse.html**