第六章 向量与矩阵微分

第 17 讲 向量函数和矩阵函数

黄定江

DaSE @ ECNU

djhuang@dase.ecnu.edu.cn

1. 17.1 向量函数和矩阵函数
2. 17.2 统计机器学习中的非概率型函数模型
3. 17.3 深度学习中的函数构造

引入：机器学习问题

学习问题是指依据经验数据选取所期望的依赖关系的问题，有两种处理学习问题的方法：

- 基于经验风险泛函最小化
- 基于估计所期望的随机依赖关系（密度、条件密度和条件概率）

学习过程是一个从给定的函数集中选择一个适当函数的过程：

- 非概率相关的函数
- 概率相关的函数

引入：机器学习问题

在机器学习领域，函数集有时也称为假设空间，从数学上，在假设空间中引入恰当的数学结构，可以形成如下空间：

- 距离空间（度量空间）
- 赋范线性空间
- Banach 空间（完备的赋范线性空间）
- 内积空间
- Hilbert 空间（完备的内积空间）
- 欧氏空间（特殊的 Hilbert 空间）

17.1.1 函数：函数定义回顾

设有两个集合 $\mathbb{M}$ 和 $\mathbb{N}$，如果 $\mathbb{M}$ 中每一个元素对应 $\mathbb{N}$ 中唯一的一个元素，则我们称这两个集合是通过函数依赖关系相互关联的。

定义 1

设 $\mathbb{M}$ 和 $\mathbb{N}$ 是两非空集合，若有对应法则 T，使得 $\mathbb{M}$ 内每一个元素 x，都有唯一的一个元素 $y \in \mathbb{N}$ 与它相对应，则称 T 是定义在 $\mathbb{M}$ 上的**函数**，记作

$$T: \mathbb{M} \to \mathbb{N}, \quad x \mapsto y$$

$\mathbb{M}$ 称为 T 的**定义域**；$T(\mathbb{M}) = \{y | y = T(x), x \in \mathbb{M}\}$ 称为 T 的**值域**。

A. 标量值（scalar-valued function）函数：定义

定义 2

设 $\mathbb{M}$ 是一非空集合，当 $\mathbb{N}=\mathbb{R}$ 时，函数 $T:\mathbb{M}\mapsto\mathbb{R}$ 称为实值函数或标量函数。特别当 $\mathbb{M}=\mathbb{N}=\mathbb{R}$ 时，函数 $y=T(x)$ 称为一元实值函数或一元函数。当 $\mathbb{M}=\mathbb{R}^n,\mathbb{N}=\mathbb{R}$ 时，函数 $y=T(x)=T(x_1,x_2,\cdots,x_n)$ 称为多元函数。

注：当 $\mathbb{M}=\mathbb{R}^{m\times n},\mathbb{N}=\mathbb{R}$ 时，函数 $y=T(A)=T(a_{11},a_{12},\cdots,a_{nn})$ 也可称为多元函数，此时，我们相当于把矩阵进行了向量化。

常见的标量值函数：内积函数

例 1

假设 $\boldsymbol{a}$ 是一个 n 维向量，我们可以定义关于 n 维向量 $\boldsymbol{x}$ 的标量值函数：

$$f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x},$$

称为内积函数。

内积函数是线性函数

- 叠加性：对于所有的 n 维向量 $\boldsymbol{x}, \boldsymbol{y}$ 和标量 α, β，例1中定义的函数满足性质：

$$\begin{aligned} f(\alpha\boldsymbol{x}+\beta\boldsymbol{y}) =& \boldsymbol{a}^T(\alpha\boldsymbol{x}+\beta\boldsymbol{y}) = \boldsymbol{a}^T(\alpha\boldsymbol{x}) + \boldsymbol{a}^T(\beta\boldsymbol{y}) \\ =& \alpha(\boldsymbol{a}\boldsymbol{x}) + \beta(\boldsymbol{a}^T\boldsymbol{y}) = \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}) \end{aligned}$$

  这个性质叫做叠加性。

- 一个函数如果满足叠加性，则这个函数称为线性函数。因此内积函数是线性函数。

叠加性的扩展

- 叠加性有时会被拆成两个性质：
  齐次性：对于任意 n 维向量 $\boldsymbol{x}$ 和标量 α 有 $f(\alpha\boldsymbol{x}) = \alpha f(\boldsymbol{x})$
  可加性：对于任意 n 维向量 $\boldsymbol{x}, \boldsymbol{y}$ 有 $f(\boldsymbol{x}+\boldsymbol{y}) = f(\boldsymbol{x}) + f(\boldsymbol{y})$
- 如果一个函数 f 是线性的，叠加性可以拓展到多个向量上：

$$f(\alpha_1\boldsymbol{x}_1 + \cdots + \alpha_k\boldsymbol{x}_k) = \alpha_1 f(\boldsymbol{x}_1) + \cdots + \alpha_k f(\boldsymbol{x}_k)$$

  对任意的 n 维向量 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ 和标量 $\alpha_1, \ldots, \alpha_k$ 成立。

线性函数的内积表示

定理 1

假设函数 f 是一个 n 维向量的标量值函数，并且是线性的。那么存在一个 n 维向量 $\boldsymbol{a}$ 使得 $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x}$ 对于任意 $\boldsymbol{x}$ 成立。我们称 $\boldsymbol{a}^T\boldsymbol{x}$ 为 f 的内积表示，并且是唯一表示。

证明.

（1）首先证明存在性。我们可以把 $\boldsymbol{x}$ 表示为 $\boldsymbol{x} = x_1\boldsymbol{e}_1 + \cdots + x_n\boldsymbol{e}_n$。如果 f 是线性的那么根据叠加性有

$$f(\boldsymbol{x}) = f(x_1\boldsymbol{e}_1 + \cdots + x_n\boldsymbol{e}_n) = x_1 f(\boldsymbol{e}_1) + \cdots + x_n f(\boldsymbol{e}_n) = \boldsymbol{a}^T\boldsymbol{x}$$

其中 $\boldsymbol{a} = (f(\boldsymbol{e}_1), f(\boldsymbol{e}_2), \ldots, f(\boldsymbol{e}_n))$。

（2）下证唯一性。我们不妨设 $f(\boldsymbol{x}) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}$ 并且 $f(\boldsymbol{x}) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}$。令 $\boldsymbol{x} = \boldsymbol{e}_i$，当使用 $f(\boldsymbol{x}) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}$ 时有 $f(\boldsymbol{e}_i) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{e}_i = a_i$。当使用 $f(\boldsymbol{x}) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}$ 时有 $f(\boldsymbol{e}_i) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{e}_i = b_i$。所以 $a_i = b_i$ 对 $i = 1, \ldots, n$ 成立。所以 $\boldsymbol{a} = \boldsymbol{b}$。 □

仿射函数

定义 3

一个线性函数加上一个常数叫做仿射函数。函数 $f: \mathbb{R}^n \to \mathbb{R}$ 是仿射的当且仅当它能够表示成 $f(\boldsymbol{x}) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b$，其中 $\boldsymbol{a}$ 是 n 维向量，b 是标量，有时候被叫做偏置项。

例 2

比如一个 3 维向量的函数

$$f(\boldsymbol{x}) = 2.3 - 2\boldsymbol{x}_1 + 1.3\boldsymbol{x}_2 - \boldsymbol{x}_3$$

它的 $b = 2.3, \boldsymbol{a} = (-2, 1.3, -1)$。

定理 2

任意仿射函数满足如下约束叠加性：

$$f(\alpha \boldsymbol{x}+\beta \boldsymbol{y})=\alpha f(\boldsymbol{x})+\beta f(\boldsymbol{y})$$

其中 $\boldsymbol{x}, \boldsymbol{y}$ 是 n 维向量，α, β 是标量，并且 $\alpha+\beta=1$。

证明.

为了证明带约束的叠加性，我们有：

$$\begin{aligned} f(\alpha \boldsymbol{x}+\beta \boldsymbol{y}) =& \boldsymbol{a}^{\mathrm{T}}(\alpha \boldsymbol{x}+\beta \boldsymbol{y})+b \\ =& \alpha \boldsymbol{a}^{T} \boldsymbol{x}+\beta \boldsymbol{a}^{T} \boldsymbol{y}+(\alpha+\beta) b \\ =& \alpha\left(\boldsymbol{a}^{T} \boldsymbol{x}+b\right)+\beta\left(\boldsymbol{a}^{T} \boldsymbol{y}+b\right) \\ =& \alpha f(\boldsymbol{x})+\beta f(\boldsymbol{y}) \end{aligned}$$

□

约束叠加性

- 对于线性函数，叠加性对于任意的 α, β 都成立，但是对于仿射函数只有它们是仿射组合（即它们的和为 1）时才成立。
- 仿射函数的约束叠加性在证明一个函数不是仿射的时候非常有用，我们只需要寻找向量 $\boldsymbol{x}, \boldsymbol{y}$ 和数 α, β 满足 $\alpha + \beta = 1$ 并且验证 $f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) \neq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y})$ 即可。例如，我们可以证明最大值函数不满足约束叠加性。
- 定理2的结论反过来也是正确的，任意标量值函数只要满足约束叠加性就是仿射函数。
- 如果 $\boldsymbol{x}$ 是标量，此时函数 $f(\boldsymbol{x}) = \alpha\boldsymbol{x} + \beta$ 是一条直线，仿射函数也被称作是线性函数。但是在标准的数学场景下，当 $\beta \neq 0$ 时，$f(\boldsymbol{x}) = \alpha\boldsymbol{x} + \beta$ 不是 $\boldsymbol{x}$ 的线性函数，它是 $\boldsymbol{x}$ 的仿射函数。在本课程中，我们将区分线性函数和仿射函数。但是由线性函数和仿射函数定义的机器学习模型我们统称为线性模型。

非线性标量值函数：二次型、二次函数和范数函数

例 3

- 二次型也是一个非常典型的标量值函数：

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$$

- 将二次型与仿射函数进行叠加得到：

$$f(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}} \boldsymbol{x} + c$$

例 4

常见的向量和矩阵范数也是标量值函数：

- 向量范数：$f(\boldsymbol{x}) = ||\boldsymbol{x}||$
- 矩阵范数：$f(\boldsymbol{A}) = ||\boldsymbol{A}||$

以矩阵为自变量的标量值函数

例 5

常见的以矩阵为自变量的标量值函数：

- 行列式：$f(\boldsymbol{A}) = |\boldsymbol{A}|$
- 秩函数：$f(\boldsymbol{A}) = \mathrm{rank}(\boldsymbol{A})$
- 迹函数：$f(\boldsymbol{A}) = Tr(\boldsymbol{A})$
- 向量 -矩阵 -向量积函数：$f(\boldsymbol{A}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$

注：前面两个是非线性函数，后面两个是线性函数。

B. 向量值函数

定义 4

设 $\mathbb{M}$ 是一非空集合，当 $\mathbb{N}=\mathbb{R}^n$ 时，函数 $T:\mathbb{M}\to\mathbb{R}^n$ 称为向量值函数，简称向量函数。

矩阵 -向量积函数

例 6

假设 $\boldsymbol{A}$ 是一个 $m\times n$ 矩阵。我们可以定义一个关于 n 维向量 $\boldsymbol{x}$ 的向量值函数：

$$f:\mathbb{R}^n\to\mathbb{R}^m,\quad f(\boldsymbol{x})=\boldsymbol{A}\boldsymbol{x},$$

称为矩阵 -向量积函数。当 $m=1$ 时，其退化为内积函数。

矩阵 -向量积函数是线性函数

- 函数 $f\colon \mathbb{R}^n \to \mathbb{R}^m$ 若定义为矩阵 -向量积函数 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$，则它是线性函数，也即满足叠加性：

$$f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y})$$

  对于 n 维向量 $\boldsymbol{x}, \boldsymbol{y}$ 和标量 α, β 成立。我们可以通过矩阵 -向量乘法，向量 -标量乘法来验证叠加性。因此关于 $\boldsymbol{A}$ 的函数 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ 是线性函数。

- 反过来也是正确的。假设 f 是一个将 n 维向量映射为 m 维向量的函数，并且是线性的，则 $f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y})$ 对于所有的 n 维向量 $\boldsymbol{x}, \boldsymbol{y}$ 和所有的标量 α, β 成立，并且存在一个 $m \times n$ 矩阵 $\boldsymbol{A}$ 使得 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ 对所有 $\boldsymbol{x}$ 成立。

仿射函数

定义 5

向量值函数 $f\colon \mathbb{R}^n \to \mathbb{R}^m$ 如果能够写成 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ 的形式，那么 f 是一个仿射函数，其中 $\boldsymbol{A}$ 是 $m \times n$ 矩阵，$\boldsymbol{b}$ 是 m 维向量。

定理 3

函数 $f\colon \mathbb{R}^n \to \mathbb{R}^m$ 是仿射函数当且仅当 $f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y})$ 对于所有的 n 维向量 $\boldsymbol{x}, \boldsymbol{y}$ 和所有的标量 α, β 成立且 $\alpha + \beta = 1$。换句话说，向量的仿射组合具有叠加性。

将仿射函数表示为 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ 的形式，矩阵 $\boldsymbol{A}$ 和向量 $\boldsymbol{b}$ 是唯一的，并且可以使用 $f(0), f(e_1), \ldots, f(e_n)$ 表示，其中 e_k 是 $\mathbb{R}^n$ 中的单位向量：

$$\boldsymbol{A} = [f(e_1) - f(0), f(e_2) - f(0), \ldots, f(e_n) - f(0)], \boldsymbol{b} = f(0).$$

与标量值函数的情形下相同，只有 $\boldsymbol{b} = 0$ 时仿射函数为线性函数。

非线性向量值函数

非线性向量值函数：违反叠加性

例 7

绝对值函数：$f(\boldsymbol{x}) = (|x_1|, |x_2|, \ldots, |x_n|)$ 是非线性向量值函数。取 $n = 1, x = 1, y = 0, \alpha = -1, \beta = 0$ 有

$$f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = 1 \neq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}) = -1$$

例 8

排序函数：f 将 $\boldsymbol{x}$ 的元素降序排列，是非线性向量值函数（$n > 1$）。取 $n = 2, \boldsymbol{x} = (1, 0), \boldsymbol{y} = (0, 1), \alpha = \beta = 1$ 则

$$f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = (1, 1) \neq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}) = (2, 0)$$

C. 矩阵值函数

定义 6

设 $\mathbb{M}$ 和 $\mathbb{N}$ 是两个非空的矩阵集合，函数 $T\colon \mathbb{M} \mapsto \mathbb{N}$ 称为矩阵值函数，简称矩阵函数。

例 9

常见的矩阵函数有

- 考虑一个矩阵 $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ 和 $T\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times n}$，$T(\boldsymbol{L}) = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L}$ 是一个矩阵函数。
- 逆函数：$f(\boldsymbol{A}) = \boldsymbol{A}^{-1}$

17.1.2 算子

定义 7

设 X 和 Y 是同一数域 $\mathbb{K}$ 上的线性赋范空间，若 T 是 X 的某个子集 D 到 Y 中的一个映射，则称 T 为子集 D 到 Y 中的算子。称 D 为算子 T 的定义域，或记为 $D(T)$；并称 Y 的子集 $TD = \{y = T(x), x \in D\}$ 为算子 T 的值域。对于 $x \in D$，通常记 x 的像 $T(x)$ 为 Tx。

注 1：上面算子的定义，从狭义的角度是指从一个函数空间到另一个函数空间（或它自身）的映射；从广义的角度看，可以把线性赋范空间推广到一般空间，包括向量空间和内积空间，或更进一步 Banach 空间和 Hilbert 空间等。

注 2：当 $X = Y = \mathbb{R}$ 时，算子 T 就是微积分中的函数，因此算子是函数概念的推广。

定义 8

设 X 和 Y 是同一数域 $\mathbb{K}$ 上的线性赋范空间，$\boldsymbol{x}_0 \in D \subset X$，$T$ 为 D 到 Y 中的算子，如果 $\forall \epsilon > 0, \exists \delta > 0$，当 $\|\boldsymbol{x}-\boldsymbol{x}_0\| < \delta$，有 $\|T\boldsymbol{x}-T\boldsymbol{x}_0\| < \epsilon$，则称算子 T 在点 $\boldsymbol{x}_0$ 处连续。若算子 T 在 D 中每一点都连续，则称 T 为 D 上的连续算子。

定义 9

设 X 和 Y 是同一数域 $\mathbb{K}$ 上的线性赋范空间，$D \subset X$，T 为 D 到 Y 中的算子，如果 $\forall \boldsymbol{x}, \boldsymbol{y} \in D, \forall \alpha, \beta \in \mathbb{K}$，有 $T(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) = \alpha T(\boldsymbol{x}) + \beta T(\boldsymbol{y})$，则称 T 为 D 上的线性算子.

定义 10

设 X 和 Y 是同一数域 $\mathbb{K}$ 上的线性赋范空间，$D \subset X$，$T: D \to Y$ 为线性算子，如果存在 $M > 0, \forall x \in D$，有 $Tx \leq Mx$，则称 T 为 D 上的线性有界算子，或称 T 有界。

例 10

- (1) 恒等算子 $I: X \to X$ 定义为，$\forall x \in X, Ix = x.$
- (2) 零算子 $0: X \to Y$ 定义为，$\forall x \in X, 0x = \theta.$
- (3) 设 $C^{(1)}[a,b]$ 是 $[a,b]$ 上所有一阶导函数连续的函数组成的空间，微分算子 $D: C^{(1)}[a,b] \to C[a,b]$ 定义为 $\forall x(t) \in C^{(1)}[a,b]$,
$$Dx = \frac{d}{dt}x(t)$$
- (4) 积分算子 $T: C[a,b] \to C[a,b]$ 定义为 $\forall x(t) \in C[a,b]$,
$$Tx = \int_a^t x(\tau)\,d\tau, t \in [a,b]$$
- (5) 设矩阵 $A = (a_{ij})_{m\times n}$，$a \in \mathbb{R}$，矩阵算子 $T: \mathbb{R}^n \to \mathbb{R}^m$ 定义为
$$\forall \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n, T\boldsymbol{x} = A\boldsymbol{x} = \boldsymbol{y}$$
其中 $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$

例 11

验证积分算子 T 为线性有界算子。

积分算子 $T\colon C[a,b]\to C[a,b]$ 定义为，$\forall x(t)\in C[a,b]$ $Tx=\int_a^t x(\tau)\,d\tau$ $t\in[a,b]$。

证明.

证明设 $x(t),y(t)\in C[a,b]$ $\alpha,\beta\in\mathbb{R}$，则

$$T(\alpha x+\beta y)=\int_a^t(\alpha x(\tau)+\beta y(\tau))\,d\tau=\alpha\int_a^t x(\tau)\,d\tau+\beta\int_a^t y(\tau)\,d\tau=\alpha Tx+\beta Ty$$

$$Tx=\max_{t\in[a,b]}\left|\int_a^t x(\tau)\,d\tau\right|\le\max_{t\in[a,b]}\int_a^t|x(\tau)|\,d\tau\le\max_{t\in[a,b]}|x(t)|\int_a^t 1\,d\tau=\|x(t)\|(b-a)$$

于是积分算子 T 为线性有界算子。 □

其它常见的算子有：梯度算子，散度算子，拉普拉斯算子，哈密顿算子等。

算子构造

每个算子 A 唯一地将集合 $\mathbb{M}$ 中的元素映射到集合 $\mathbb{N}$ 中的元素。这一过程可以用方程表示：

$$A\mathbb{M} = \mathbb{N}$$

我们从算子集中挑选出实现 $\mathbb{M}$ 到 $\mathbb{N}$ 的一对一映射的算子。对于这些算子，解算子方程

$$Af(t) = F(t)$$

的问题可以看成在 $\mathbb{M}$ 中寻找元素 $f(t)$，它刚好对应 $\mathbb{N}$ 中的元素 $F(x)$。

17.1.3 泛函

定义 11

设 X 为实（或复）线性赋范空间，则由 X 到实（或复）数域的算子称为泛函。

例 12

例如，若 $x(t)$ 是任意一个可积函数：$x(t) \in L^1[a,b]$，则其积分

$$f(x) = \int_a^b x(t)\,dt$$

就是一个定义在 $L^1[a,b]$ 上的泛函，而且是线性的：

$$f(\alpha x + \beta y) = \alpha \int_a^b x(t)\,dt + \beta \int_a^b y(t)\,dt = \alpha f(x) + \beta f(y)$$

还是有界的：

$$|f(x)| \le \int_a^b |x(t)|\,dt = \|x\|$$

例 13

设 $x(t) \in C[a,b]$，η 是 $[a,b]$ 上任一固定点，则 $\delta_\eta(x) = x(\eta)$ 是定义在 $C[a,b]$ 上的有界线性泛函。它就是熟知的单位脉冲函数 δ 函数。

例 14

令 $J(x) = \int_a^b g(x(t),t)\,dt$，其中 g 为二元连续函数。则 $J(x)$ 是定义在 $C[a,b]$ 上的泛函，但一般地它不是线性的。如果 $g(x,t)$ 的偏导数 g'_x 存在且有界，则泛函 $J(x)$ 是连续的，这是因为

$$|J(x_1) - J(x_2)| \le \int_a^b |g(x_1,t) - g(x_2,t)|dt \le \int_a^b |g'_x(\eta,t)|dt \|x_1 - x_2\|_{C[a,b]} \le M\|x_1 - x_2\|$$

例 15

设 X 为线性赋范空间，则 $f(x) = \|x\|$ 是连续泛函，但非线性。

17.1.4 机器学习中的风险泛函

下面讨论机器学习中寻找函数依赖关系的模型，称之为从实例学习的模型。模型包括 3 个组成部分（如图所示）：

1. 数据（实例）的发生器 G。
2. 目标算子 S（有时称为训练器算子，或简单地称为训练器）。
3. 学习机器 LM。

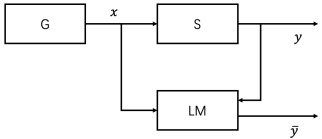


图 1: 从实例学习的模型。在学习过程中，学习机器观测一系列点对 (x, y)（训练集）。训练后，机器对任何一个给定的 x 必须返回一个 $\bar{y}$ 值。目标是返回一个非常接近于训练器响应 y 的 $\bar{y}$。

从实例学习的一般方法过程如下：

- 首先，要确定训练器将采用何种类型的算子。假定训练器依据条件分布函数 $F(y|x)$ 返回向量 x 上的输出值 y（它包括了训练器采用函数 $y = f(x)$ 的情形）
- 学习机器观察训练集，该训练集是依据联合分布函数 $F(x, y) = F(x)F(y|x)$ 随机独立抽取出来的。利用这一训练集，学习机器构造对未知算子的逼近，也即构造一个机器来实现某一固定的函数集。
- 因此，学习过程是一个从给定的函数集中选择一个适当函数的过程。如何选择函数？将依赖于恰当的评价准则来进行选取。
- 每当遇到用所期望的评价准则来选取一个函数的问题时，都可以考虑这样一个模型：在所有可能的函数中，找出一个函数，它以最佳可能方式满足给定的评价准则。

- 在形式上，这种处理方式的含义是，在向量空间 $\mathbb{R}^n$ 的子集 $\mathbb{Z}$ 上，给定一个容许函数集 $|g(z)|, z \in \mathbb{Z}$，定义一个泛函：

$$R = R(g(z))$$

  该泛函就是选取函数的评价准则，然后需要从函数集 $|g(z)|$ 中找出一个最小化泛函的函数 $g^*(z)$。

- 假定泛函的最小值对应于最好的评价，且 $|g(z)|$ 中存在泛函的最小值。在显式地给出函数集 $|g(z)|$ 和泛函 $R(g(z))$ 的情况下，寻找最小化 $R(g(z))$ 的函数 $g^*(z)$，这个问题是变分法的研究主题。

我们考虑另外一种情况，即在 Z 上定义概率分布函数 $F(z)$，并将泛函定义为数学期望：

$$R(g(z)) = \int L(z, g(z))\, dF(z)$$

其中，函数 $L(z, g(z))$ 对任意 $g(z) \in |g(z)|$ 都是可积的。现在的问题是，在未知概率分布 $F(z)$，但得到了依据 $F(z)$ 独立地随机抽取出的观测样本

$$z_1, \ldots, z_t$$

的情况下，最小化泛函 $R(g(z)) = \int L(z, g(z))\, dF(z)$。

风险泛函

当用公式给出上述最小化问题时，函数集 $g(z)$ 是以参数的方式给出的：$|g(z,a), a \in \Lambda|$

定义 12

函数 $Q(z, a^*) = L(z, g(z, a^*))$ 的期望损失是由下列积分确定的

$$R(a^*) = \int Q(z, a^*) dF(z)$$

这一泛函称为风险泛函或者风险。

当概率分布函数未知，但给定了随机独立观测数据 $z_1, \ldots, z_t$ 时，我们的问题是在函数集 $Q(z, a), a \in \Lambda$ 中选取一个最小化风险的函数 $Q(z, a_0)$。

经验风险泛函

经验损失 (empirical loss)

给定一个训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$$

模型 $f(X)$ 关于训练数据集的平均损失称为经验风险或经验损失，记作 R_{emp}:

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)), \tag{1}$$

经验风险最小化准则

经验风险最小化 (empirical risk minimization, ERM)

在假设空间、损失函数以及训练数据集确定的情况下，经验风险函数式(1)就可以确定。经验风险最小化的策略认为，经验风险最小的模型是最优的模型。根据这一策略，按照经验风险最小化求最优模型就是求解最优化问题：

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)), \tag{2}$$

其中，$\mathcal{F}$ 是假设空间。

当样本容量足够大时，经验风险最小化能保证有很好的学习效果。

结构风险泛函

经验风险最小化时常常会出现过拟合现象，我们可以通过引入所谓的结构风险最小化策略来防止过拟合。

结构风险 (structural risk)

在假设空间、损失函数以及训练数据集确定的情况下，结构风险定义为在经验风险上加上表示模型复杂度的正则化项或罚项：

$$R_{srm}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f), \tag{3}$$

其中 $J(f)$ 为模型的复杂度，是定义在假设空间 $\mathcal{F}$ 上的泛函。模型 f 越复杂，复杂度 $J(f)$ 就越大；反之，模型 f 越简单，复杂度 $J(f)$ 就越小。也就是说，复杂度表示了对复杂模型的惩罚。$\lambda \geq 0$ 是系数，用以权衡经验风险和模型复杂度。结构风险小需要经验风险与模型复杂度同时小。结构风险小的模型往往对训练数据以及未知的测试数据都有较好的预测。

结构风险最小化准则和正则化

结构风险最小化准则 (structural risk minimization, SRM)

结构风险最小化策略认为结构风险最小的模型是最优的模型。所以求最优模型，就是求解最优化问题：

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f), \tag{4}$$

上述最优化问题一般也称为正则化 (regularization)，正则化是结构风险最小化策略的实现。

数据科学中的向量与矩阵函数

在数据科学中，我们常常在三个地方遇到向量函数或者矩阵函数。

- 机器学习模型（机器学习模型部分的函数）
- 损失函数（机器学习策略部分的函数）
- 目标函数（机器学习算法部分的函数）

下面我们将分别举一些机器学习中相关的例子，并且着重讲述一些相关的特殊向量函数与矩阵函数。

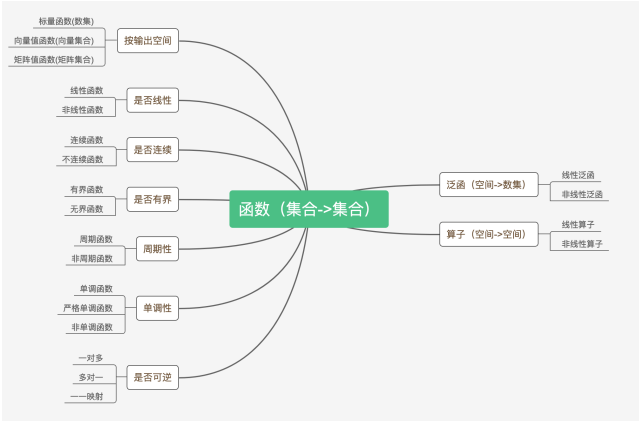

函数（集合->集合）
按输出空间
标量函数(数集)
向量值函数(向量集合)
矩阵值函数(矩阵集合)
是否线性
线性函数
非线性函数
是否连续
连续函数
不连续函数
是否有界
有界函数
无界函数
周期性
周期函数
非周期函数
单调性
单调函数
严格单调函数
非单调函数
是否可逆
一对多
多对一
一一映射
泛函（空间->数集）
线性泛函
非线性泛函
算子（空间->空间）
线性算子
非线性算子

图 2: 函数、算子和泛函

17.2.1 线性模型中的函数

例 16

给定由 d 个属性描述的示例 $\boldsymbol{x} = (x_1; x_2; \ldots; x_d)$，其中 x_i 是 $\boldsymbol{x}$ 在第 i 个属性上的取值，线性模型 *(linear model)* 试图学得一个通过属性的线性组合来进行预测的函数，即

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b,$$

其中 $\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$。$\boldsymbol{w}$ 和 b 学得之后，模型就得以确定。

- 线性模型中的函数是仿射函数
- 线性模型可以用于机器学习中的回归和分类，分别对应于线性回归和线性判别

线性回归模型中的函数

给定数据集 $\mathbb{T} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ...(\boldsymbol{x}_m, y_m)\}$，其中 $\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{id})$，$y_i \in \mathbb{R}$。

- 模型函数。线性回归试图学得一个线性模型以尽可能准确地预测实值输出标记，也即学得

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b使得f(\boldsymbol{x}_i) \simeq y_i$$

- 损失函数。如何确定 $\boldsymbol{w}$ 和 b 呢？基本策略是把模型预测的结果 $f(x_i)$ 与真实标记 y_i 进行比较，也即衡量 $f(x)$ 与 y 之间的差别。通过引入损失函数，均方误差（也即平方损失，是回归任务中最常用的性能度量）来度量：

$$L(f; T) = \frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{x}_i) - y_i)^2$$

- 目标函数。因此我们可试图让均方误差最小化，即

$$(\boldsymbol{w}^*, b^*) = \arg\min_{(\boldsymbol{w},b)}\sum_{i=1}^{m}(f(\boldsymbol{x}_i) - y_i)^2 = \arg\min_{(\boldsymbol{w},b)}\sum_{i=1}^{m}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b - y_i)^2$$

线性回归模型中的函数变体

- 如果我们令模型的预测值逼近 y 的变形，比如我们认为示例所对应的输出标记是在指数尺度上变化，那就可以将输出标记的对数作为线性模型逼近的目标，即

$$\ln y = \boldsymbol{w}^T \boldsymbol{x} + b$$

这个模型叫做对数线性回归。

- 更一般地，考虑单调可微函数 $g(\cdot)$，令

$$y = g^{-1}(\boldsymbol{w}^T \boldsymbol{x} + b)$$

这样得到的模型称为广义线性模型，其中 $g(\cdot)$ 称为“联系函数”。显然，对数线性回归是广义线性模型在 $g(\cdot) = \ln(\cdot)$ 时的特例。

线性回归模型中的函数变体

- 对二分类任务，当任务输出标记为 $y \in \{0, 1\}$ 时，而线性回归模型产生的预测值 $z = \boldsymbol{w}^T \boldsymbol{x} + b$ 是实值，于是我们需要将实值 z 转换为 $0/1$ 值，可以通过单位阶跃函数（unit-step function）来实现：

$$y = \begin{cases} 0, & z < 0 \\ 0.5, & z = 0 \\ 1, & z > 0 \end{cases}$$

即若预测值 z 大于零就判为正例，小于零则判为反例，预测值为临界值可任意判别。

- 但是，单位阶跃函数不连续，可以使用对数几率函数 $y = \frac{1}{1+e^{-z}}$ 来替代广义线性模型中联系函数的反函数 g^{-1}。这样我们就可以得到对数几率回归的模型：

$$y = \frac{1}{1 + e^{\boldsymbol{w}^T \boldsymbol{x} + b}}$$

线性回归模型中的函数变体

- 类似对数线性回归，对数几率回归可以变化为

$$\ln \frac{y}{1-y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

如果将 y 视为样本 $\boldsymbol{x}$ 作为正例的可能性，则 $1-y$ 是其反例可能性，两者的比值

$$\frac{y}{1-y}$$

称为几率，反应了 $\boldsymbol{x}$ 作为正例的相对可能性，对几率取对数则得到“对数几率”

$$\ln \frac{y}{1-y}$$

从空间的角度来理解线性回归模型中的函数关系

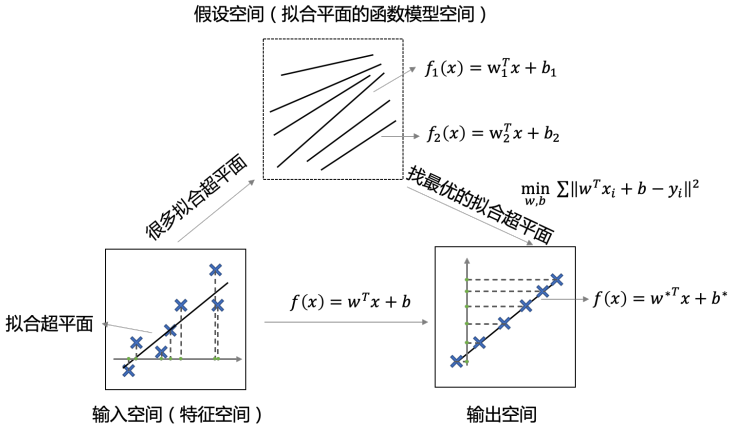


图 3: 线性回归

17.2.2 感知机模型中的函数

例 17

（感知机）假设输入空间（特征空间）是 $\mathbb{X} \subset \mathbb{R}^n$，输出空间是 $\mathbb{Y} = \{+1, -1\}$。输入 $x \in \mathbb{X}$ 表示实例的特征向量，对应于输入空间（特征空间）的点；输出 $y \in \mathbb{Y}$ 表示实例的类别。由输入空间到输出空间的如下函数：

$$f(x) = \mathbf{sign}(\boldsymbol{w}^T \boldsymbol{x} + b)$$

称为感知机。其中，$\boldsymbol{w}$ 和 b 为感知机模型参数，$\boldsymbol{w} \in \mathbb{R}^n$ 叫作权值（*weight*）或权值向量（*weight vector*),$b \in \mathbb{R}$ 叫作偏置（*bias*），$\boldsymbol{w}^T \boldsymbol{x}$ 表示 $\boldsymbol{w}$ 和 $\boldsymbol{x}$ 的内积。**sign** 是符号函数，即

$$\mathbf{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

感知机的模型函数

- 感知机是一种线性分类模型，属于判别模型。感知机模型的假设空间是定义在特征空间中的所有线性分类模型（linear classification model）或线性分类器（linear classifier），即函数集合 $\{f|f(x) = \boldsymbol{w}^T\boldsymbol{x} + b\}$。
- 函数模型对应的线性方程 $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ 称为对应于特征空间的分离超平面，它由法向量 $\boldsymbol{w}$ 和截距 b 决定，可用 $(\boldsymbol{w}, b)$ 来表示。分离超平面将特征空间划分为两部分，一部分是正类，一部分是负类。法向量指向的一侧为正类，另一侧为负类。
- 为了找出这样的超平面，即确定感知机模型参数 $\boldsymbol{w}, b$，需要确定一个学习策略，即定义（经验）损失函数并将损失函数极小化。

定义 13

数据集的线性可分性 给定一个数据集

$$T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$$

其中，$x_i\in\mathbb{X}=\mathbb{R}^n$，$y_i\in\mathbb{Y}=\{+1,-1\},i=1,2,\dots,N$。如果存在某个超平面 S

$$\boldsymbol{w}^T\boldsymbol{x}+b=0$$

能够将数据集的正实例点和负实例点完全正确地划分到超平面的两侧，即对所有 $y_i=1$ 的实例 i，有 $\boldsymbol{w}^T\boldsymbol{x}_i+b>0$，对所有 $y_i=-1$ 的实例 i，有 $\boldsymbol{w}^T\boldsymbol{x}_i+b<0$，则称数据集 T 为线性可分数据集（*linearly separable dataset*）；否则，称数据集 T 线性不可分。

感知机中的损失函数：误分类最小策略

- 假设训练数据集是线性可分的，为了定义损失函数，首先写出输入空间 $\mathbb{R}^n$ 中任一点 x_0 到超平面 S 的距离：

$$\frac{1}{\|\boldsymbol{w}\|}|\boldsymbol{w}^T\boldsymbol{x}_0 + b|$$

这里，$\|\boldsymbol{w}\|$ 是 $\boldsymbol{w}$ 的 L_2 范数。误分类点 x_i 到超平面 S 的距离是

$$-\frac{1}{\|\boldsymbol{w}\|}y_i(\boldsymbol{w}^T\boldsymbol{x}_0 + b)$$

- 这样，假设超平面 S 的误分类点集合为 M，那么所有误分类点到超平面 S 的总距离为

$$-\frac{1}{\|\boldsymbol{w}\|}\sum_{x_i\in M} y_i(\boldsymbol{w}^T\boldsymbol{x}_0 + b)$$

不考虑 $\frac{1}{\|\boldsymbol{w}\|}$，就得到感知机学习的损失函数：

$$L(\boldsymbol{w}, b) = -\sum_{x_i\in M} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)$$

其中 M 为误分类点的集合。这个损失函数就是感知机学习的经验风险函数。

感知机中的目标函数

感知机学习算法的原始形式

感知机学习算法是对以下最优化问题的算法。给定一个训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

其中，$x_i \in \mathbb{X} = \mathbb{R}^n$，$y_i \in \mathbb{Y} = \{+1, -1\}, i = 1, 2, \ldots, N$。求参数 $\boldsymbol{w}, b$，使其为以下损失函数极小化问题的解

$$\min_{\boldsymbol{w}, b} L(\boldsymbol{w}, b) = -\sum_{x_i \in M} y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

其中 M 为误分类点的集合。$L(\boldsymbol{w}, b)$ 为感知机模型中的目标函数。

注：在感知机模型中，损失函数和目标函数是一致的。

从空间的角度来理解感知机模型中的函数关系

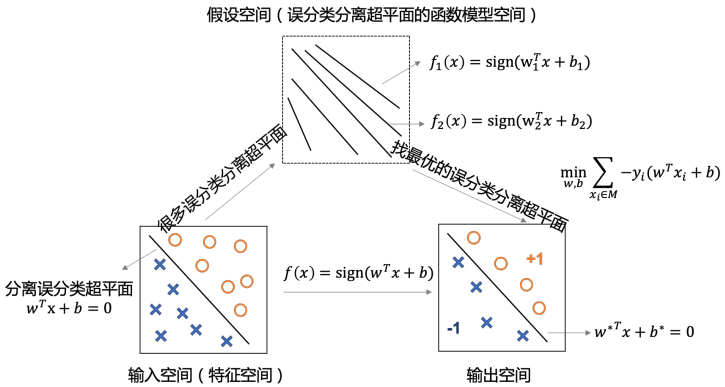


图 4: 感知机

17.2.3 支持向量机

支持向量机是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机。按照训练数据的特征，支持向量机分为 3 种类型：

- 线性可分支持向量机：当训练数据线性可分时，通过硬间隔最大化学习得到的线性分类器，又称为硬间隔支持向量机
- 线性支持向量机：当训练数据近似线性可分时，通过软间隔最大化学习得到的线性分类器，又称为软间隔支持向量机
- 非线性支持向量机：当训练数据线性不可分时，通过使用核技巧（Kernel trick）及软间隔最大化，学习得到的非线性分类器

支持向量机

- 考虑一个二类分类问题。假设输入空间与特征空间为两个不同的空间。输入空间为欧氏空间或离散集合，特征空间为欧氏空间或希尔伯特空间。
- 线性可分支持向量机、线性支持向量机假设这两个空间的元素一一对应，并将输入空间中的输入映射为特征空间中的特征向量。
- 非线性支持向量机利用一个从输入空间到特征空间的非线性映射将输入映射为特征向量。
- 所以，输入都由输入空间转换到特征空间，支持向量机的学习是在特征空间进行的。

A. 线性可分支持向量机：模型函数

例 18

给定线性可分训练数据集，通过间隔最大化或等价地求解相应的凸二次规划问题学习得到的分离超平面为

$$\boldsymbol{w}^{*T}\boldsymbol{x} + b^* = 0$$

以及相应的分类决策函数

$$f(x) = \mathbf{sign}(\boldsymbol{w}^{*T}\boldsymbol{x} + b^*)$$

称为线性可分支持向量机。

分类策略：函数间隔

一般来说，一个点距离分离超平面的远近可以表示分类预测的确信程度。在超平面 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$ 确定的情况下，$|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b|$ 能够相对地表示点距离超平面的远近。而 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 的符号与类标记 y 的符号是否一致能够表示分类是否正确。所以可用量 $y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)$ 来表示分类的正确性及确信度，这就是函数间隔（functional margin）的概念。

定义 14

给定训练数据集 T 和超平面 (w,b)，定义超平面关于样本点 (x_i,y_i) 的函数间隔为

$$\hat{\gamma}_i=y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)$$

定义超平面 (w,b) 关于训练数据集 T 的函数间隔为超平面 (w,b) 关于 T 中所有样本点 (x_i,y_i) 的函数间隔的最小值，即

$$\hat{\gamma}=\min_{i=1,\ldots,N}\hat{\gamma}_i$$

分类策略：几何间隔

定义 15

给定训练数据集 T 和超平面 (w,b)，定义超平面关于样本点 (x_i,y_i) 的几何间隔为

$$\gamma_i = y_i(\frac{w}{\|w\|}x_i + \frac{b}{\|w\|})$$

定义超平面 (w,b) 关于训练数据集 T 的几何间隔为超平面 (w,b) 关于 T 中所有样本点 (x_i,y_i) 的几何间隔的最小值，即

$$\gamma = \min_{i=1,\ldots,N} \gamma_i$$

函数间隔和几何间隔有如下的关系

$$\gamma_i = \frac{\hat{\gamma}_i}{\|w\|}$$

$$\gamma = \frac{\hat{\gamma}}{\|w\|}$$

学习算法：几何间隔最大化

- 支持向量机学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。一般地，当训练数据集线性可分时，存在无穷个分离超平面可将两类数据正确分开。感知机利用误分类最小的策略，求得分离超平面，不过这时的解有无穷多个，但是线性可分支持向量机利用几何间隔最大化求最优分离超平面，这时，解是唯一的。这里的间隔最大化又称为硬间隔最大化（与将要讨论的训练数据集近似线性可分时的软间隔最大化相对应）。
- 间隔最大化的直观解释是：对训练数据集找到几何间隔最大的超平面意味着以充分大的确信度对训练数据进行分类。也就是说，不仅将正负实例点分开，而且对最难分的实例点（离超平面最近的点）也有足够大的确信度将它们分开。这样的超平面应该对未知的新实例有很好的分类预测能力。

学习算法：几何间隔最大化

下面考虑如何求得一个几何间隔最大的分离超平面，即最大间隔分离超平面。具体地，这个问题可以表示为下面的约束最优化问题：

$$\max_{w,b} \gamma$$

$$s.t. \quad y_i\left(\frac{w}{\|w\|}x_i + \frac{b}{\|w\|}\right) \geq \gamma, i = 1, 2, \ldots, N$$

考虑几何间隔和函数间隔的关系，可以将问题改写成

$$\max_{w,b} \frac{\hat{\gamma}}{\|w\|}$$

$$s.t. \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq \hat{\gamma}, i = 1, 2, \ldots, N$$

更进一步，将 $\hat{\gamma} = 1$ 代入上面的最优化问题，注意到最大化 $\frac{1}{\|w\|}$ 和最小化 $\frac{1}{2}\|w\|^2$ 是等价的，于是就得到下面的线性可分支持向量机学习的最优化问题：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t. \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 \geq 0, i = 1, 2, \ldots, N$$

从空间的角度来理解线性可分支持向量机中的函数关系

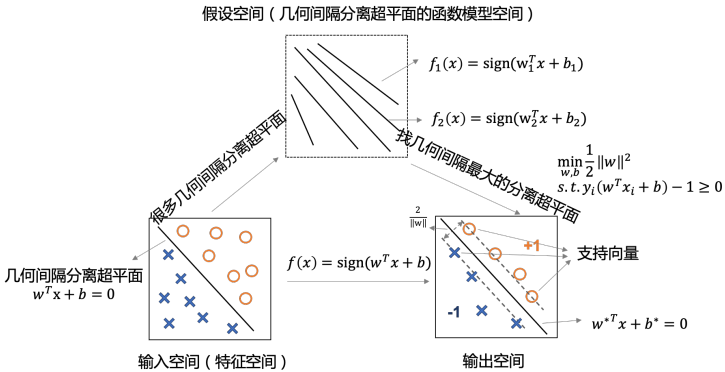


图 5: 线性可分支持向量机

B. 线性支持向量机：软间隔最大化

- 线性可分问题的支持向量机学习方法，对线性不可分训练数据是不适用的。线性不可分意味着某些样本点 (x_i, y_i) 不能满足函数间隔大于等于 1 的约束条件。
- 缓解该问题的一个办法是允许支持向量机在一些样本点上出错。为此要引入“软间隔”的概念，它允许某些样本不满足约束

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1,$$

但是，在最大化间隔的同时，不满足约束的样本应尽可能少。

软间隔最大化

- 这样目标函数由原来的 $\frac{1}{2}\|\boldsymbol{w}\|^2$ 变成

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} L_{0/1}(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1)$$

且软间隔优化目标可写为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} L_{0/1}(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1),$$

这里，$L_{0/1}$ 是“0/1 损失函数”

$$L_{0/1}(z) = \begin{cases} 1, & if \ \ z < 0 \\ 0, & otherwise \end{cases};$$

$C > 0$ 称为惩罚参数，一般由应用问题决定，C 值大时对误分类的惩罚增大，C 值小时对误分类的惩罚减小。最小化目标函数包含两层含义：使 $\frac{1}{2}\|\boldsymbol{w}\|^2$ 尽量小即间隔尽量大，同时使误分类点的个数尽量小，C 是调和二者的系数。显然，当 C 为无穷大时，上式迫使所有样本均满足约束 $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$；当 C 取有限值时，上式允许一些样本不满足约束。

然而，$L_{0/1}$ 非凸、非连续，数学性质不太好，使得该优化问题不易直接求解，于是，人们通常用其他一些函数来代替 $L_{0/1}$，称为“替代损失”(surrogate loss)．替代损失函数一般具有较好的数学性质，如它们通常是凸的连续函数且是 $L_{0/1}$ 的上界。下面是三种常用的替代损失函数：

- hinge 损失：$L_{hinge}(z) = \max(0, 1 - z)$；
- 指数损失 (exponential loss)：$L_{exp}(z) = \exp(-z)$；
- 对率损失 (logistic loss)：$L_{log}(z) = \log(1 + \exp(-z))$

若采用 hinge 损失，则软件隔优化目标变成

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

引入“松弛变量”(slack variables)$\xi_i \geq 0$，可重写为

$$\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$s.t.\ \ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, N$$

$$\xi_i \geq 0, i = 1, 2, \ldots, N$$

这就是常用的线性支持向量机，也称为“软间隔支持向量机”。

线性支持向量机：学习算法

原始问题

线性不可分的线性支持向量机的学习问题变成如下凸二次规划（convex quadratic programming）问题：

$$\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t. \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, N$$

$$\xi_i \geq 0, i = 1, 2, \ldots, N$$

线性支持向量机：模型函数

定义 16

(线性支持向量机）对于给定的线性不可分的训练数据集，通过求解凸二次规划问题，即软间隔最大化问题，得到的分离超平面为

$$\boldsymbol{w}^{*T}\boldsymbol{x} + b^* = 0$$

以及相应的分类决策函数

$$f(\boldsymbol{x}) = \mathbf{sign}(\boldsymbol{w}^{*T}\boldsymbol{x} + b^*)$$

称为线性支持向量机。

从空间的角度来理解线性支持向量机中的函数关系

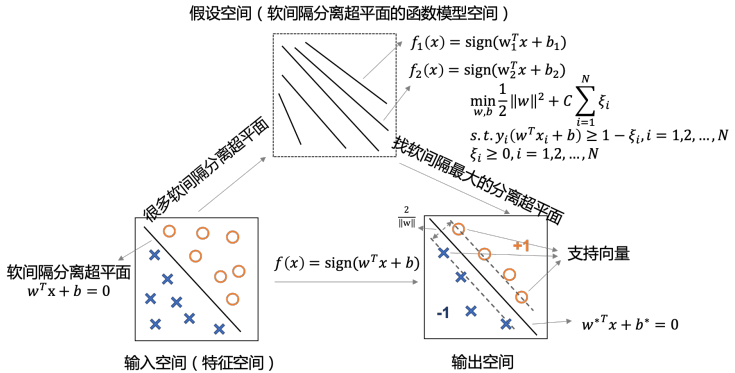


图 6: 线性支持向量机

C. 非线性支持向量机与核函数：非线性分类问题

- 非线性分类问题是指通过利用非线性模型才能很好地进行分类的问题。对给定的一个训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$，其中，实例 x_i 属于输入空间，$x_i\in\mathbb{X}=\mathbb{R}^n$，对应的标记有两类 $y_i\in\mathbb{Y}=\{+1,-1\},i=1,2,\ldots,N$。如果能用 $\mathbb{R}^n$ 中的一个超曲面将正负例正确分开，则称这个问题为非线性可分问题。
- 非线性问题往往不好求解，所以希望能用解线性分类问题的方法解决这个问题。所采取的方法是进行一个非线性变换，将非线性问题变换为线性问题，通过解变换后的线性问题的方法求解原来的非线性问题。

例 19

设原空间为 $\mathbb{X} \subset \mathbb{R}^2, x = (x^{(1)}, x^{(2)})^T \in \mathbb{X}$，新空间为 $\mathbb{Z} \subset \mathbb{R}^2, z = (z^{(1)}, z^{(2)})^T \in \mathbb{Z}$，定义从原空间到新空间的变换（映射）：

$$z = \phi(x) = (((x^{(1)})^2, (x^{(2)})^2))^T$$

经过变换 $z = \phi(x)$，原空间 $\mathbb{X} \subset \mathbb{R}^2$ 变换为新空间 $\mathbb{Z} \subset \mathbb{R}^2$，原空间中的点相应地变换为新空间中的点，原空间中的超曲面（比如一个椭圆）

$$w_1(x^{(1)})^2 + w_2(x^{(2)})^2 + b = 0$$

变换成为新空间中的直线

$$w_1 z^{(1)} + w_2 z^{(2)} + b = 0$$

在变换后的新空间里，直线 $w_1 z^{(1)} + w_2 z^{(2)} + b = 0$ 可以将变换后的正负实例点正确分开。这样，原空间的非线性可分问题就变成了新空间的线性可分问题。

用线性分类方法求解非线性分类问题步骤：

从上述例子可以看出，用线性分类方法求解非线性分类问题分为两步：

- 首先使用一个变换将原空间的数据映射到新空间；
- 然后在新空间里用线性分类学习方法从训练数据中学习分类模型。

关键问题：如何构造变换？可以用核函数来实现。使用核函数的分类方法称为核技巧或核方法。

核函数

定义 17

设 $\mathbb{X}$ 是输入空间（欧氏空间 $\mathbb{R}^n$ 的子集或者离散集合），又设 $\mathbb{H}$ 为特征空间（希尔伯特空间），如果存在一个从 $\mathbb{X}$ 到 $\mathbb{H}$ 的映射

$$\phi(x) : \mathbb{X} \to \mathbb{H}$$

使得对所有 $x, z \in \mathbb{X}$，函数 $K(x, z)$ 满足条件

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

则称 $K(x, z)$ 为核函数，$\phi(x)$ 为映射函数。

核技巧的想法是：在学习与预测中只定义核函数 $K(x, z)$，而不显式地定义映射函数 ϕ。通常，直接计算 $K(x, z)$ 比较容易，而通过 $\phi(x)$ 和 $\phi(z)$ 计算 $K(x, z)$ 并不容易。注意 ϕ 是输入空间 $\mathbb{R}^n$ 到特征空间 $\mathbb{H}$ 的映射，特征空间 $\mathbb{H}$ 一般是高维的，甚至是无穷维的。此外，对于给定的 $K(x, z)$，特征空间 $\mathbb{H}$ 和映射函数 ϕ 的取法并不唯一，可以取不同的特征空间，即便是在同一特征空间也可以取不同的映射。

核函数和映射函数关系的例子

例 20

假设输入空间是 $\mathbb{R}^2$，核函数是 $K(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z})^2$，试找出其相关的特征空间 $\mathbb{H}$ 和映射 $\phi(\boldsymbol{x}):\mathbb{R}^2\to\mathbb{H}$。

解：取特征空间 $\mathbb{H}=\mathbb{R}^3$，记 $\boldsymbol{x}=(x^{(1)},x^{(2)})^T, z=(z^{(1)},z^{(2)})^T$，由于

$$(\boldsymbol{x}^T\boldsymbol{z})^2=(x^{(1)}z^{(1)}+x^{(2)}z^{(2)})^2=(x^{(1)}z^{(1)})^2+2x^{(1)}z^{(1)}x^{(2)}z^{(2)}+(x^{(2)}z^{(2)})^2$$

所以可以取映射 $\phi(\boldsymbol{x})=((x^{(1)})^2,\sqrt{2}x^{(1)}x^{(2)},(x^{(2)})^2)^T$，

容易验证 $\phi(\boldsymbol{x})^T\phi(\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z})^2=K(\boldsymbol{x},\boldsymbol{z})$。

仍取 $\mathbb{H}=\mathbb{R}^3$ 以及 $\phi(\boldsymbol{x})=\frac{1}{\sqrt{2}}((x^{(1)})^2-(x^{(2)})^2,2x^{(1)}x^{(2)},(x^{(1)})^2+(x^{(2)})^2)^T$，

同样有 $\phi(\boldsymbol{x})^T\phi(\boldsymbol{z})=(\boldsymbol{x}^T\boldsymbol{z})^2=K(\boldsymbol{x},\boldsymbol{z})$。

还可以取 $\mathbb{H}=\mathbb{R}^4$ 和 $\phi(\boldsymbol{x})=((x^{(1)})^2,x^{(1)}x^{(2)},x^{(1)}x^{(2)},(x^{(2)})^2)^T$。

核函数的判定和构造

定理 4

设 $K:\mathbb{X}\times\mathbb{X}\to\mathbb{R}$ 是对称函数，则 $K(\boldsymbol{x},\boldsymbol{z})$ 为正定核函数的充要条件是对任意 $\boldsymbol{x}_i\in\mathbb{X}, i=1,2,\ldots,m$，$K(\boldsymbol{x},\boldsymbol{z})$ 对应的 *Gram* 矩阵

$$\boldsymbol{K}=[K(\boldsymbol{x}_i,\boldsymbol{x}_j)]_{m\times m}$$

是半正定矩阵。

定义 18

设 $\mathbb{X}\subset\mathbb{R}^n$，$K(\boldsymbol{x},\boldsymbol{z})$ 是定义在 $\mathbb{X}\times\mathbb{X}$ 上的对称函数，如果对于任意 $\boldsymbol{x}_i\in\mathbb{X}, i=1,2,\ldots,m$，$K(\boldsymbol{x},\boldsymbol{z})$ 对应的 *Gram* 矩阵

$$\boldsymbol{K}=[K(\boldsymbol{x}_i,\boldsymbol{x}_j)]_{m\times m}$$

是半正定矩阵，则称 $K(\boldsymbol{x},\boldsymbol{z})$ 是正定核。

常见的核函数

- 线性核函数：$\kappa(x,z)=x^Tz+c$
- 多项式核函数：$\kappa(x,z)=(x^Tz)^d$
- 高斯核函数:$\kappa(x,z)=e^{\frac{||x-z||^2}{-2\sigma^2}}$
- 拉普拉斯核:$\kappa(x,z)=e^{\frac{||x-z||}{-\sigma}}$
- Sigmoid 核：$\kappa(x,z)=\tan(ax^tz+c)$
- 字符串核函数
- $\cdots$

核函数性质一

若 K_1, K_2 为核函数，则对于任意正数 γ_1, γ_2，其线性组合

$$\gamma_1 K_1 + \gamma_2 K_2$$

是核函数。

核函数性质二

若 K_1, K_2 为核函数，则核函数的直积

$$K_1 \otimes K_2(\boldsymbol{x}, \boldsymbol{z}) = K_1(\boldsymbol{x}, \boldsymbol{z}) K_2(\boldsymbol{x}, \boldsymbol{z})$$

是核函数。

核函数性质三

若 K_1 为核函数，则对于任意函数 $g(x)$，

$$K(\boldsymbol{x}, \boldsymbol{z}) = g(\boldsymbol{x}) K_1(\boldsymbol{x}, \boldsymbol{z}) g(\boldsymbol{z})$$

是核函数。

核技巧应用到支持向量机，其基本想法：

- 通过一个非线性变换将输入空间（欧氏空间 $\mathbb{R}^n$ 或离散集合）对应于一个特征空间（希尔伯特空间 $\mathbb{H}$），使得在输入空间 $\mathbb{R}^n$ 中的超曲面模型对应于特征空间 $\mathbb{H}$ 中的超平面模型（支持向量机）。
- 这样分类问题的学习任务通过在特征空间中求解线性支持向量机就可以完成。
- 在核技巧中，我们并不需要显式地定义映射函数，而是通过核函数来隐式地定义映射函数。
- 在通常情况下，我们只需要将一个线性模型化成带有内积的形式，然后将内积部分替换成核函数即可。

非线性支持向量机

定义 19

（非线性支持向量机）从非线性分类训练集，通过核函数与软间隔最大化，学习得到的分类决策函数

$$f(\boldsymbol{x}) = \mathbf{sign}(\sum_{i=1}^{N} \alpha_i^* y_i K(x, x_i) + b^*)$$

称为非线性支持向量机，$K(x, z)$ 是正定核函数。

非线性支持向量机学习算法

选取适当的核函数 $K(x, z)$ 和适当的参数 C，构造并求解最优化问题：

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1, 2, \ldots, N$$

从空间的角度来理解非线性支持向量机中的函数关系

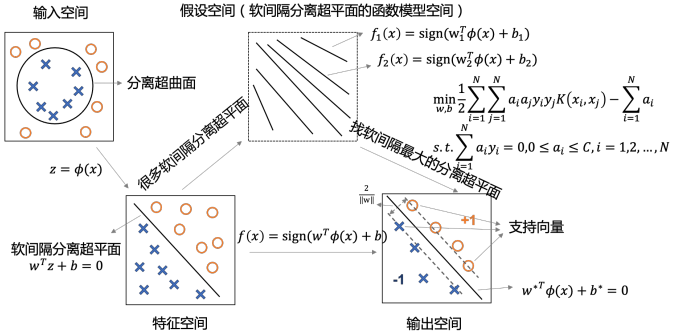


图 7: 支持向量机

17.2.4 降维和主成分分析中函数

维数灾难

在数据分析和机器学习领域，在高维情形下所有机器学习方法都面临数据样本稀疏、距离计算困难等问题，称为“维数灾难”（curse of dimensionality)。

维数约简

维数约简也称降维，是缓解维数灾难的一个重要途径，即通过某种数学变换将原始高维属性空间转变为一个低维“子空间”。

线性和非线性降维

一般来说，欲获得低维子空间，主要有两类方法：

- 线性降维：对原始高维空间进行线性变换，代表性的方法有主成分分析（简称 PCA）。
- 非线性降维：对原始高维空间进行非线性变换，代表性的方法有流形学习。

主成分分析的基本思想

- 主成分分析属于多元统计分析的经典方法，首先由 Pearson 于 1901 年提出，但只针对非随机变量，1933 年由 Hotelling 推广到随机变量。
- 统计分析中，数据的变量之间可能存在相关性，以致增加了分析的难度。于是考虑由少数不相关的变量来代替相关的变量，用来表示数据，并且要求能够保留数据中的大部分信息。
- 主成分分析主要利用正交变换把由线性相关变量表示的观测数据转换为少数几个由线性无关变量表示的数据，线性无关的变量称为主成分。主成分的个数通常小于原始变量的个数。所以主成分分析属于降维方法。

主成分分析中的模型函数

假设给定 d 维原始空间中的 m 个样本 $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m) \in \mathbb{R}^{d\times m}$，主成分分析通过模型函数

$$f(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{x}$$

将 m 个样本变换到 $d' \leq d$ 维子空间中

$$\boldsymbol{Z} = \boldsymbol{W}^T\boldsymbol{X},$$

其中 $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$ 是正交变换矩阵，$\boldsymbol{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_m) \in \mathbb{R}^{d'\times m}$ 是新空间 (d' 维子空间) 中的 m 个样本，也即原始样本在新空间中的表达。

主成分分析中的策略损失函数

那么如何选择新的空间（这等价于选择正交变换矩阵 $\boldsymbol{W}$），使得新空间中的 m 个样本是原始空间中的 m 个样本的一个恰当的表达？我们把这个新空间想象成是由超平面张成的，可通过两种策略来选择正交变换矩阵 $\boldsymbol{W}$：

- 最近重构性：样本点到这个超平面的距离都足够近；
- 最大可分性：样本点在这个超平面上的投影能尽可能分开.

可以证明，基于最近重构性和最大可分性这两种策略，能分别得到主成分分析的两种等价推导。我们先从最近重构性来推导，关于最大可分性，我们将在后面概率论部分进行介绍。

主成分分析中的策略损失函数

假定数据样本进行了中心化，即 $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$，再假定投影变换后得到的新坐标系为 $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'}\}$，其中 $\boldsymbol{w}_i$ 是标准正交基向量。令 $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'})$，那么 $\boldsymbol{W}$ 就是一个投影矩阵，所以新样本点在子空间中的坐标为 $\boldsymbol{z}_i = \boldsymbol{W}^T\boldsymbol{x}_i$。而这些样本点在原始空间中的坐标为 $\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i$。我们在原始空间中考虑原样本点 $\boldsymbol{x}_i$ 与基于投影重构的样本点 $\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i$ 之间的距离为

$$\|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i\|_2$$

那么考虑整个训练集，对于所有样本总的距离为

$$\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}\|_F,$$

这可以看成投影变换后新样本和原样本之间的损失函数。

目标函数

我们只需要最小化这个距离即可，这等价于优化

$$\begin{aligned}\min_{\boldsymbol{W}} \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}\|_F^2 &= \min_{\boldsymbol{W}} \mathsf{Tr}((\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})^T(\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})) \\ &= \min_{\boldsymbol{W}} \mathsf{Tr}(\boldsymbol{X}^T\boldsymbol{X} - 2\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}) \\ &= \min_{\boldsymbol{W}} \mathsf{Tr}(-\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\end{aligned}$$

最后我们还需要注意 $\boldsymbol{W}$ 是一个正交矩阵，并且利用迹函数的轮换性

$$\begin{aligned}&\min_{\boldsymbol{W}} \mathsf{Tr}(-\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}) \\ &s.t.\, \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}\end{aligned}$$

是主成分分析的优化目标.

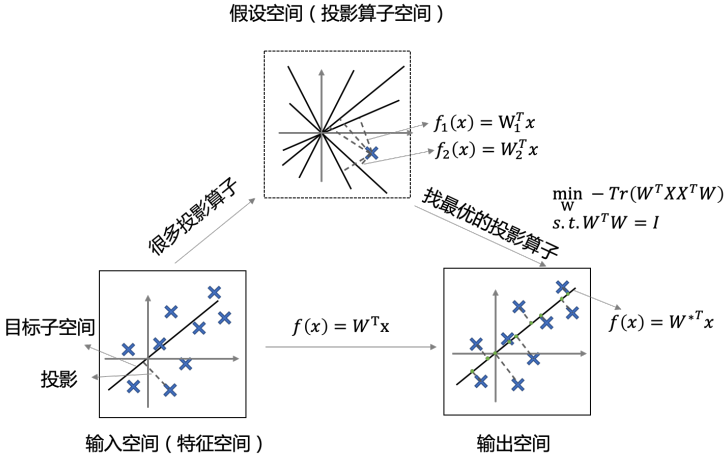

假设空间（投影算子空间）
$f_1(x) = \mathrm{W}_1^T x$
$f_2(x) = W_2^T x$
很多投影算子
找最优的投影算子
$\min_{\mathrm{W}} \ -Tr(W^T X X^T W)$
$s.t. W^T W = I$
目标子空间
投影
$f(x) = W^{\mathrm{T}}\mathrm{x}$
$f(x) = W^{*T} x$
输入空间（特征空间）
输出空间

图 8: 降维

17.2.5 聚类中的函数

聚类

聚类是针对给定的样本，依据他们特征的相似度或距离，将其归并到若干个“类”或“簇”的数据分析问题。一个类是给定样本集合的一个子集。直观上，相似的样本聚集在相同的类，不相似的样本分散在不同的类。这里，样本之间的相似度或距离起着重要的作用。

聚类目的

聚类的目的是通过得到类或簇来发现数据的特点或对数据进行处理，在数据挖掘，模式识别等领域有着广泛的运用。聚类属于无监督学习，因为只是根据样本的相似度或距离来将其进行归类，而类或簇事先并不知道。

聚类算法有很多，主要有两类方法：

- 层次聚类：
- k 均值聚类：

k 均值聚类中的模型函数

- 给定 n 个样本的集合 $\mathbb{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n\}$，每个样本由一个特征向量表示，特征向量的维数是 m。k 均值聚类的目标是将 n 个样本分到 k 个不同的类或簇中，这里假设 $k<n$。k 个类 $G_1,G_2,\ldots,G_k$ 形成对样本集合 $\mathbb{X}$ 的划分，其中 $G_i\cap G_j=\emptyset,\cup_{i=1}^{k}G_i=\mathbb{X}$。用 C 表示划分，一个划分对应着一个聚类结果。
- 划分 C 是一个多对一的函数。事实上，如果把每个样本用一个整数 $i\in\{1,2,\ldots,n\}$ 表示，每个类也用一个整数 $l\in\{1,2,\ldots,k\}$ 表示，那么划分或者聚类可以用函数

$$l=C(i)$$

表示，其中 $i\in\{1,2,\ldots,n\},l\in\{1,2,\ldots,k\}$。所以 k 均值聚类的模型是一个从样本到类的函数。

k 均值聚类中的策略损失函数

- k 均值聚类归结为样本集合 $\mathbb{X}$ 的划分，或者从样本到类的函数的选择问题。k 均值聚类的策略是通过损失函数的最小化选取最优的划分或函数 C^*。
- 首先，采用欧氏距离平方（squared Euclidean distance）作为样本之间的距离 $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{m}(x_{ki} - x_{kj})^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

- 然后，定义样本与其所属类的中心之间的距离的总和为损失函数，即

$$W(C) = \sum_{l=1}^{k}\sum_{C(i)=l}\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_l\|^2$$

式中 $\bar{\boldsymbol{x}}_l = (\bar{x}_{1l}, \bar{x}_{2l}, \ldots, \bar{x}_{ml})$ 是第 l 个类的均值或中心，$n_l = \sum_{i=1}^{n} I(C(i) = l), I(C(i) = l)$ 是指示函数，取值为 1 或 0。函数 $W(C)$ 也称为能量，表示相同类中的样本相似的程度。

- k 均值聚类就是求解最优化问题：

$$C^* = \arg\min_C W(C) = \arg\min_C \sum_{l=1}^{k} \sum_{C(i)=l} \| Vx_i - \bar{\boldsymbol{x}}_l \|^2$$

- 相似的样本被聚到同类时，损失函数值最小，这个目标函数的最优化能达到聚类的目的。但是，这是一个组合优化问题，n 个样本分到 k 类，所有可能分法的数目是：

$$S(n,k) = \frac{1}{k!} \sum_{l=1}^{k} (-1)^{k-l} \binom{k}{l} k^n$$

这个数字是指数级的。事实上，k 均值聚类的最优解求解问题是 NP 困难问题。现实中采用迭代的方法求解。

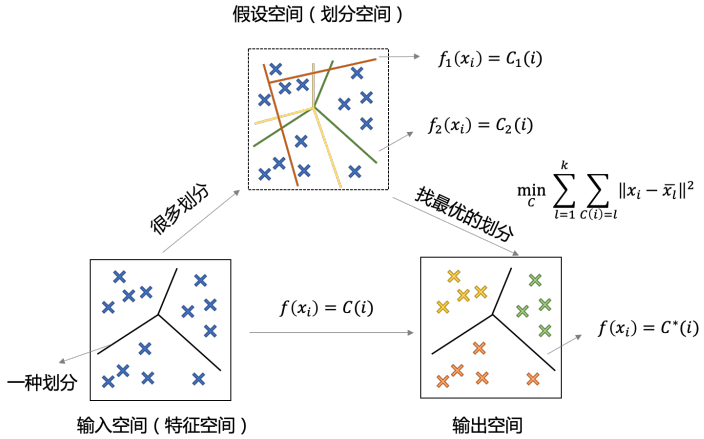

假设空间（划分空间）
$f_1(x_i) = C_1(i)$
$f_2(x_i) = C_2(i)$
$\min_C \sum_{l=1}^{k} \sum_{C(i)=l} \|x_i - \bar{x}_l\|^2$
很多划分
找最优的划分
一种划分
$f(x_i) = C(i)$
$f(x_i) = C^*(i)$
输入空间（特征空间）
输出空间

图 9: k 均值聚类

17.3.1 深度神经网络中的函数构造：分类手写数字回顾

例 21

在 *MNIST* 数字识别的任务中，假设我们把训练图像数据集看作 28×28 维向量空间 $\mathbb{R}^{784}$，图片向量为 $\boldsymbol{x}$；把标签不再看作一个数字，如果标签为 i，那么我们把它看作只有第 i 个分量为 1，其余分量为 0 的 10 维向量 $\boldsymbol{y}$，则所有标签向量在 10 维向量空间 $\mathbb{R}^{10}$ 中。对于训练集中的每个 $\boldsymbol{x}$，已知它所代表的数字。我们想要找到一个函数 f(也即分类规则，位于假设空间中)，

$$f\colon \mathbb{R}^{784} \to \mathbb{R}^{10}$$

$$\boldsymbol{y}' = f(\boldsymbol{x})$$

将 $\mathbb{R}^{784}$ 维向量空间中的输入，映射到 10 维向量空间中去，每个输入对应的输出在 0 到 9 之间，其中 $\boldsymbol{y}'$ 也是 $\mathbb{R}^{10}$ 中的向量。机器学习试图学习到这个函数，使其适用于 (大部分) 训练图像，并且在测试集中也能获得好的表现，这一基本要求称为泛化。我们可以通过使 $\|\boldsymbol{y}' - \boldsymbol{y}\|$ 尽可能小，也即求解最优化问题

$$\min \|\boldsymbol{y}' - \boldsymbol{y}\|$$

线性函数和仿射函数复合分类手写数字

首先，我们想到这个函数 $f(\boldsymbol{x})$ 应是 $\mathbb{R}^{784}$ 到 $\mathbb{R}^{10}$ 上的线性函数 (一个 $10 \times p$ 矩阵)。十个输出是数字 0 到 9 的概率，我们将通过 10^p 个条目和 M 个训练样本来得到近似正确的结果。

1. 线性函数和线性函数的复合分类手写数字

   如果我们令 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ 或者令 $f(\boldsymbol{x}) = f_2(f_1(\boldsymbol{x})) = \boldsymbol{A}_2\boldsymbol{A}_1\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}$，则优化问题变为

   $$\min_{\boldsymbol{A}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|$$

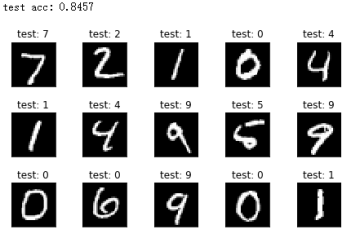
   其中 $\boldsymbol{A}$ 是参数矩阵，复合函数 $f_2(f_1(\boldsymbol{x}))$ 表示先用 f_1 将图像映射成 50 维的向量，再用 f_2 将 50 维的向量映射为 10 维的向量。

2. 仿射函数和仿射函数的复合分类手写数字

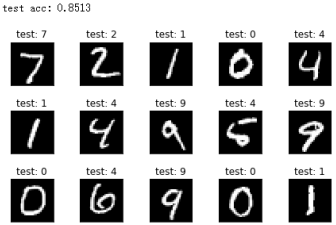
   如果我们令 $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ 或者令 $f(\boldsymbol{x}) = f_2(f_1(\boldsymbol{x})) = \boldsymbol{A}_2(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2 = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$，则优化问题变为

   $$\min_{\boldsymbol{A},\boldsymbol{b}} \|\boldsymbol{A}\boldsymbol{x} + b - \boldsymbol{y}\|$$

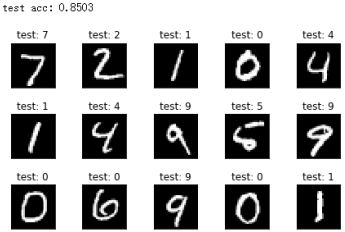
   其中 $\boldsymbol{A}, \boldsymbol{b}$ 是参数矩阵。



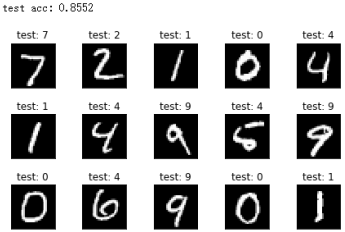
(a) 线性映射分类准确率



(b) 双层线性网络准确率



(c) 仿射映射模型分类准确率



(d) 仿射映射复合模型分类准确率

非线性函数及其复合分类手写数字

但是，线性函数的泛化能力是十分受限的。从艺术角度上看，两个 0 可以构成 8，1 和 0 可以组合成手写体的 9 或是 6，而图像不具有可加性，因而它的输入 -输出规则远不是线性的。如果我们将线性函数改为非线性函数，重新构造模型并使用优化算法求解，最终得到的模型的分类准确率产生了一定的提高。

3. 非线性函数分类手写数字

   如果我们令 $f(\boldsymbol{x}) = \mathrm{ReLU}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$，其中 $\boldsymbol{A}, \boldsymbol{b}$ 是参数矩阵，$\mathrm{ReLU}(\boldsymbol{x}) = \boldsymbol{x}_+ = \max\ (\boldsymbol{x}, 0)$ 是非线性函数，则优化问题变为

$$\min_{\boldsymbol{A},\boldsymbol{b}} \|\mathrm{ReLU}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) - \boldsymbol{y}\|$$

- 非线性函数复合分类手写数字

  如果我们令 $f(\boldsymbol{x}) = f_2(f_1(\boldsymbol{x})) = \mathrm{ReLU}(\boldsymbol{A}_2\mathrm{ReLU}(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$，其中 $\boldsymbol{A}_1, \boldsymbol{b}_1, \boldsymbol{A}_2, \boldsymbol{b}_2$ 是参数矩阵，则优化问题变为

$$\min_{\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{b}_1,\boldsymbol{b}_2} \|\mathrm{ReLU}(\boldsymbol{A}_2\mathrm{ReLU}(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) - \boldsymbol{y}\|$$

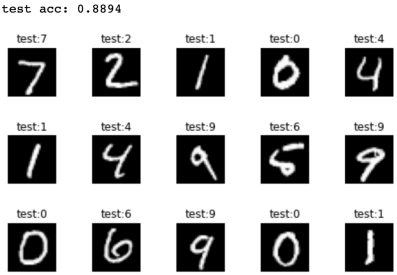


图 10: 非线性映射分类准确率

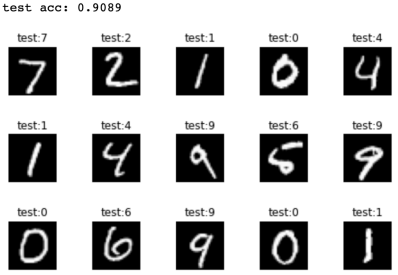


图 11: 双层非线性网络准确率

17.3.2 深度神经网络中的函数构造：从浅层网络到深度网络

- 上述例子中，我们在双层非线性网络中使用了 ReLU 函数，这是一个连续分片（或分段）线性 (CPL) 函数，这里线性是为了保持简单起见，连续性是为了建模一条未知但合理的规则，而分段用于实现真实图像和数据所要求的非线性。这个函数的使用是机器学习中一个超越预期的成功发现，它把浅层学习转化为深度学习。

定义 20

如果函数 $f: \mathbb{R}^n \to \mathbb{R}$，对于任意一个点 $\boldsymbol{x} \in \mathbb{R}^n$，存在一个无洞的子集 $\mathbb{I} \subset \mathbb{R}^n$ 包含 $\boldsymbol{x}$，使得 f 在 $\mathbb{I}$ 上是一个一次函数，则称 f 为**分片线性函数**。

CPL 函数所在的假设空间是连续分片线性函数空间。这带来了可计算性的关键问题：什么参数能够快速描述一大族 CPL 函数？我们首先来看看 CPL 函数的构造。

连续分片线性函数构造

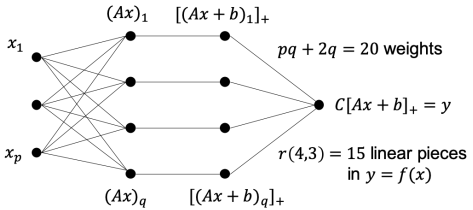


图 12: 数据向量 $\boldsymbol{x}$ 的分片线性函数的神经网络架构

上图是数据向量 $\boldsymbol{x}$ 的分片线性函数的初步构造：

- 首先确定矩阵 $\boldsymbol{A}$ 和向量 $\boldsymbol{b}$;
- 接着将 $\boldsymbol{Ax}+\boldsymbol{b}$ 中所有的负分量设为 0(此步是非线性的)。
- 随后乘上矩阵 $\boldsymbol{C}$，得到输出 $\boldsymbol{y} = F(\boldsymbol{x}) = \boldsymbol{C}(\boldsymbol{Ax}+\boldsymbol{b})_+$。向量 $(\boldsymbol{Ax}+\boldsymbol{b})_+$ 形成了在输入 $\boldsymbol{x}$ 和输出 $\boldsymbol{y}$ 间的“隐藏层”。

连续分片线性函数图像

- 在上图中，$(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})_+$ 的每个分量都是双半平面的 (由于 $\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$ 中负分量处为 0，其中一个半平面是水平的)。
- 若 $\boldsymbol{A}$ 是 $q\times p$ 的矩阵，输入空间 $\mathbb{R}^P$ 将被 q 个超平面分割成 $r(p,q)$ 个部分，这些分块是可数的，它度量了整个函数 $F(\boldsymbol{x})$ 的“表达性”，其中

$$r(p,q)=C_q^0+C_q^1+\cdots+C_q^p$$

这个数字给出了 F 的图像的一个描述，尽管 F 的形式还没有明确给出。

例 22

令

$$\boldsymbol{A}=\begin{pmatrix}2 & -1\\ -1 & 1\end{pmatrix},\boldsymbol{C}=\begin{pmatrix}-1 & 1\end{pmatrix},\boldsymbol{b}=\begin{pmatrix}1\\ 2\end{pmatrix},$$

考虑 $\boldsymbol{y}=F(\boldsymbol{x})=\boldsymbol{C}\mathrm{ReLU}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})$ 的图像。可以看到函数的输入空间 $\mathbb{R}^2$ 被两个超平面 $2x_1-x_2+1=0$，$-x_1+x_2+2=0$ 划分成了四个区域。函数 $\boldsymbol{y}=F(\boldsymbol{x})$ 在每一个区域中约束为一个线性函数。

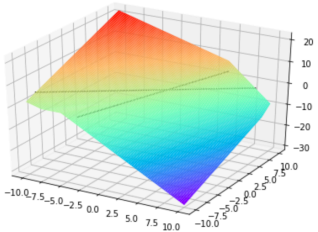


图 13: 一层的神经网络

深度神经网络的函数模型：连续分片线性函数的复合

- 要想获得对数据更好的表达能力，我们需要更复杂的函数 F。
- 构造一个更加复杂的 F 最好的方法是通过复合运算，从简单函数中创造复杂函数。
- 每个 F_i 都是对线性的 (或仿射的) 函数施加 ReLU，即 $F_i(\boldsymbol{x}) = (\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i)_+$ 是非线性的，它们的复合是 $F(\boldsymbol{x}) = \boldsymbol{C}F_L(F_{L-1}(\dots F_2(F_1(\boldsymbol{x}))))$，在最终输出层之前, 得到了 L 个隐藏层。随着 L 的增加, 网络将会变得更深。

例 23

考虑一个具有三个隐藏层的神经网络，其中

$$F_1 = \mathrm{ReLU}\left(\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}\boldsymbol{x} + \begin{pmatrix} 1 \\ -2 \end{pmatrix}\right),$$

$$F_2 = \mathrm{ReLU}\left(\begin{pmatrix} 1 & 2 \\ -2 & -3 \end{pmatrix}\boldsymbol{x} + \begin{pmatrix} -1 \\ 2 \end{pmatrix}\right),$$

$$F_3 = \mathrm{ReLU}\left(\begin{pmatrix} 2 & 4 \\ -2 & 3 \end{pmatrix}\boldsymbol{x} + \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right)$$

复合后得：

$$F(\boldsymbol{x}) = \begin{pmatrix} -1 & 1 \end{pmatrix} F_3(F_2(F_1(\boldsymbol{x}))),$$

其图像如右图所示。

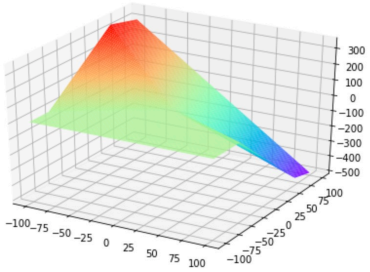


图 14: 三个隐藏层的神经网络

激活函数

激活函数

前面提到的 ReLU 函数在深度神经网络中被称为**激活函数**。激活函数通常是一类非线性函数，其满足以下性质：

- 连续并可导 (允许少数点上不可导)，
- 本身及其导数计算简单，
- 导函数的值域要在一个合适的区间内。

常见的激活函数：ReLU 型函数

ReLU 函数是目前最常用的激活函数，它有多种不同的变体。

- ReLU(Rectified Linear Unit，修正线性单元)：

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x)$$

- 带泄露的 ReLU(LeakyReLU)：

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & x > 0 \\ \gamma x & x \leq 0 \end{cases} = \max(0, x) + \gamma \min(0, x)$$

- 带参数的 ReLU(Parametric ReLU，PReLU)：

$$\mathrm{PReLU}_i(x) = \begin{cases} x & x > 0 \\ \gamma_i x & x \leq 0 \end{cases} = \max(0, x) + \gamma_i \min(0, x)$$

ReLU 型函数的性质

- 上面三种激活函数都是分片线性函数。所以如果一个神经网络中只用这类激活函数，那么最终得到的模型函数也是分片线性函数。
- 上面三种激活函数只需要进行加、乘和比较的操作，计算上非常高效。
- ReLU 函数被认为有生物上的解释性，比如单侧抑制、宽兴奋边界（即兴奋程度也可以非常高）。
- ReLU 函数的缺点是输出是非零中心化的，给后一层的神经网络引入偏置偏移，会影响梯度下降的效率。ReLU 神经元指采用 ReLU 作为激活函数的神经元。

ReLU 型函数的性质

- 此外，ReLU 神经元在训练时比较容易“死亡”。在训练时，如果参数在一次不恰当的更新后，第一个隐藏层中的某个 ReLU 神经元在所有的训练数据上都不能被激活，那么这个神经元自身参数的梯度永远都会是 0。
- 在实际使用中，为了避免上述情况，我们就可以使用 LeakyReLU 和 PReLU。
- LeakyReLU 在输入 $x < 0$ 时，保持一个很小的梯度 λ。这样当神经元非激活时也能有一个非零的梯度可以更新参数，避免永远不能被激活。
- 而 PReLU 则引入一个可学习的参数，可以使得不同神经元可以有不同的参数。

常见的激活函数：Sigmoid 型函数

- **Logistic 函数**，其具有形式：

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

- **Tanh 函数**，其具有形式：

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)},$$

Tanh 函数可以看作是放大并平移的 Logistic 函数，其值域为 $(-1, 1)$，它与 Logistic 函数满足如下关系：

$$\tanh(x) = 2\sigma(2x) - 1$$

Logistic 函数也叫 Sigmoid 函数，因此我们把它和 Tanh 函数统称为 Sigmoid 型函数。

Sigmoid 型函数的性质

定义 21

对于函数 $f(x)$，若 $x \to -\infty$ 时，其导数 $f'(x) \to 0$，则称其为左饱和。若 $x \to +\infty$ 时，其导数 $f'(x) \to 0$，则称其为右饱和。当同时满足左、右饱和时，就称为两端饱和。

定理 5

Sigmoid 型函数具有饱和性。

- Sigmoid 型激活函数会导致一个非稀疏的神经网络，但是 ReLU 具有很好的稀疏性。
- 相对于 ReLU，Logistic 有更好的光滑性，通常认为连续导数将有助于优化模型，这种想法是合理的，但它被证明是错误的，因为饱和性容易导致梯度消失。

Logistic 函数的性质

“挤压”函数

Logistic 函数可以看成是一个“挤压”函数，把一个实数域的输入“挤压”到 $(0,1)$。当输入值在 0 附近时，Sigmoid 型函数近似为线性函数；当输入值靠近两端时，对输入进行抑制。输入越小，越接近于 0；输入越大，越接近于 1。

因为 Logistic 函数的性质，使得装备了 Logistic 激活函数的神经元具有以下两点性质：

- 其输出直接可以看作是概率分布，使得神经网络可以更好地和统计学习模型进行结合。
- 其可以看作是一个软性门（Soft Gate），用来控制其他神经元输出信息的数量。

Sigmoid 型函数的近似

Logistic 函数和 Tanh 函数计算开销较大。因为这两个函数都是在中间（0 附近）近似线性，两端饱和，因此这两个函数可以通过分段函数来近似。

Logistic 函数的近似

因为 Logistic 函数的导数为 $\sigma'(x) = \sigma(x)(1-\sigma(x))$，所以 Logistic 函数在 0 附近的一阶泰勒展开（Taylor expansion）为

$$g_l(x) = \sigma(0) + x \times \sigma'(0) = 0.25x + 0.5$$

这样 Logistic 函数可以用分段函数 hard-logistic(x) 来近似

$$\text{hard-logistic}(x) = \begin{cases} 1 & g_l(x) \geq 1 \\ g_l(x) & 0 < g_l(x) < 1 \\ 0 & g_l(x) \leq 0 \end{cases}$$

$$= \max(\min(g_l(x), 1), 0) \qquad = \max(\min(0.25x + 0.5, 1), 0)$$

Sigmoid 型函数的近似

Tanh 函数的近似

同样，Tanh 函数在 0 附近的一阶泰勒展开为

$$g_t(x) = \tanh(0) + x \times \tanh'(0) = x$$

这样 Tanh 函数也可以用分段函数 hard-tanh(x) 来近似。

$$\text{hard-tanh}(x) = \max(\min(x, 1), -1)$$

其他一些激活函数

- ELU（Exponential Linear Unit，指数线性单元）：

$$\text{ELU}(x) = \begin{cases} x & x > 0 \\ \gamma(\exp(x) - 1) & x \leq 0 \end{cases} = \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

- Softplus 函数

$$\text{Softplus}(x) = \log(1 + \exp(x))$$

Softplus 函数其导数刚好是 Logistic 函数。Softplus 函数虽然也具有单侧抑制、宽兴奋边界的特性，却没有稀疏激活性。

- Swish 函数

$$\text{Swish}(x) = x\sigma(\beta x)$$

常见激活函数的图像

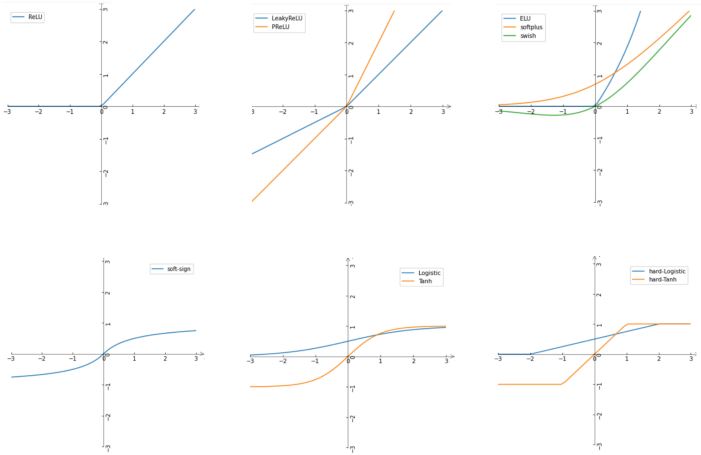


图 15: 激活函数一览

跨步和下采样

我们除了使用激活函数来捕获数据的非线性，有时我们还可以通过跨步和下采样来实现对数据降维。假设我们从长度为 128 的一维信号开始，我们希望过滤该信号，也即将向量乘以权重矩阵 $\boldsymbol{A}$，将长度减小为 64。为达到这个目标，可以通过：

1. 下采样过滤：将 128 维向量 $\boldsymbol{v}$ 乘以 $\boldsymbol{A}$，然后丢弃输出的奇数分量，这样输出为 $(\downarrow 2)\boldsymbol{A}\boldsymbol{v}$。
2. 跨步过滤：丢弃奇数行矩阵 $\boldsymbol{A}$，得到又短又宽的新矩阵 $\boldsymbol{A}_2$：64 行和 128 列。此时，过滤的“步幅”为 2。现在将 128 分量向量 $\boldsymbol{v}$ 乘以 $\boldsymbol{A}_2$，然后得到 $\boldsymbol{A}_2\boldsymbol{v}$，这和 $(\downarrow 2)\boldsymbol{A}\boldsymbol{v}$ 是一样的。如果步幅为 3，将在每 3 个分量中保留一个分量。

显然，跨步方法更为有效。如果步幅为 4，则将每一个维度总分量除以 4。在两个维度 (对于图像) 的数表中，尺寸将减小 16 倍。而下采样则清楚地表明，丢失了一半或者四分之三的信息。跨步和下采样能降低数据的维度，但是会丢失数据的重要信息。

池化

还有一种将维数从 128 减少到 64 的方法，但是却能减少破坏重要信息的风险：池化（pooling），它是某一种形式的下采样。

- 对于图像 (二维信号)，我们可以在每 2×2 平方像素上使用池化。那么在每个维度上元素减少一半，而对于图像来说总的像素减少 4 倍。从而导致隐藏层上神经元的数量会变为原来的 1/4，这可以加快训练速度。
- 池化使得尺寸减小，除了减少计算量，还减少了过拟合的可能性。池化函数会不断地减小数据的空间大小，因此参数的数量和计算量也会下降，这在一定程度上控制了过拟合。
- 池化包括最大池化和平均池化。

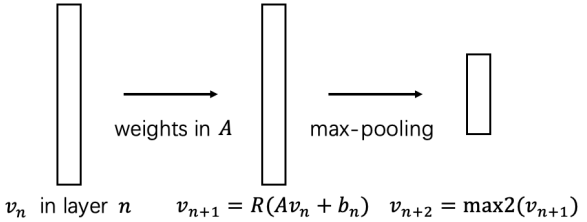


图 16: 最大池化

- 如果将向量 $\boldsymbol{v}$ 乘以 $\boldsymbol{A}$，然后进行下采样时，是从每若干个元素取出其中的最大值作为结果的方式叫做最大池化。
- 注意：最大合并简单而又快速，但不是线性操作，这是减少尺寸，纯粹和简单的明智方法。

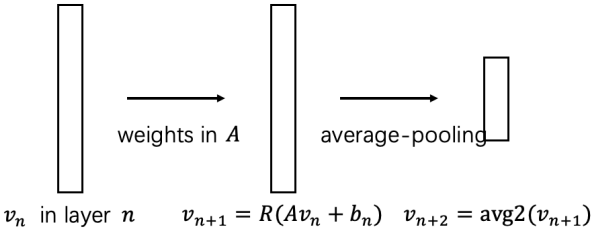


图 17: 平均池化

- 如果将向量 $\boldsymbol{v}$ 乘以 $\boldsymbol{A}$，然后进行下采样时，是将每若干个元素的平均值作为结果的方式叫做平均池化。
- 平均池化将保持每个池中的平均值，并且平均池化是线性的。

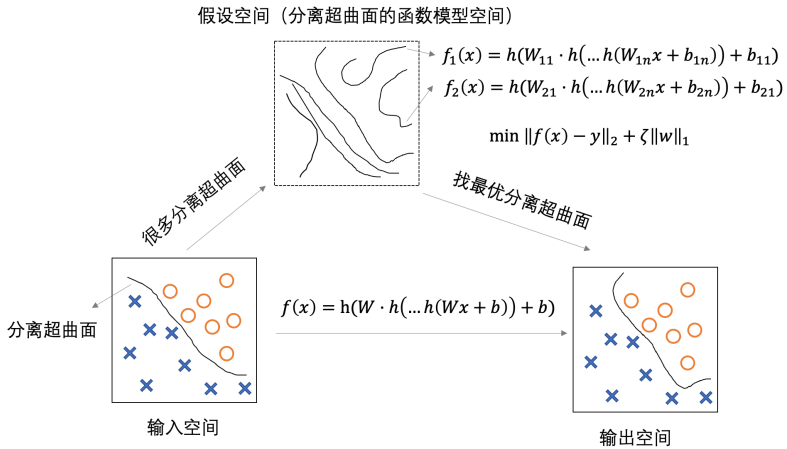
深度神经网络中函数的一般构造方式

最后我们简单总结一下一个一般的神经网络构造方式：

- 记神经网络中每一层的函数为 $F_1, F_2, F_3, \ldots, F_n$，权重 $\boldsymbol{W}$ 是连接各层，并且将在训练 F 的时候被更新。向量 $\boldsymbol{x} = \boldsymbol{x}_0$ 来自训练集，函数 F_k 在第 k 层产生了向量 $\boldsymbol{x}_k$。
- 通常 F_k 由两部分组成，首先是线性部分，比如 $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ 或者卷积，然后再通过激活函数作用变成一个非线性函数。
- 神经网络中最核心的操作就是函数的复合，我们最终得到的模型 F 就是一系列函数的复合 $F(\boldsymbol{x}) = F_n(\ldots F_2(F_1(\boldsymbol{x})))$。

神经网络模型是一个非线性模型，其中非线性主要是通过激活函数来提供。在训练神经网络过程中，我们通常使用随机梯度下降。为了做到这一点，我们就需要链式法则和对向量函数或者矩阵函数求梯度，我们将在下一讲详细讲述。

从空间的角度来理解神经网络中的函数关系



本讲小结

机器学习中函数：模型函数、损失函数、目标函数。

（线性和非线性）向量和矩阵函数

- 标量值函数
- 向量值函数
- 矩阵值函数
- 算子
- 泛函：经验风险泛函

机器学习中的非概率型函数

- 线性模型和感知机模型中的函数
- 支持向量机中的函数：间隔函数、核函数
- 主成分分析和 k 均值聚类的函数
- 深度学习中的函数：仿射函数、激活函数、跨步和下采样、池化函数

函数所在的空间：各种未知的假设空间。函数的性质：连续、可微、凸性等。