第 28 章 神经网络

28.1 动机

命题 28.1.1 (感知器线性聚合的图示视角)

将 T 个感知器 $\{g_t\}_{t=1}^{T}$ 以线性方式聚合，可写成两层权重与两层符号函数的网络：

1. 第一层：每个 g_t 为

$$g_t(x) = \operatorname{sign}(w_t^\top x),$$

其中 $x_0 \equiv 1$（偏置项），$w_t \in \mathbb{R}^{d+1}$ 为第 t 个感知器的权重。

2. 第二层：线性聚合

$$G(x) = \operatorname{sign}\Big(\sum_{t=1}^{T} \alpha_t \, g_t(x)\Big),$$

其中 α_t 为第二层的投票权重。

结论　两层权重 $\{w_t\}$ 与 $\{\alpha_t\}$，以及两层符号函数 $\operatorname{sign}(\cdot)$，共同构成“感知器线性聚合”的紧凑网络结构。

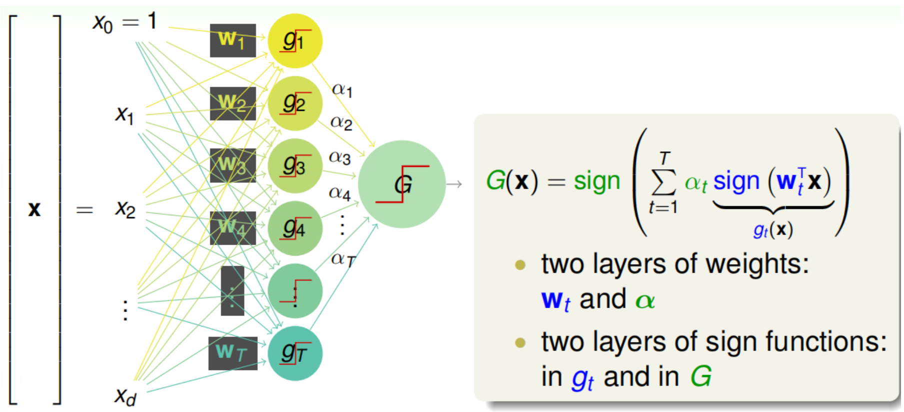

图 28.1.1: 单层神经网络示意图

命题 28.1.2 (感知器的强大力量与局限性)

考虑感知器的线性组合，其表达能力取决于基函数的数量和复杂性。

1. 强大力量

- 使用 8 个感知器可以实现复杂的决策边界。
- 使用 16 个感知器可以实现更平滑的决策边界。
- 对于“凸集”假设，感知器的 VC 维是有限的，但随着感知器数量的增加，可以逼近任意复杂的边界。

2. 局限性

- XOR 问题不能通过简单的线性组合解决。

$$\mathrm{XOR}(g_1, g_2) = \mathrm{OR}(\mathrm{AND}(-g_1, g_2), \mathrm{AND}(g_1, -g_2))$$

这表明 XOR 问题在特征空间 $\Phi(x) = (g_1(x), g_2(x))$ 下不是线性可分的。

结论　感知器的线性组合虽然强大，但不能解决所有问题。对于非线性可分问题，需要更复杂的特征变换或更深层次的网络结构。

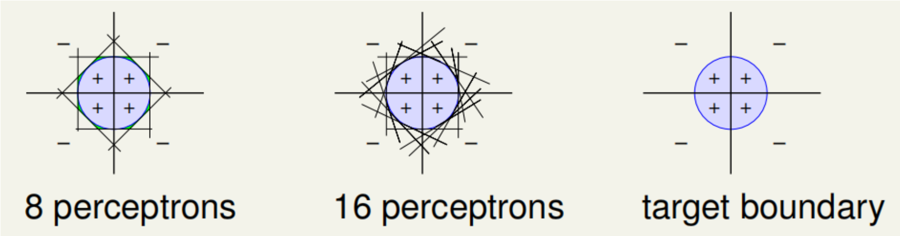

图 28.1.2: 感知机实现复杂决策边界

命题 28.1.3 (逻辑运算的线性聚合实现)

利用感知器的线性聚合，可精确实现任意布尔函数。

1. AND 门示例

设两输入感知器

$$g_1(x) = \operatorname{sign}(w_1^\top x), \quad g_2(x) = \operatorname{sign}(w_2^\top x).$$

构造

$$G(x) = \operatorname{sign}\big(-1 + g_1(x) + g_2(x)\big),$$

则

$$G(x) = +1 \iff g_1(x) = g_2(x) = +1 \text{ (TRUE)}; \quad \text{否则} G(x) = -1 \text{ (FALSE)}.$$

2. OR、**NOT** 的类似实现

$$\mathrm{OR}(g_1, g_2) = \operatorname{sign}\big(+1 + g_1(x) + g_2(x)\big),$$
$$\mathrm{NOT}(g) = \operatorname{sign}\big(-g(x)\big).$$

3. XOR 的层级实现

$$\mathrm{XOR}(g_1, g_2) = \mathrm{OR}\big(\mathrm{AND}(-g_1, g_2), \mathrm{AND}(g_1, -g_2)\big),$$

需再加一层 **AND+OR** 变换。

结论　单层感知器 $\to$ 线性聚合 $\to$ 多层感知器（MLP），逐步增强表达能力，可解决非线性可分问题。

例题 28.1 选择题：AdaBoost 中实现逻辑或（OR）的权重组合

已知基学习器 $g_0(\mathbf{x}) = +1$，需找到权重 $(\alpha_0, \alpha_1, \alpha_2)$，使得集成模型 $G(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=0}^{2} \alpha_t g_t(\mathbf{x})\right)$ 实现逻辑或运算 $\mathrm{OR}(g_1, g_2)$（即 g_1 或 g_2 为 $+1$ 时 G 输出 $+1$，仅当 g_1, g_2 全为 -1 时输出 -1）。

1) $(-3, +1, +1)$
2) $(-1, +1, +1)$
3) $(+1, +1, +1)$
4) $(+3, +1, +1)$

解答　正确选项为 $\boxed{3}$。逻辑或要求：全假（$g_1, g_2 = -1$）时 $G = -1$，否则 $G = +1$。

验证选项 3 $(\alpha_0, \alpha_1, \alpha_2) = (+1, +1, +1)$：

- 全假场景：$1 \cdot 1 + 1 \cdot (-1) + 1 \cdot (-1) = -1 < 0$，$G = -1$（符合）。
- 有真场景（如 $g_2 = +1$）：$1 \cdot 1 + 1 \cdot (-1) + 1 \cdot (+1) = 1 > 0$，$G = +1$（符合）。

综上，选项 3 满足逻辑或规则。 ■

28.2 神经网络假设

命题 28.2.1 (神经网络假设：特征变换)

神经网络通过逐层变换实现复杂的特征映射，其核心在于选择合适的变换函数。

1. 变换函数的选择

- 线性变换：整个网络退化为线性模型，表达能力受限。
- 离散变换（如符号函数）：优化困难，难以对权重 w 进行梯度下降。
- 连续变换（如 tanh）：

$$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} = 2\sigma(2s) - 1,$$

其中 $\sigma(s)$ 为 Sigmoid 函数。tanh 函数是符号函数的连续近似，优化更易实现。

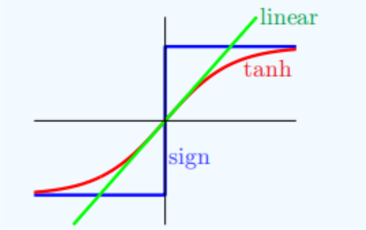

图 28.2.1: 连续变换函数

2. 神经网络结构

神经网络由多层组成，每层的输出是下一层的输入。

$$\begin{aligned}
&\text{输入层：} x^{(0)} = x, \\
&\text{隐藏层：} x^{(l)} = \tanh\big(W^{(l)} x^{(l-1)}\big), \quad l = 1, \dots, L-1, \\
&\text{输出层：} G(x) = \tanh\big(W^{(L)} x^{(L-1)}\big).
\end{aligned}$$

结论　神经网络通过逐层变换实现复杂的特征映射，tanh 等连续变换函数因其优化友好性而被广泛使用。 ♠

命题 28.2.2 (神经网络的物理解释)

每一隐藏层都可视为对输入 x 的“模式匹配”变换：

- 第 ℓ 层输出

$$\Phi^{(\ell)}(x) = \tanh\big(W^{(\ell)} \Phi^{(\ell-1)}(x)\big), \quad \Phi^{(0)}(x) = x,$$

其中 $W^{(\ell)} \in \mathbb{R}^{d^{(\ell)} \times d^{(\ell-1)}}$ 的每一行对应一条“权重模式”。
- $\tanh(\cdot)$ 的符号与大小反映 x 与该模式的匹配程度：正值表示“匹配”，负值表示“不匹配”，绝对值越大越显著。

整体视角　神经网络通过多层连接权重逐层提取并组合局部模式，最终实现复杂决策边界。 ♠

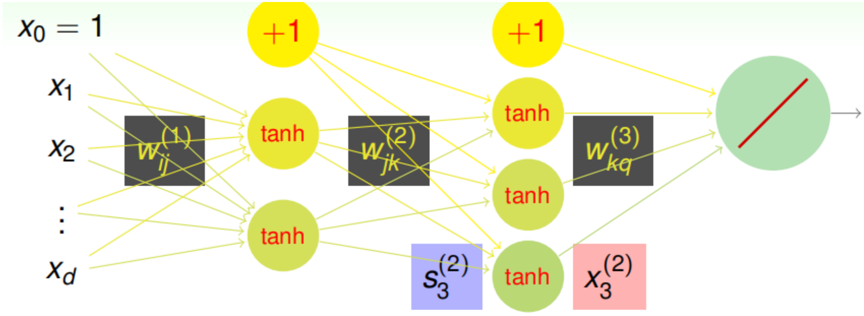

图 28.2.2: 多层神经网络示意图

例题 28.2 选择题：3 - 5 - 1 神经网络的权重数量

一个 3 - 5 - 1 神经网络（3 - 5 - 1 NNet ）中，权重 $\{w_{ij}^{(\ell)}\}$ 的数量是多少？

1) 9
2) 15
3) 20
4) 26

解答 正确选项为 $\boxed{4}$。“3 - 5 - 1 神经网络”指输入层 3 个神经元、隐藏层 5 个神经元、输出层 1 个神经元的网络结构。

计算权重数量需考虑层间连接：

- 输入层 → 隐藏层：3×5=15 个连接权重 + 5 个偏置 = 20 个
- 隐藏层 → 输出层：5×1=5 个连接权重 + 1 个偏置 = 6 个

总权重数：20+6=26 个。 ■

28.3 神经网络学习

命题 28.3.1 (神经网络权重学习：梯度计算与反向传播)

1. 网络结构与符号适配

设多层前馈神经网络，遵循以下符号与传播规则：

- 输入层：$\boldsymbol{x} = [x_0, x_1, \dots, x_d]^\top$（其中 $x_0 = 1$ 为偏置输入）。
- 层间传播：第 ℓ 层第 j 个神经元的加权输入为 $s_j^{(\ell)}$，经 tanh 激活后输出 $x_j^{(\ell)} = \tanh(s_j^{(\ell)})$；第 ℓ 层到 $\ell+1$ 层的权重记为 $w_{jk}^{(\ell+1)}$（即第 ℓ 层第 j 个神经元连接到 $\ell+1$ 层第 k 个神经元的权重）。

以含隐藏层的网络为例，前向传播关系可表示为：

$$s_j^{(\ell)} = \sum_{i=0}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)}, \quad x_j^{(\ell)} = \tanh(s_j^{(\ell)})$$

输出层误差定义为平方误差：

$$e_n = \left(y_n - \mathrm{NNet}(\boldsymbol{x}_n)\right)^2$$

2. 输出层梯度推导

对输出层（记总层数为 L，输出层为第 L 层），第 1 个神经元的加权输入满足：

$$s_1^{(L)} = \sum_{i=0}^{d^{(L-1)}} w_{i1}^{(L)} x_i^{(L-1)}$$

根据链式法则，输出层权重 $w_{i1}^{(L)}$ 的梯度为：

$$\frac{\partial e_n}{\partial w_{i1}^{(L)}} = \frac{\partial e_n}{\partial s_1^{(L)}} \cdot \frac{\partial s_1^{(L)}}{\partial w_{i1}^{(L)}} = -2\big(y_n - s_1^{(L)}\big) \cdot x_i^{(L-1)}$$

定义输出层误差项 $\delta_1^{(L)} = -2\big(y_n - s_1^{(L)}\big)$，则梯度可简化为：

$$\frac{\partial e_n}{\partial w_{i1}^{(L)}} = \delta_1^{(L)} \cdot x_i^{(L-1)}$$

3. 隐藏层梯度与反向传播递推

对一般层 ℓ（$1 \leq \ell < L$），第 j 个神经元的加权输入为 $s_j^{(\ell)}$，激活输出为 $x_j^{(\ell)} = \tanh\big(s_j^{(\ell)}\big)$。

定义误差项 $\delta_j^{(\ell)} = \frac{\partial e_n}{\partial s_j^{(\ell)}}$，通过链式法则推导递推关系：

由于第 ℓ 层激活 $x_j^{(\ell)}$ 会参与第 $\ell+1$ 层多个神经元的加权输入（即 $s_k^{(\ell+1)} = \sum_j w_{jk}^{(\ell+1)} x_j^{(\ell)}$），因此：

$$\delta_j^{(\ell)} = \frac{\partial e_n}{\partial s_j^{(\ell)}} = \sum_{k=1}^{d^{(\ell+1)}} \frac{\partial e_n}{\partial s_k^{(\ell+1)}} \cdot \frac{\partial s_k^{(\ell+1)}}{\partial x_j^{(\ell)}} \cdot \frac{\partial x_j^{(\ell)}}{\partial s_j^{(\ell)}}$$

代入激活函数导数 $\frac{\partial x_j^{(\ell)}}{\partial s_j^{(\ell)}} = \tanh'\big(s_j^{(\ell)}\big)$，以及权重关系 $\frac{\partial s_k^{(\ell+1)}}{\partial x_j^{(\ell)}} = w_{jk}^{(\ell+1)}$，最终得反向传播递推公式：

$$\delta_j^{(\ell)} = \left(\sum_k \delta_k^{(\ell+1)} w_{jk}^{(\ell+1)}\right) \cdot \tanh'\big(s_j^{(\ell)}\big)$$

利用 $\delta_j^{(\ell)}$，第 ℓ 层第 j 个神经元对第 i 个输入权重 $w_{ji}^{(\ell)}$ 的梯度为（输入来自前层激活 $x_i^{(\ell-1)}$）：

$$\frac{\partial e_n}{\partial w_{ji}^{(\ell)}} = \delta_j^{(\ell)} \cdot x_i^{(\ell-1)}$$

4. 计算流程与算法核心

神经网络权重学习遵循以下流程：

1. 前向传播：从输入层开始，逐层计算 $s_j^{(\ell)}$ 和 $x_j^{(\ell)}$，直至输出层得到预测值并计算误差 e_n。
2. 反向传播：从输出层出发，利用上述递推公式反向逐层计算 $\delta_j^{(\ell)}$，结合前层激活 $x_i^{(\ell-1)}$ 求解所有权重梯度 $\frac{\partial e_n}{\partial w_{ji}^{(\ell)}}$。
3. 权重更新：通过（随机）梯度下降（SGD），依据梯度 $\frac{\partial e_n}{\partial w_{ji}^{(\ell)}}$ 迭代更新权重 $w_{ji}^{(\ell)}$，逐步最小化训练误差 E_{in}。

算法 28.3.1: 反向传播（Backpropagation）算法

输入: 训练集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$；网络结构层数及各层维度 $d^{(0)}, \ldots, d^{(L)}$

输出: 训练完成的神经网络 $g_{\mathrm{NN}}(\boldsymbol{x})$

初始化阶段

随机初始化神经网络各层权重 $w_{ij}^{(\ell)}$，其中 ℓ 表示网络层数（$\ell = 1, \ldots, L$），i、j 为对应层神经元索引。

for $t = 0, 1, \ldots, T$ **do**

随机样本选取: 从训练集里随机挑选一个样本 n，即 $n \in \{1, 2, \cdots, N\}$（支持 mini-batch 时可多选）;

前向传播计算

以选中样本 $\boldsymbol{x}^{(0)} = x_n$ 为输入，逐层计算神经元激活值：对第 ℓ 层（$\ell = 1, \ldots, L$），先算线性组合

$$s_j^{(\ell)} = \sum_i w_{ij}^{(\ell)}\, x_i^{(\ell-1)}$$

再通过 tanh 激活函数得到激活值

$$x_j^{(\ell)} = \tanh\bigl(s_j^{(\ell)}\bigr)$$

反向传播：误差梯度计算

从输出层（$\ell = L$）开始，反向推导各层误差项 $\delta_j^{(\ell)}$：

- 输出层误差：结合真实标签 y_n 与激活函数导数：

$$\delta_j^{(L)} = \bigl(y_{nj} - x_j^{(L)}\bigr) \odot \tanh'\bigl(s_j^{(L)}\bigr)$$

- 隐藏层误差（$\ell = L-1, \ldots, 1$）：利用上层误差与权重，反向传递计算：

$$\delta_j^{(\ell)} = \left(\sum_k w_{jk}^{(\ell+1)}\, \delta_k^{(\ell+1)}\right) \odot \tanh'\bigl(s_j^{(\ell)}\bigr)$$

权重更新：梯度下降

依据误差项，用梯度下降规则更新各层权重，学习率 η 控制更新步长：

$$w_{ij}^{(\ell)} \leftarrow w_{ij}^{(\ell)} - \eta\, x_i^{(\ell-1)}\, \delta_j^{(\ell)}$$

输出神经网络模型

训练后，神经网络可表示为多层 tanh 激活嵌套的形式（以两层隐藏逻辑示例，适配实际网络深度）:

$$g_{\mathrm{NN}}(\boldsymbol{x}) = \tanh\left(\sum_j w_{jk}^{(2)} \tanh\left(\sum_i w_{ij}^{(1)}\, x_i\right)\right)$$

注 实际训练中，常采用 **mini-batch** 策略：多次并行执行“样本选取 - 前向传播 - 反向传播”流程，将多组样本的梯度平均后再更新权重，平衡计算效率与收敛稳定性。反向传播核心价值是高效推导梯度，让权重更新精准指向“降低预测误差”方向，支撑神经网络学习复杂映射关系。

28.4 优化与正则化

命题 28.4.1 (神经网络的优化特性)

当网络包含多个隐藏层时，目标函数

$$\mathrm{err}(W) = \sum_{n=1}^{N} \ell(y_n, \mathrm{NNet}(x_n; W))$$

整体非凸，故难以保证全局最优。

关键现象

- GD/SGD 仅收敛到局部极小，其位置对初始权重敏感；
- 权重过大 $\to$ tanh 饱和 $\to$ 梯度趋零，训练停滞；
- 实用建议：采用随机小权重初始化并多组尝试。

结论　神经网络优化困难，但在实践中通过随机初始化与足够迭代通常可获良好局部解。

命题 28.4.2 (神经网络模型的 VC 维)

对采用 tanh 类激活函数的前馈网络，其 VC 维大致满足

$$d_{\mathrm{vc}} = \mathcal{O}(VD)$$

其中 V 为神经元总数，D 为权重总数。

解读

- 优势：若 V 足够大，网络可逼近任意函数；
- 劣势：过大的 V 使 d_{vc} 急剧上升，易导致过拟合。

实践提示　使用神经网络时务必监控模型复杂度，防止因 V 过大而失控地过拟合。

命题 28.4.3 (神经网络的两种常用正则化)

1. 权重正则化

- **L2**（权重衰减）

$$\Omega_{\mathrm{L2}}(w) = \lambda \|w\|_2^2 \quad \Longrightarrow \quad \text{大权重} \to \text{大收缩，小权重} \to \text{小收缩。}$$

- **L1**（稀疏化）

$$\Omega_{\mathrm{L1}}(w) = \lambda \|w\|_1 \quad \Longrightarrow \quad \text{部分权重直接置零，有效降低} d_{\mathrm{vc}}.$$

- 权重消除 **weight-elimination**（**Scaled L2**）

$$\Omega_{\mathrm{WE}}(w) = \lambda \sum_i \frac{w_i^2}{1 + w_i^2} \quad \Longrightarrow \quad \text{大权重} \to \text{中等收缩，小权重} \to \text{中等收缩。}$$

2. 早停（**Early Stopping**）

随着迭代次数 t 增加：

- 训练误差 E_{in} 单调下降；
- 测试误差 E_{test} 先降后升，呈“过拟合”拐点；
- 较小 t 等价于“较低 VC 维”模型。

策略　在验证集监控 E_{test}，选择其最低点的迭代次数 t^* 停止训练，从而有效防止过拟合。

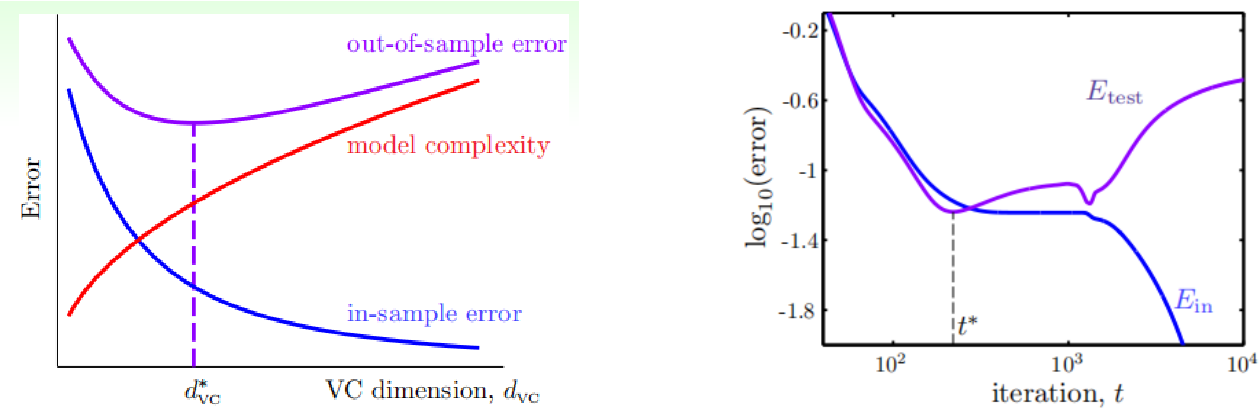

图 28.4.1: 早停（Early Stopping）示意图

28.5 总结

笔记 [神经网络]

- 动机：借鉴生物神经结构，通过多层非线性映射获得强大表达能力。
- 神经网络假设：逐层提取模式，最终输出线性可分的表示。
- 神经网络学习：反向传播高效计算梯度。
- 优化与正则化：初始化技巧、正则项及早停等实用策略。