神经网络基础教程

2025 年 9 月 28 日

目录

1　人工神经元模型（感知机）

感知机是人工神经网络中最基本的神经元模型。给定输入向量 $\mathbf{x} = [x_1, \ldots, x_d]^\top$、权重向量 $\mathbf{w} = [w_1, \ldots, w_d]^\top$ 以及偏置 b，感知机先计算线性组合，再通过符号函数得到二分类输出：

$$y = \operatorname{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right), \tag{1}$$

其中 $\operatorname{sign}(z)$ 在 $z \geq 0$ 时输出 1，否则输出 -1。等式 $\mathbf{w}^\top \mathbf{x} + b = 0$ 描述的超平面将输入空间划分成两个类别。

1.1　学习规则

经典的感知机学习算法在样本 $(\mathbf{x}, t)$（$t \in \{-1, 1\}$）被误分类时才会更新参数：

$$\mathbf{w} \leftarrow \mathbf{w} + \eta t \mathbf{x}, \quad b \leftarrow b + \eta t, \tag{2}$$

其中 $\eta > 0$ 为学习率。该更新会推动决策边界向正确分类方向移动。当数据线性可分时，算法保证在有限步内收敛。

1.2　几何直观

图 **??** 展示了感知机的决策边界以及样本到超平面的符号距离，有助于理解分类几何结构。

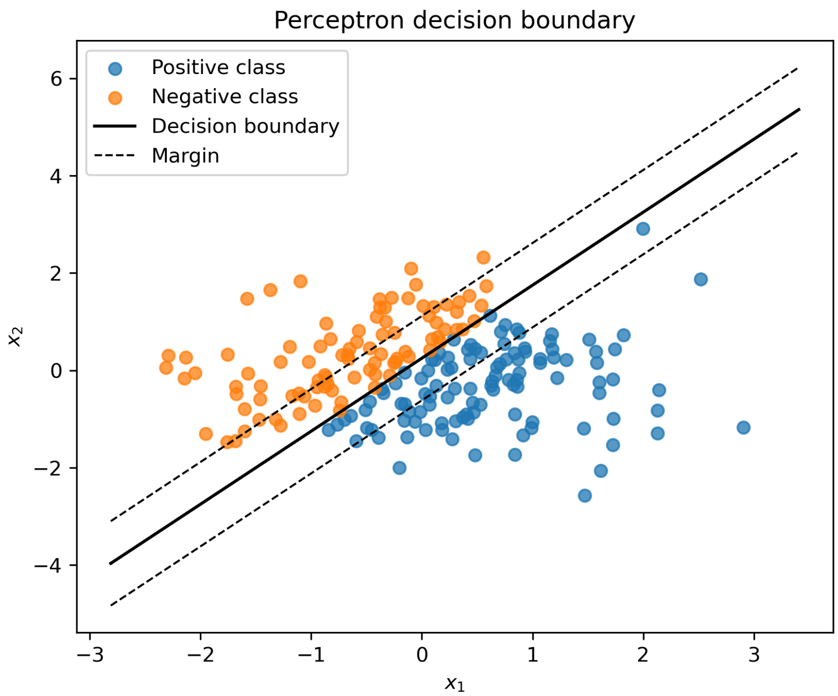

图 1: 感知机决策边界及其间隔示意。

2 多层感知机（MLP）与前向传播

多层感知机通过堆叠多层神经元并引入非线性激活函数，能够逼近复杂的多维映射。设网络共有 L 层，其前向计算为

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \qquad \mathbf{h}^{(1)} = \phi^{(1)}\big(\mathbf{a}^{(1)}\big), \tag{3}$$

$$\mathbf{a}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \qquad \mathbf{h}^{(\ell)} = \phi^{(\ell)}\big(\mathbf{a}^{(\ell)}\big), \tag{4}$$

$$\hat{\mathbf{y}} = \mathbf{h}^{(L)}. \tag{5}$$

其中 $\phi^{(\ell)}$ 表示第 ℓ 层的逐元素激活函数。

2.1 前向传播算法

前向传播沿层级顺序依次计算线性变换与非线性映射，以下 Python 代码给出了简洁实现：

Listing 1: 全连接 MLP 的前向传播示例。

```
import numpy as np

def forward_pass(weights, biases, activations, x):
    h = x
    for W, b, act in zip(weights, biases, activations):
        a = W @ h + b
```

```
        h = act(a)
    return h
```

2.2 表达能力

通用逼近定理指出，具有有限神经元的一隐藏层前馈网络在连续激活函数下，可以在紧致域上逼近任意连续函数。更深的网络通过复用中间特征，通常能以更少参数获得同等表示能力。

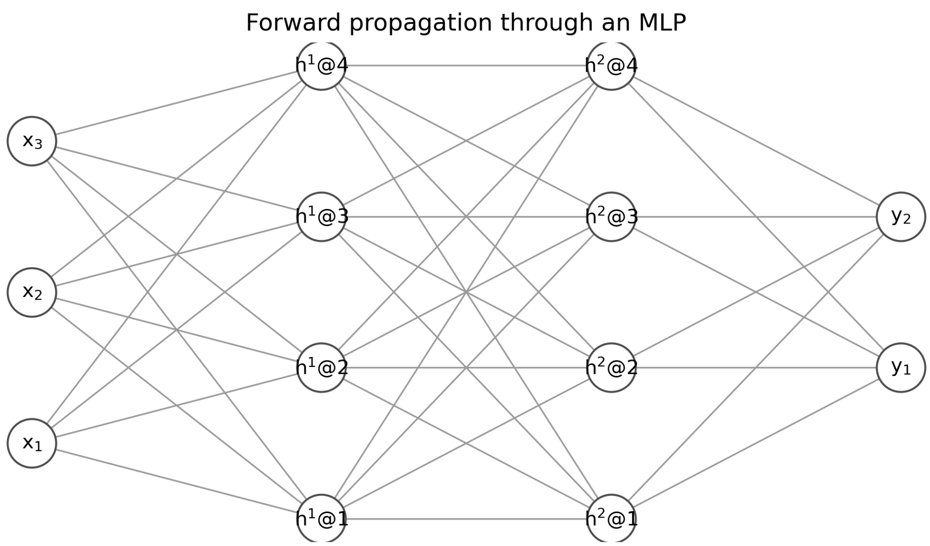

图 2: MLP 的前向传播流程，线性变换与非线性激活交替进行。

3 激活函数

激活函数为网络提供非线性能力，也影响梯度传播特性。图 ?? 比较了常见激活函数的曲线形状。

3.1 Sigmoid

Logistic Sigmoid 将实数压缩到 $(0, 1)$：

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma'(z) = \sigma(z)\big(1-\sigma(z)\big). \tag{6}$$

Sigmoid 在 $|z|$ 大时趋于饱和，可能导致梯度消失。

3.2 Tanh

双曲正切函数为零中心分布：

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \frac{d}{dz}\tanh(z) = 1 - \tanh^2(z). \tag{7}$$

相较 Sigmoid，其输出范围扩大至 $(-1, 1)$，更利于梯度传播。

3.3 ReLU

修正线性单元定义为

$$\mathrm{ReLU}(z) = \max(0, z), \quad \mathrm{ReLU}'(z) = \begin{cases} 1, & z > 0, \\ 0, & z < 0. \end{cases} \tag{8}$$

ReLU 易于优化，但若神经元长期处于负半轴，会出现 “死亡 ReLU” 问题。

3.4 Leaky ReLU

Leaky ReLU 在负半轴保留小斜率以缓解死亡问题：

$$\mathrm{LeakyReLU}(z) = \begin{cases} z, & z \geq 0, \\ \alpha z, & z < 0, \end{cases} \tag{9}$$

常取 $\alpha \approx 0.01$。

3.5 GELU

GELU 使用高斯累积分布函数对输入加权：

$$\mathrm{GELU}(z) = z\Phi(z) = \frac{z}{2}\left[1 + \mathrm{erf}\left(\frac{z}{\sqrt{2}}\right)\right], \tag{10}$$

其中 Φ 是标准正态分布的累积分布函数，erf 为误差函数。GELU 的平滑性质在 Transformer 中表现突出。

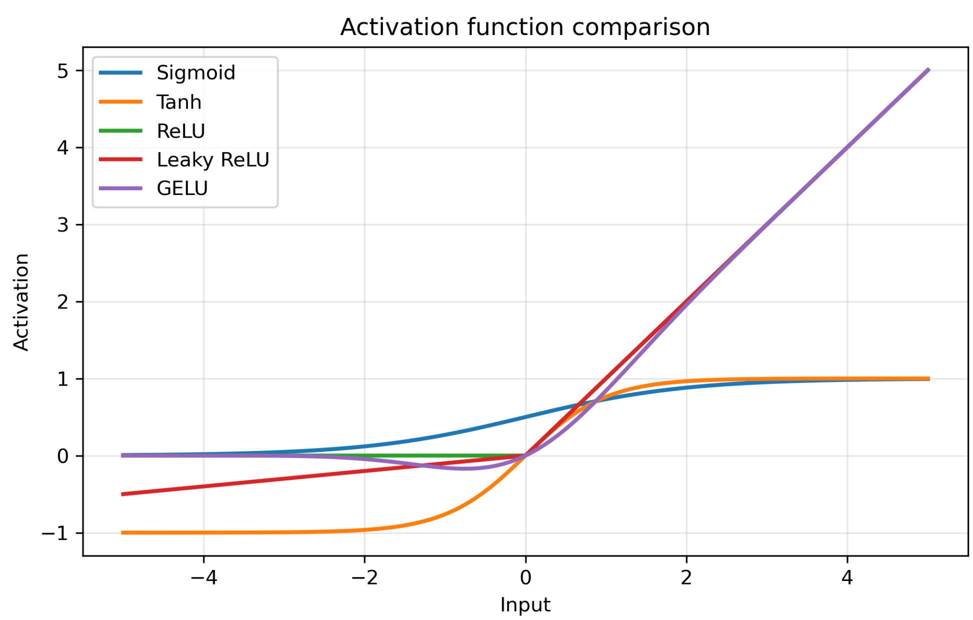

图 3: 常见激活函数曲线对比。

4 损失函数

损失函数衡量模型预测与真实标签之间的差异，是梯度驱动优化的目标。

4.1 均方误差（MSE）

对于回归任务，目标 t_i 与预测 $\hat{y}_i$ 的均方误差为

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - t_i\right)^2, \tag{11}$$

其关于 $\hat{y}_i$ 的梯度为 $\frac{2}{N}(\hat{y}_i - t_i)$。

4.2 交叉熵

二分类的交叉熵损失使用概率输出 p_i：

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[t_i \log p_i + (1-t_i)\log(1-p_i)\right]. \tag{12}$$

对于多分类问题，若 softmax 输出为 $p_{i,k}$，目标 $t_{i,k}$ 为 one-hot 编码，则

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{i,k}\log p_{i,k}. \tag{13}$$

4.3 Huber 损失

Huber 损失兼具 L1 与 L2 的优点，对异常值更稳健：

$$\mathcal{L}_{\delta}(r) = \begin{cases} \frac{1}{2}r^2, & |r| \le \delta, \\ \delta(|r| - \frac{1}{2}\delta), & |r| > \delta, \end{cases} \tag{14}$$

其中 $r = \hat{y} - t$，δ 控制 L2 与 L1 的切换位置。

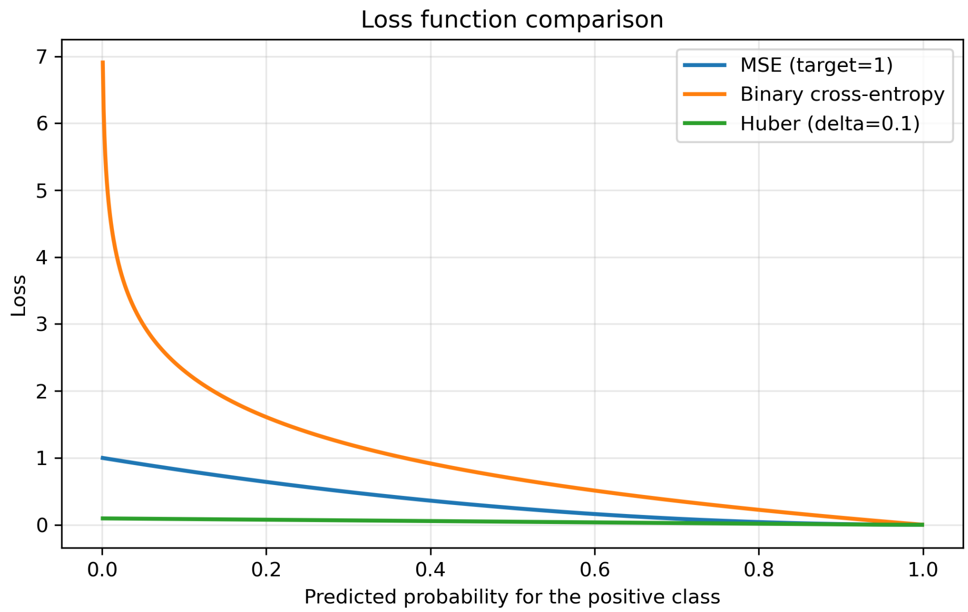

图 4: MSE、二元交叉熵及 Huber 损失的曲线形状比较。

5 实践建议

- **参数初始化：**使用 Xavier 或 He 初始化可避免激活值爆炸或消失。
- **归一化：**批归一化或层归一化帮助稳定不同层的分布。
- **优化方法：** Adam、RMSprop 等自适应算法能动态调整学习率。
- **正则化：** Dropout、权重衰减、提前停止等方法可缓解过拟合。