神经网络

汪小圈

2025-03-24

内容安排

- 神经网络基础
  - 浅层神经网络 (Shallow Neural Networks)
  - 深层神经网络 (Deep Neural Networks)
  - 损失函数 (Loss Functions)
  - 模型拟合 (Model Fitting)
  - 梯度下降与优化 (Gradients & Optimization)
  - 参数初始化 (Initialization)
  - 性能评估 (Performance)
  - 正则化方法 (Regularization)
- 深度学习简介
  - 深度学习基本概念
  - 常见深度学习架构

回顾：集成学习的核心思想

- **集成学习**是一种将**多个弱学习器 (Weak Learner) 组合成一个强学习器 (Strong Learner) 的技术**
- **核心思想：“三个臭皮匠，顶个诸葛亮”**
  - 组合多个弱学习器的预测结果，获得更全面、更鲁棒的预测能力
- **降低误差的方式**：
  - **降低方差**：通过并行训练多个基学习器，对结果平均或投票（如 Bagging）
  - **降低偏差**：通过串行训练基学习器，每个学习器纠正前一个的错误（如 Boosting）
  - **提高鲁棒性**：对异常值和噪声数据具有更强的抵抗力

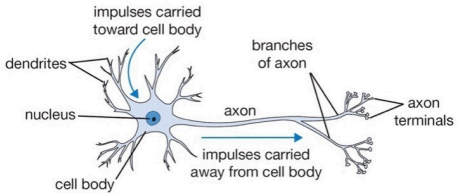
回顾：集成学习主要方法

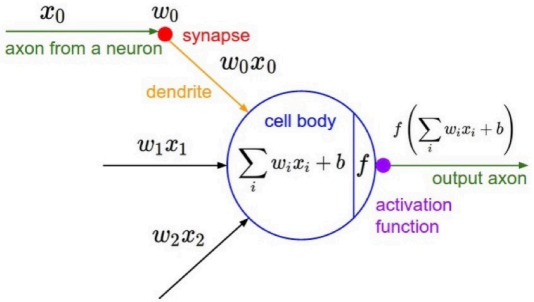
- **Bagging (Bootstrap Aggregating)**
  - 并行集成，通过自助采样创建多个训练数据集
  - 典型代表：随机森林
- **Boosting (提升法)**
  - 串行集成，每个新的基学习器都试图纠正前一个的错误
  - 典型代表：AdaBoost、GBDT
- **Stacking (堆叠法)**
  - 层次集成，使用另一个学习器组合基学习器的输出
  - 在各种机器学习竞赛中广泛应用

神经网络简介 (Neural Network)

- **神经网络**是一种模拟生物神经系统的**计算模型**
- **基本结构**:
  - 由大量相互连接的神经元组成
  - 能学习复杂的非线性关系
  - 具有强大的模式识别能力
- **历史发展**:
  - 1943 年: McCulloch 和 Pitts 提出第一个神经元模型
  - 1958 年: Rosenblatt 提出感知机
  - 1986 年: 反向传播算法提出
  - 2006 年: 深度学习兴起
  - 2012 年: AlexNet 赢得 ImageNet 挑战赛，标志深度学习的突破

神经元图示

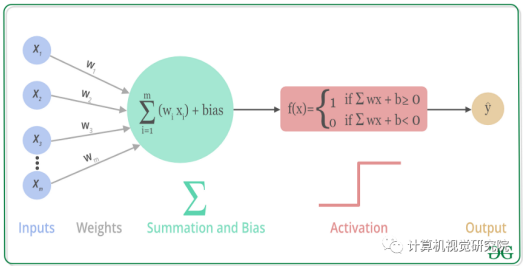




Non-linearity!

浅层神经网络 - 感知机模型

- **感知机 (Perceptron)** 是最基本的神经元模型
- **单个神经元组成**:
  - **输入 (x , x , ..., x)**: 特征向量
  - **权重 (w , w , ..., w)**: 特征重要性
  - **偏置 (b)**: 调整激活阈值
  - **加权求和**: $z = \sum_{i=1}^{n} w_i x_i + b$
  - **激活函数**: $a = \sigma(z)$
  - **输出**: $\hat{y} = a$



浅层神经网络 - 单层感知机局限性

- **线性可分性限制**:
  - 单层感知机只能解决线性可分问题
  - 无法解决 XOR 等非线性问题
- **解决方案**:
  - 引入多层结构
  - 使用非线性激活函数

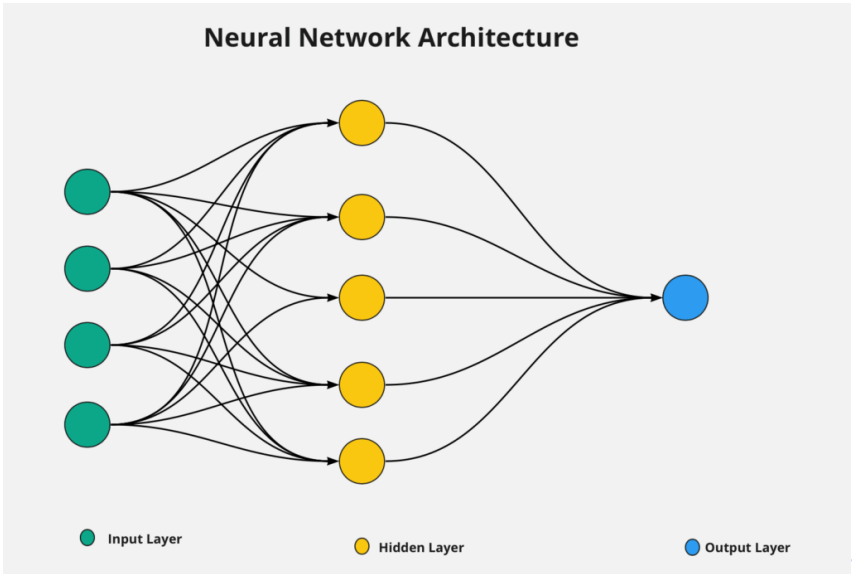
浅层神经网络 - 激活函数

- **激活函数**引入非线性，是神经网络强大表达能力的关键
- **常用激活函数**：
  - **Sigmoid**：$\sigma(z) = \frac{1}{1+e^{-z}}$
    - 输出范围 (0,1)，适合二分类
    - 缺点：存在梯度消失问题
  - **Tanh**：$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
    - 输出范围 (-1,1)，零中心化
    - 仍存在梯度消失问题
  - **ReLU**：$\mathrm{ReLU}(z) = \max(0, z)$
    - 计算简单，缓解梯度消失
    - 可能出现” 神经元死亡” 问题

浅层神经网络 - 多层感知机

- **多层感知机 (MLP)** 是浅层神经网络的典型代表
- **基本结构**:
  - **输入层**：接收原始特征
  - **隐藏层**：通常 1-2 层，提取特征
  - **输出层**：产生预测结果
- **前向传播**：信息从输入层流向输出层
  - $\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$
  - $\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$
  - $\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$
  - $\mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$

多层感知机结构


Neural Network Architecture
Input Layer
Hidden Layer
Output Layer

深层神经网络 - 基本概念

- **深层神经网络**包含多个隐藏层 (通常 >2 层)
- **深度学习优势**:
  - **层次化特征学习**：低层学习简单特征，高层学习复杂特征
  - **表达能力增强**：可以拟合更复杂的函数
  - **参数效率**：相比宽而浅的网络，参数利用更高效
- **挑战**:
  - 训练困难（梯度消失/爆炸）
  - 需要大量数据
  - 计算资源需求高
  - 过拟合风险增加

深层神经网络 - 表达能力

- **通用近似定理**:
  - 具有单个隐藏层的前馈神经网络可以近似任何连续函数
  - 深度网络比宽度网络更高效地表示某些函数
- **深度的优势**:
  - 参数数量随层数线性增长，而表达能力可指数增长
  - 能够学习更抽象的特征表示
  - 可以模拟复杂的决策过程

3. 损失函数 - 回归问题

- **回归问题常用损失函数**：
  - **均方误差 (MSE)**：$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
    - 最常用的回归损失函数
    - 惩罚大误差，对异常值敏感
  - **平均绝对误差 (MAE)**：$L = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
    - 对异常值更鲁棒
    - 梯度恒定，可能不利于学习
  - **Huber 损失**：结合 MSE 和 MAE 优点
    - $L_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta(|y-\hat{y}| - \frac{\delta}{2}) & \text{otherwise} \end{cases}$
    - 对小误差使用 MSE，大误差使用 MAE
    - 同时获得平滑梯度和鲁棒性

3. 损失函数 - 分类问题

- **分类问题常用损失函数**：
  - **二元交叉熵**：$L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$
    - 用于二分类问题
    - 搭配 Sigmoid 激活函数
  - **多类交叉熵**：$L = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$
    - 用于多分类问题
    - 搭配 Softmax 激活函数
  - **Softmax 函数**：$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$
    - 将输出转换为概率分布
    - 所有输出和为 1

梯度下降与优化 - 基本原理

- **梯度下降**是训练神经网络的基础优化算法
  - **核心思想**：沿着损失函数的负梯度方向更新参数
  - **参数更新公式**：$\theta = \theta - \alpha \nabla_\theta J(\theta)$
  - **学习率**α 控制更新步长
- **梯度下降变体**：
  - **批量梯度下降 (BGD)**：使用所有样本计算梯度
  - **随机梯度下降 (SGD)**：每次使用单个样本
  - **小批量梯度下降 (Mini-batch GD)**：使用小批量样本

梯度下降与优化 - 反向传播

- **反向传播**是高效计算梯度的算法
- **算法步骤**：
  1. **前向传播**：计算每层的激活值和输出
  2. **计算输出层误差**：$\delta^{[L]} = \nabla_{a^{[L]}} J \odot \sigma'(z^{[L]})$
  3. **反向传播误差**：$\delta^{[l]} = (W^{[l+1]})^T \delta^{[l+1]} \odot \sigma'(z^{[l]})$
  4. **计算梯度**：$\nabla_{W^{[l]}} J = \delta^{[l]} (a^{[l-1]})^T$
  5. **更新参数**：$W^{[l]} = W^{[l]} - \alpha \nabla_{W^{[l]}} J$

梯度下降与优化 - 高级优化算法

- **动量法 (Momentum)**:
  - 累积过去梯度，加速收敛
  - $v_t = \gamma v_{t-1} + \alpha \nabla_\theta J(\theta)$
  - $\theta = \theta - v_t$
- **AdaGrad**:
  - 自适应学习率，为不同参数调整更新速度
  - 对频繁更新的参数减小步长
- **RMSProp**:
  - 解决 AdaGrad 学习率递减问题
  - 使用移动平均累积梯度平方
- **Adam**:
  - 结合动量和自适应学习率
  - 当前最流行的优化算法之一
  - 自动调整每个参数的学习率

参数初始化 - 重要性与挑战

- **参数初始化**对神经网络训练至关重要：
  - 影响收敛速度和稳定性
  - 影响最终模型性能
  - 可能导致梯度消失/爆炸
- **不当初始化的问题**：
  - 零初始化：导致隐藏单元对称性，无法学习不同特征
  - 过大初始值：导致梯度爆炸
  - 过小初始值：导致梯度消失

参数初始化 - 常用方法

- **常见初始化策略**：
  - **随机初始化**：$W \sim \mathcal{U}(-\epsilon, \epsilon)$
    - 打破对称性
    - 需要谨慎选择 ϵ
  - **Xavier/Glorot 初始化**：$W \sim \mathcal{N}(0, \sqrt{2/(n_{in} + n_{out})})$
    - 适用于 tanh、sigmoid 激活函数
    - 保持方差在前向传播和反向传播中稳定
  - **He 初始化**：$W \sim \mathcal{N}(0, \sqrt{2/n_{in}})$
    - 适用于 ReLU 激活函数
    - 为深层 ReLU 网络设计
- **偏置初始化**：
  - 通常初始化为 0 或小常数
  - 某些情况下可设为正值（如 ReLU 激活函数）

正则化方法 - Dropout

- **Dropout**:
  - 训练时随机”关闭”一部分神经元
  - 每个神经元以概率 p 被保留
  - 测试时不关闭神经元，但权重需缩放
- **原理**:
  - 防止神经元共适应
  - 近似集成多个不同网络
  - 迫使网络学习更鲁棒的特征

正则化方法 - 其他技术

- **早停 (Early Stopping)**:
  - 监控验证集性能
  - 在验证误差开始上升时停止训练
- **数据增强 (Data Augmentation)**:
  - 通过变换已有数据创建新样本
  - 常用于图像 (旋转、缩放、翻转)
  - 增加训练数据多样性
- **批量归一化 (Batch Normalization)**:
  - 标准化每层的输入分布
  - 加速训练收敛
  - 允许使用更高学习率
  - 具有轻微正则化效果

深度学习基本概念

- **深度学习**是机器学习的一个分支，使用多层神经网络学习数据表示
- **核心特点**：
  - 自动特征提取
  - 端到端学习
  - 需要大量数据
  - 需要强大计算资源
  - 可处理非结构化数据
- **常见深度学习架构**：
  - **卷积神经网络 (CNN)**：处理图像和网格数据
  - **循环神经网络 (RNN/LSTM/GRU)**：处理序列数据
  - **Transformer**：处理序列数据，基于自注意力机制

深度学习应用与前沿

- **计算机视觉**:
  - 图像分类、目标检测、图像分割
  - 人脸识别、姿态估计
- **自然语言处理**:
  - 机器翻译、情感分析
  - 问答系统、文本生成
  - 大型语言模型 (LLM)
- **生成模型**:
  - GAN(生成对抗网络)
  - 扩散模型 (Diffusion Models)
  - 文本到图像转换
- **自监督学习**:
  - 无需大量标注数据
  - 从数据本身学习有用表示