集成学习与神经网络

汪小圈

2025-03-17

内容安排

- 集成学习进阶
  - Stacking
  - 高级 GBDT 实现：XGBoost 与 LightGBM
  - CatBoost
- 神经网络基础
  - 感知机模型
  - 前向传播与反向传播
  - 激活函数
  - 损失函数
- 深度学习
  - 卷积神经网络
  - 循环神经网络
  - Transformer

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

Stacking（堆叠集成）

- **核心思想**：
  - 训练多个不同类型的基学习器
  - 使用一个元学习器 (meta-learner) 组合基学习器的输出
  - 形成多层堆叠的结构
- **算法流程**：
  1. 训练多个不同类型的基学习器 (如决策树、SVM、神经网络等)
  2. 基学习器对训练集进行预测，生成新特征
  3. 使用这些新特征训练元学习器
  4. 元学习器做出最终预测
- **优点**：
  - 利用不同算法的优势
  - 减少单一模型的局限性
  - 通常比单一模型性能更好
  - 适合处理复杂问题

高级 GBDT 实现：XGBoost

- **XGBoost (eXtreme Gradient Boosting)**
  - 对传统 GBDT 的高效实现和扩展
  - 由陈天奇在 2014 年提出
- **主要创新点**：
  - 引入了正则化项，避免过拟合
  - 使用二阶导数优化，提高收敛速度
  - 支持并行计算，训练速度快
  - 处理缺失值能力强
  - 内置交叉验证功能
- **优化目标函数**：$Obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{T} \Omega(f_j)$
  其中 $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$

高级 GBDT 实现：LightGBM

- **LightGBM (Light Gradient Boosting Machine)**
  - 微软开发的 GBDT 实现
  - 在大规模数据上的性能尤为优秀
- **主要创新点**：
  - 采用基于直方图的算法，减少内存使用
  - 使用叶子生长策略 (Leaf-wise) 而非水平生长 (Level-wise)
  - 支持类别特征的直接输入
  - 使用互斥特征捆绑 (EFB) 减少特征数量
  - 采用基于梯度的单边采样 (GOSS) 减少样本数量
- **叶子生长策略**：
  - 每次分裂增益最大的叶子节点
  - 比传统水平生长更能减少误差

CatBoost

- **CatBoost (Categorical Boosting)**
  - Yandex 开发的 Boosting 算法
  - 专为处理类别型特征而设计
- **主要特点**:
  - 有效处理类别型特征，无需手动编码
  - 使用排序提升 (Ordered Boosting) 减少过拟合
  - 自动处理缺失值
  - 支持 GPU 加速
  - 默认参数下表现优异，易于使用
- **类别特征处理方法**:
  - 目标统计 (Target Statistics)
  - 组合类别特征
  - 防止类别特征泄露信息

神经网络简介 (Neural Network)

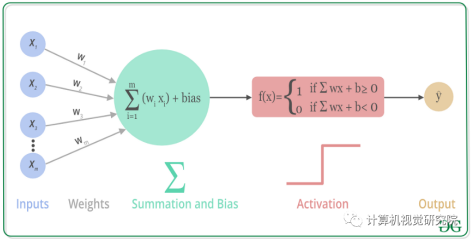
- **神经网络**是一种模拟生物神经系统的**计算模型**
- **基本结构**:
  - 由大量相互连接的神经元组成
  - 能学习复杂的非线性关系
  - 具有强大的模式识别能力
- **历史发展**:
  - 1943 年: McCulloch 和 Pitts 提出第一个神经元模型
  - 1958 年: Rosenblatt 提出感知机
  - 1986 年: 反向传播算法提出
  - 2006 年: 深度学习兴起
  - 2012 年: AlexNet 赢得 ImageNet 挑战赛，标志深度学习的突破
- **优势**:
  - 强大的非线性建模能力
  - 自动特征提取能力
 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

神经元模型 (Neuron Model)

- **神经元**是神经网络的基本单元，也称为**感知机 (Perceptron)**
- **神经元组成部分**：
  - **输入 (Input)**：来自其他神经元或外部环境的信号
  - **权重 (Weight)**：表示连接的强度或重要性
  - **偏置 (Bias)**：调整激活阈值
  - **加权求和 (Weighted Sum)**：$z = \sum_i w_i x_i + b$
  - **激活函数 (Activation Function)**：引入非线性，$a = \sigma(z)$
  - **输出 (Output)**：传递给下一层神经元
- **数学表达式**：$a = \sigma(\sum_i w_i x_i + b)$
- **生物神经元类比**：
  - **树突 (Dendrites)** 输入
  - **突触 (Synapses)** 权重
  - **细胞体 (Cell Body)** 加权求和与激活
  - **轴突 (Axon)** 输出

多层神经网络 (Multilayer Neural Network)

- **多层神经网络**又称**多层感知机 (MLP)**，由多层神经元组成
- **网络层次**：
  - **输入层 (Input Layer)**：接收外部输入数据
  - **隐藏层 (Hidden Layer)**：提取特征，可有多层
  - **输出层 (Output Layer)**：产生最终预测结果
- **全连接网络**：
  - 前一层的每个神经元都与后一层的所有神经元连接
  - 每个连接都有一个可训练的权重



- **深度神经网络 (DNN)**：
  - 具有多个隐藏层的神经网络
  - 能学习更抽象的特征表示
  - 具有更强的表达能力

前向传播 (Forward Propagation)

- **前向传播**是指信号从输入层经过隐藏层传递到输出层的过程
- **计算步骤**：
  1. **输入层**：接收输入数据 $\mathbf{x}$
  2. **隐藏层**：对于第 l 层的每个神经元，计算
     - $\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}$
     - $\mathbf{a}^{[l]} = \sigma^{[l]}(\mathbf{z}^{[l]})$
  3. **输出层**：计算最终输出 $\hat{\mathbf{y}} = \mathbf{a}^{[L]}$
- **矩阵表示**：
  - $\mathbf{W}^{[l]}$ 是权重矩阵
  - $\mathbf{b}^{[l]}$ 是偏置向量
  - $\mathbf{a}^{[l]}$ 是第 l 层的激活值（输出）
  - $\sigma^{[l]}$ 是第 l 层的激活函数

反向传播算法 (Backpropagation)

- **反向传播**是训练神经网络的核心算法，用于更新网络权重
- **核心思想**：
  - 计算损失函数关于网络参数的梯度
  - 使用梯度下降法更新参数
  - 误差信号从输出层反向传播到输入层
- **算法步骤**：
  1. **前向传播**：计算网络预测输出
  2. **计算损失**：衡量预测与真实值的差异
  3. **反向传播误差**：计算每层的误差项 $\delta^{[l]}$
     - 输出层：$\delta^{[L]} = \frac{\partial L}{\partial \mathbf{z}^{[L]}}$
     - 隐藏层：$\delta^{[l]} = (\mathbf{W}^{[l+1]})^T \delta^{[l+1]} \odot \sigma'^{[l]}(\mathbf{z}^{[l]})$
  4. **计算梯度**：$\frac{\partial L}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{a}^{[l-1]})^T$
  5. **更新参数**：$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \frac{\partial L}{\partial \mathbf{W}^{[l]}}$

激活函数 (Activation Functions)

- **激活函数**引入非线性，使神经网络能学习复杂模式
- **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
- **改进版 ReLU**：
  - **Leaky ReLU**：$f(z) = \max(\alpha z, z)$，α 是小常数
  - **ELU**：$f(z) = \begin{cases} z, & \text{if } z > 0 \\ \alpha(e^z - 1), & \text{if } z \leq 0 \end{cases}$
- **专用激活函数**：
  - **Softmax**：$\mathrm{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$

损失函数 (Loss Functions)

- **损失函数**衡量模型预测与真实值之间的差异，是神经网络优化的目标
- **常用损失函数**：
  - **均方误差 (MSE)**：$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
    - 适用于回归问题
    - 对异常值敏感
  - **交叉熵损失 (Cross-Entropy Loss)**：$L_{CE} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$
    - 适用于分类问题
    - 二分类：$L = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$
    - 多分类：$L = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$
  - **Huber 损失**：结合 MSE 和 MAE 的优点，对异常值更鲁棒
  - **Hinge 损失**：$L = \max(0, 1 - y \cdot \hat{y})$，用于 SVM 和某些分类问题

深度学习 (Deep Learning)

- **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**：
  - 多层次特征提取
  - 端到端学习
  - 需要大量数据
  - 需要强大计算资源
- **关键要素**：
  - 深层神经网络架构
  - 有效的优化算法
  - 正则化技术
  - 大规模数据集
  - GPU 加速计算
- **常见深度学习架构**：
  - 卷积神经网络 (CNN)
  - 循环神经网络 (RNN)
  - Transformer
  - 生成对抗网络 (GAN)

卷积神经网络 (CNN)

- **卷积神经网络**专门用于处理网格结构数据，特别是图像
- **核心组件**：
  - **卷积层**：应用卷积核提取局部特征
  - **池化层**：降低维度，提取显著特征
  - **全连接层**：综合特征进行最终预测
- **卷积操作**：
  - 使用小型卷积核在输入上滑动
  - 自动学习空间层次特征
  - 参数共享减少模型复杂度
- **优势**：
  - 局部感受野和权重共享
  - 平移不变性
  - 特征层次化表示
  - 大幅减少参数数量
- **应用领域**：
  - 图像分类与识别
  - 目标检测

循环神经网络 (RNN)

- **循环神经网络**专门用于处理序列数据
- **核心思想**:
  - 引入循环连接，记忆之前的信息
  - 同一层的参数在各时间步共享
  - 能处理变长序列输入
- **基本 RNN 结构**:
  - $h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$
  - $y_t = W_{hy}h_t + b_y$
- **缺点**:
  - 存在梯度消失/爆炸问题
  - 难以捕捉长期依赖关系
- **LSTM (长短期记忆网络)**:
  - 引入门控机制 (遗忘门、输入门、输出门)
  - 解决长期依赖问题
  - $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
  - $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
  - $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Transformer

- **Transformer** 是一种基于自注意力机制的神经网络
- **核心创新**:
  - 完全放弃了 CNN 和 RNN 结构
  - 引入自注意力机制
  - 并行计算，训练更高效
  - 能捕捉全局依赖关系
- **关键组件**:
  - **多头自注意力 (Multi-Head Attention)**
  - **位置编码 (Positional Encoding)**
  - **前馈神经网络 (Feed-Forward Network)**
  - **残差连接与层归一化**
- **自注意力机制**:
  - 计算输入序列中每个位置与所有位置的关联程度
  - $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
- **应用领域**:
  - 自然语言处理 (BERT, GPT)
  - 计算机视觉 (ViT)

深度学习的最新进展

- **大型语言模型 (LLM)**:
  - GPT-4, Claude, LLaMA 等
  - 具有惊人的语言理解和生成能力
  - 基于 Transformer 架构
- **扩散模型 (Diffusion Models)**:
  - DALL-E, Stable Diffusion, Midjourney
  - 高质量图像生成
  - 文本到图像转换
- **自监督学习 (Self-Supervised Learning)**:
  - 无需大量标注数据
  - 从数据本身学习有用表示
  - 代表作：BERT, SimCLR, MAE
- **多模态学习 (Multimodal Learning)**:
  - 融合不同类型的数据 (文本、图像、音频)
  - CLIP, DALL-E, GPT-4 等
- **联合嵌入学习 (Contrastive Learning)**:
  - 学习相似样本在特征空间的聚集