从零搭建 CTR 模型

MLE 算法指北

2025 年 2 月 15 日

1 问题描述

本项目的目标是构建一个最最简化版点击率预测模型（Click Prediction Model），用于预测用户是否会点击当前的广告或内容。数据集包括以下特征：

- **X1**: 用户阅读猫相关帖子的时长（float）
- **X2**: 用户阅读狗相关帖子的时长（float）
- **X3**: 用户阅读兔子相关帖子的时长（float）
- **X4**: 当前帖子类别（类别型：猫、狗、兔子）
- **y**: 点击（标签，0 或 1，存在类别不平衡问题）

2 数据预处理

2.1 缺失值处理

数据集中可能存在缺失值，我们采用均值填充：

```
df.fillna(df.mean(), inplace=True)
```

2.2 类别特征处理

对于类别型特征 X4，我们使用 **One-Hot Encoding** 进行编码：

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse_output=False)
X4_encoded = encoder.fit_transform(df[['X4']])
```

2.3 特征标准化

对于数值型特征，我们使用 **Z-Score 标准化 **：

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['X1', 'X2', 'X3']] = scaler.fit_transform(df[['X1', 'X2', 'X3']])
```

2.4 延伸问题

- 为什么要对数值特征进行标准化?
- 你如何处理类别特征的高维问题?
- 你考虑过其他编码方式（如 Target Encoding、Embedding）吗?

2.5 延伸问题答案解析

2.5.1 为什么要对数值特征进行标准化？

在机器学习模型（尤其是神经网络和基于梯度优化的模型，如逻辑回归、SVM、MLP 等）中，数值特征的尺度差异可能会导致模型的训练过程变得缓慢或收敛不稳定。标准化（Standardization）通常指将特征转换为 ** 均值为 0，标准差为 1** 的数据分布，其主要优势包括：

- **提升优化效率**: 在梯度下降过程中，如果特征的取值范围相差较大，模型参数的更新幅度可能不一致，导致训练速度变慢。标准化可以使所有特征具有相似的数值范围，加快收敛速度。
- **防止特征主导模型**: 某些特征的取值范围较大（如” 收入” 可能在 [10,000, 1,000,000] 之间，而” 年龄” 可能在 [0, 100] 之间），如果不标准化，模型可能会更加依赖高数值范围的特征，而忽略低数值范围的特征。

- **提升模型稳定性**：线性模型（如逻辑回归、线性回归）和神经网络对数据分布较敏感，标准化可以减少训练中的数值不稳定问题。
- **提升距离度量的有效性**：在使用基于距离的算法（如 KNN、SVM、K-Means）时，标准化可以确保不同尺度的特征在计算欧几里得距离时不会产生不均衡影响。

常见标准化方法：

- **Z-Score 标准化**：$X' = \frac{X-\mu}{\sigma}$
- **Min-Max 归一化**：$X' = \frac{X-X_{min}}{X_{max}-X_{min}}$

适用场景：

- 对于梯度优化模型（如 MLP、逻辑回归、SVM），推荐使用 Z-Score 标准化。
- 对于神经网络，Batch Normalization 或 Layer Normalization 也可以替代标准化。
- 对于树模型（如 XGBoost、Random Forest），标准化通常不是必要的。

2.5.2 你如何处理类别特征的高维问题？

在真实数据中，类别特征可能包含数百甚至数十万个类别（如" 用户 ID"、"IP 地址"、" 城市名称" 等），如果直接使用 One-Hot Encoding，可能会导致：

- **维度爆炸**：如果一个类别特征有 1000 个唯一值，One-Hot Encoding 会生成 1000 维的特征向量，增加计算和存储成本。
- **数据稀疏性**：高维度 OHE 生成的矩阵是 ** 稀疏矩阵 **，会降低模型计算效率，尤其是对于神经网络。
- **信息丢失**：如果训练数据中某些类别很少出现，OHE 可能无法充分学习它们的模式。

解决方案：

- **Target Encoding（目标编码）**：用每个类别对应的 ** 目标均值 ** 替代原始类别值，一般来说适用于线性模型和树模型。适用于 ** 类别数量较多但稳定 ** 的场景，如" 城市"、" 职业" 等。但如果类别数量太多（如用户 ID），可能会导致 ** 过拟合 **，可以使用 ** 平滑技术 **（如贝叶斯均值编码）

```
target_mean = df.groupby('category')['y'].mean()
df['category_encoded'] = df['category'].map(target_mean)
```

- **Embedding（嵌入）**：在神经网络中，可以将高维类别特征映射到低维向量空间。适用于推荐系统、广告点击率预测等任务，可以用 Embedding 进行用户行为建模。例如：给定 10000 个唯一类别，可以映射到 32 维空间。

```
category_embedding = nn.Embedding(num_categories, embedding_dim)
```

适用于高维类别特征（如推荐系统中的用户 ID、商品 ID）。

- **降维方法（PCA、特征选择）**：如果类别特征经过 One-Hot Encoding 后维度过高，可以使用 PCA 降维，或者采用 Lasso 进行特征选择。
- **Hashing Trick（特征哈希）**：对于极高维类别特征（如 URL、IP 地址），可以用哈希函数将类别映射到固定数量的桶中，以减少维度。适用于 ** 超高维类别特征 **（如广告 ID），通过哈希函数将类别映射到一个固定大小的哈希桶中。缺点是存在哈希冲突，信息可能会丢失。

3 特征工程

3.1 构造交互特征

为了增强模型的表达能力，我们构造交互特征：

```
df['interaction'] = df['X1'] * df['X2']
```

3.2 类别比例特征

计算用户在不同类别上的时长比例：

```
df['cat_ratio'] = df['X1'] / (df['X1'] + df['X2'] + df['X3'] + 1e-5)
```

延伸问题:

- 交互特征如何帮助模型?
- 交互特征会导致维度增加，如何控制计算成本?
- 在高维数据情况下，如何进行特征选择?

3.3 延伸问题答案解析

3.3.1 交互特征如何帮助模型？

交互特征（Feature Interaction）指的是将两个或多个原始特征组合，以捕捉它们之间的相互关系。这种方法对于 ** 线性模型（如逻辑回归）** 和 ** 神经网络 ** 都非常重要。

交互特征的作用：

- **增强模型表达能力**：原始特征可能无法单独描述目标变量的模式，而交互特征可以引入更复杂的信息。例如：

$$\text{new\_feature} = X_1 \times X_2 \tag{1}$$

  如果 X_1 是" 用户阅读猫帖子的时长"，X_2 是" 用户阅读狗帖子的时长"，那么 $X_1 \times X_2$ 可能反映了" 用户对宠物帖子的总关注度"。

- **让线性模型学习非线性关系**：例如，逻辑回归本质上是线性的，而交互特征可以引入非线性信息。例如：

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \times X_2) \tag{2}$$

  其中 β_3 代表 X_1 和 X_2 交互对目标变量的影响。

- **提高模型的泛化能力**：对于神经网络，使用交互特征可以减少模型参数，使其在小样本数据上表现更好。

交互特征构造方法：

- ** 乘积特征（Multiplication Feature）**：$X_1 \times X_2$
- ** 多项式特征（Polynomial Features）**：$X_1^2, X_2^2, X_1 \times X_2$
- ** 决策树生成交互特征 **：使用 **XGBoost、LightGBM** 训练树模型，并提取重要特征组合。
- ** 深度学习自动学习交互特征 **：如 **Wide & Deep 模型、FM（Factorization Machines）** 等。

3.3.2 交互特征会导致维度增加，如何控制计算成本？

- ** 先进行相关性分析，筛选重要的交互特征 **，避免无效特征带来的维度增加。
- ** 利用神经网络（DeepFM、AutoEncoder）自动学习交互特征 **，减少人工构造的特征组合。
- ** 使用 PCA、Lasso、Hashing Trick 等方法 ** 降低维度，提高计算效率。
- 在 ** 推荐系统、CTR 预测、搜索广告等高维任务 ** 中，DeepFM 和 Wide & Deep 是更优的交互特征建模方式。

3.3.3 在高维数据情况下，如何进行特征选择？

当数据维度非常高时（如文本数据、推荐系统中的 ID 特征），需要进行特征选择，以减少计算成本并提高模型泛化能力。

特征选择方法：

- **过滤法（Filter Methods）**：基于统计分析
  - ** 方差分析（Variance Analysis）**：如果某个特征在整个数据集中方差极小，说明该特征几乎恒定，对模型无贡献，可删除。
  - ** 皮尔逊相关系数（Pearson Correlation）**：计算特征和目标变量的相关性，绝对值接近 1 说明特征较重要。
  - ** 互信息（Mutual Information）**：衡量特征对目标变量的信息增益（适用于分类任务）。
- **嵌入法（Embedded Methods）**：基于模型选择特征
  - **Lasso 回归（L1 正则化）**：自动压缩不重要的特征
  - **基于树模型的特征选择**：看特征重要性矩阵
- **包装法（Wrapper Methods）**：使用模型评估不同特征组合
  - 递归特征消除（RFE）：逐步去除低权重特征
- **深度学习方法**：
  - AutoEncoder（自动编码器）可以降维：
  - SHAP 解释 XGBoost 模型的特征重要性：

4 模型搭建与选择

4.1 MLP 模型

使用 **Scikit-Learn** 训练 MLP：

```
from sklearn.neural_network import MLPClassifier
mlp_model = MLPClassifier(hidden_layer_sizes=(64, 32), activation='relu', solver='adam')
mlp_model.fit(X_train, y_train)
y_pred = mlp_model.predict(X_test)
y_pred_prob = mlp_model.predict_proba(X_test)[:, 1]
```

4.2 树模型

除了 MLP，我们还使用 **Random Forest、XGBoost、LightGBM** 进行对比：

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)  # 预测类别
y_pred_prob = rf_model.predict_proba(X_test)[:, 1]  # 预测概率

import xgboost as xgb
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
y_pred = xgb_model.predict(X_test)
y_pred_prob = xgb_model.predict_proba(X_test)[:, 1]

import lightgbm as lgb
lgb_model = lgb.LGBMClassifier()
lgb_model.fit(X_train, y_train)
y_pred = lgb_model.predict(X_test)
y_pred_prob = lgb_model.predict_proba(X_test)[:, 1]
```

4.3 简单的双塔模型

我们将使用 PyTorch 搭建一个双塔模型（Two-Tower Model），其中：

- 用户塔（User Tower）处理用户行为特征（用户在不同类别帖子上的阅读时长等）。
- 广告塔（Ad Tower）处理广告特征（当前帖子类别的 One-Hot 编码等）。
- 最终匹配分数通过点积计算用户兴趣与广告匹配度，用于预测点击概率。

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import numpy as np

# 划分 feature
X_user_tensor = torch.tensor(user_features, dtype=torch.float32)
X_ad_tensor = torch.tensor(ad_features, dtype=torch.float32)
y_tensor = torch.tensor(y, dtype=torch.float32).unsqueeze(1)  # (batch_size, 1)

# 创建数据集和 DataLoader
class ClickDataset(Dataset):
    def __init__(self, user_features, ad_features, labels):
        self.user_features = user_features
        self.ad_features = ad_features
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return self.user_features[idx], self.ad_features[idx], self.labels[idx]
```

```
dataset = ClickDataset(X_user_tensor, X_ad_tensor, y_tensor)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)

class TwoTowerModel(nn.Module):
    def __init__(self, user_dim, ad_dim, embedding_dim=16):
        super(TwoTowerModel, self).__init__()
        # 用户塔 (User Tower)
        self.user_tower = nn.Sequential(
            nn.Linear(user_dim, 64),
            nn.ReLU(),
            nn.Linear(64, embedding_dim)
        )
        # 广告塔 (Ad Tower)
        self.ad_tower = nn.Sequential(
            nn.Linear(ad_dim, 64),
            nn.ReLU(),
            nn.Linear(64, embedding_dim)
        )
        # Sigmoid 层用于二分类
        self.sigmoid = nn.Sigmoid()
    def forward(self, user_x, ad_x):
        user_embedding = self.user_tower(user_x)  # 计算用户嵌入
        ad_embedding = self.ad_tower(ad_x)  # 计算广告嵌入
        # 计算匹配分数 (点积)
        match_score = (user_embedding * ad_embedding).sum(dim=1, keepdim=True)
        # 归一化输出点击概率
        output = self.sigmoid(match_score)
        return output

# 定义模型
model = TwoTowerModel(user_dim=5, ad_dim=3, embedding_dim=16)
criterion = nn.BCELoss()  # 二分类交叉熵损失
optimizer = optim.Adam(model.parameters(), lr=0.01)

model.eval()
with torch.no_grad():
    y_pred_prob = model(X_user_tensor, X_ad_tensor).squeeze().numpy()
    y_pred = (y_pred_prob > 0.5).astype(int)
```

4.4 延伸问题:

- 树模型的相关八股，详见之前笔记
- 神经网络模型相关八股，详见之前笔记

5 模型评估

在机器学习任务中，模型评估是至关重要的环节，决定了模型在实际业务场景中的可用性。本章节将详细解析常见的分类模型评估指标，并结合点击率预测的业务场景，分析各指标的适用性。

5.1 分类模型评估指标

对于二分类任务（如点击率预测），常用的评估指标包括：

- **准确率（Accuracy）**
- **精确率（Precision）**
- **召回率（Recall）**
- **F1 分数（F1-Score）**
- **ROC-AUC 曲线（Receiver Operating Characteristic - Area Under Curve）**
- **PR-AUC 曲线（Precision-Recall - Area Under Curve）**
- **对数损失（Log Loss）**

5.1.1 准确率（Accuracy）

定义：

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

其中：

- TP（True Positive）：预测点击，且实际点击
- TN（True Negative）：预测不点击，且实际不点击
- FP（False Positive）：预测点击，但实际未点击
- FN（False Negative）：预测不点击，但实际点击

适用场景：

- 适用于类别均衡的分类任务。
- 在点击率预测中，由于点击行为往往较少（正负样本比例严重失衡），准确率可能不能很好地衡量模型效果。例如，如果 95% 的样本都没有点击，那么一个始终预测” 不点击” 的模型也能达到 95% 的准确率，但它实际上毫无价值。

业务示例：

- 在垃圾邮件分类任务中，如果垃圾邮件和正常邮件的比例接近，准确率是一个合理的评估指标。
- 在 CTR 预测中，由于点击行为较少，准确率可能误导模型效果评估，需结合其他指标使用。

5.1.2 精确率（Precision）

定义：

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

精确率表示在所有被预测为” 点击” 的样本中，实际点击的比例。

适用场景：

- 适用于 ** 错误代价较高的场景 **，例如：
  - ** 广告投放 **：如果系统错误地预测用户会点击广告（False Positive），广告可能被无效展示，浪费预算。
  - ** 医疗诊断 **：如果一个疾病预测模型误诊患者为阳性（False Positive），可能会导致不必要的治疗。

业务示例：

- 在广告推荐系统中，高精确率意味着展示的广告更有可能被点击，有助于提升转化率（CTR）。

5.1.3 召回率（Recall）

定义：

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

召回率表示在所有实际点击的样本中，成功被模型预测为" 点击" 的比例。

适用场景：

- 适用于 ** 遗漏代价较高的场景 **，例如：
  - ** 欺诈检测 **：如果模型无法识别所有欺诈交易（False Negative），可能导致损失。
  - ** 搜索引擎 **：如果用户搜索某个关键词，系统没有返回相关结果（False Negative），用户体验会受影响。

业务示例：

- 在广告投放中，高召回率意味着更多的广告会被展示，但可能会降低精确率，导致更多无效点击预测。

5.1.4 F1 分数（F1-Score）

定义：

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

F1 分数是精确率和召回率的调和平均数，综合了两者的优势，适用于 ** 类别不均衡的数据集 **。

适用场景：

- 适用于 ** 类别不均衡问题 **，如 CTR 预测、欺诈检测等。

业务示例：

- 在 CTR 预测中，单独优化精确率或召回率可能会导致模型倾向于某一类，因此 F1 分数能够更全面地衡量模型效果。

5.1.5 ROC-AUC 曲线

定义：ROC 曲线（Receiver Operating Characteristic）绘制了 ** 真正例率（Recall）与假正例率（False Positive Rate, FPR）** 之间的关系，其中：

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

AUC（Area Under Curve）表示 ROC 曲线下的面积，其范围在 [0,1] 之间，越接近 1 表示模型越优。

适用场景：

- 适用于需要 ** 综合评估分类能力 ** 的任务，如 CTR 预测、信用评分等。
- 在类别不均衡数据中，ROC-AUC 仍然可以提供较好的评估能力。

业务示例：

- 在 CTR 预测任务中，AUC 反映了模型区分" 点击" 和" 不点击" 用户的能力，可用于不同广告投放策略的比较。AUC 反映了模型将正例排在负例之前的概率，很好的展现了 CTR 模型的排序能力

5.2 评估指标代码实现

5.2.1 计算准确率（Accuracy）

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

5.2.2 计算精确率（Precision）

```
precision = precision_score(y_true, y_pred)
print(f"Precision: {precision:.4f}")
```

5.2.3 计算召回率（Recall）

```
recall = recall_score(y_true, y_pred)
print(f"Recall: {recall:.4f}")
```

5.2.4 计算 F1 分数（F1-Score）

```
f1 = f1_score(y_true, y_pred)
print(f"F1 Score: {f1:.4f}")
```

5.2.5 计算 ROC-AUC 分数

```
roc_auc = roc_auc_score(y_true, y_pred_prob)
print(f"ROC-AUC Score: {roc_auc:.4f}")
```

5.3 延伸问题

- 在实际业务中，根据优化目标的不同，你会如何选择合适的评估指标?

5.4 延伸问题答案解析

5.4.1 在实际业务中，根据优化目标的不同，你会如何选择合适的评估指标？

在 CTR 预测任务中，我们需要关注模型的整体性能，并综合考虑 ** 预测准确性、排序能力、业务收益 **。常见的评估指标包括：

- **AUC（Area Under Curve）** - 适用于排序任务
- **Log Loss（对数损失）** - 衡量模型的概率预测能力
- **PR-AUC（Precision-Recall AUC）** - 适用于类别不均衡数据
- **Calibration（校准曲线）** - 评估概率预测的可靠性

不同业务场景下的指标选择：

- ** 广告排序与竞价 ** ⇒ **AUC**：
  - AUC 衡量模型对点击行为的区分能力，适用于 ** 广告竞价排名（Ad Ranking）**，因为广告系统关注的是 ** 相对排序，而非具体概率 **。
- ** 广告点击率预估（CTR 预测）** ⇒ **Log Loss**：
  - Log Loss 关注的是 ** 概率预测的准确性 **，如果模型预测的概率接近真实分布，则 Log Loss 较小。
  - 适用于 ** 广告预算分配 **，如确定某个广告的投放概率。
- ** 目标优化（ROI / 转化率优化）** ⇒ **PR-AUC**：
  - 如果业务目标是最大化点击转化率（Conversion Rate, CVR），则 Precision-Recall 曲线更能反映点击行为的精准度。
- ** 实际点击预测（Calibration 校准）**：
  - 评估模型预测的概率是否符合真实点击率，如：

$$\mathbb{E}[p(click)|\text{Model Prediction} = p] \approx p \tag{8}$$

  - 通过 ** 校准曲线（Calibration Curve）** 检测模型是否存在偏差