24 | GBDT+LR：排序算法经典中的经典

2023-06-09 黄鸿波 来自北京

《手把手带你搭建推荐系统》

你好，我是黄鸿波。

在前面的课程中，我们讲了推荐系统中的数据处理、接口实现和一些召回算法和模型，从本章开始，我们就会进入一个新的篇章：推荐系统中的排序算法。

我一共设计了三节课，分别是 GBDT+LR、DeepFM 和重排序。这三节课的内容代表了基于算法的排序、基于模型的排序和基于人工的排序。

在本节课先聚焦在 GBDT+LR 上，你将会学到下面四个部分的知识。

1. 逻辑回归（LR）模型。
2. GBDT 算法概述。
3. GBDT 与 LR 的结合。

4. 实现一个 GBDT+LR 算法。

什么是逻辑回归（LR）模型

逻辑回归（LR,Logistic Regression）是一种传统机器学习分类模型，也是一种比较重要的非线性回归模型，其本质上是在线性回归模型的基础上，加了一个 Sigmoid 函数（也就是非线性映射），由于其简单、高效、易于并行计算的特点，在工业界受到了广泛的应用。

一般来讲，我们使用 LR 模型主要是用于分类任务，且大多以二分类为主。在推荐系统的实际业务中，LR 常用来作为 Baseline 模型，实现快速上线的目的。

从本质上来讲，逻辑回归和线性回归一样同属于广义线性模型。虽然说逻辑回归可以实现回归预测，但是在推荐算法中（或者说在大多数的业务场景中），我们都将其看作是线性模型并把它应用在分类任务中。

作为广义线性模型的一种，LR 假设因变量 y 服从伯努利分布。在推荐系统中我们用它来预估点击率，实际上就是来预测“点击”这个事件是否发生。这个“是否发生”实际上就是因变量 y。因为点击事件只有两种可能性，点击或者不点击（二分类问题）。这个问题，实际上就是服从伯努利分布的。

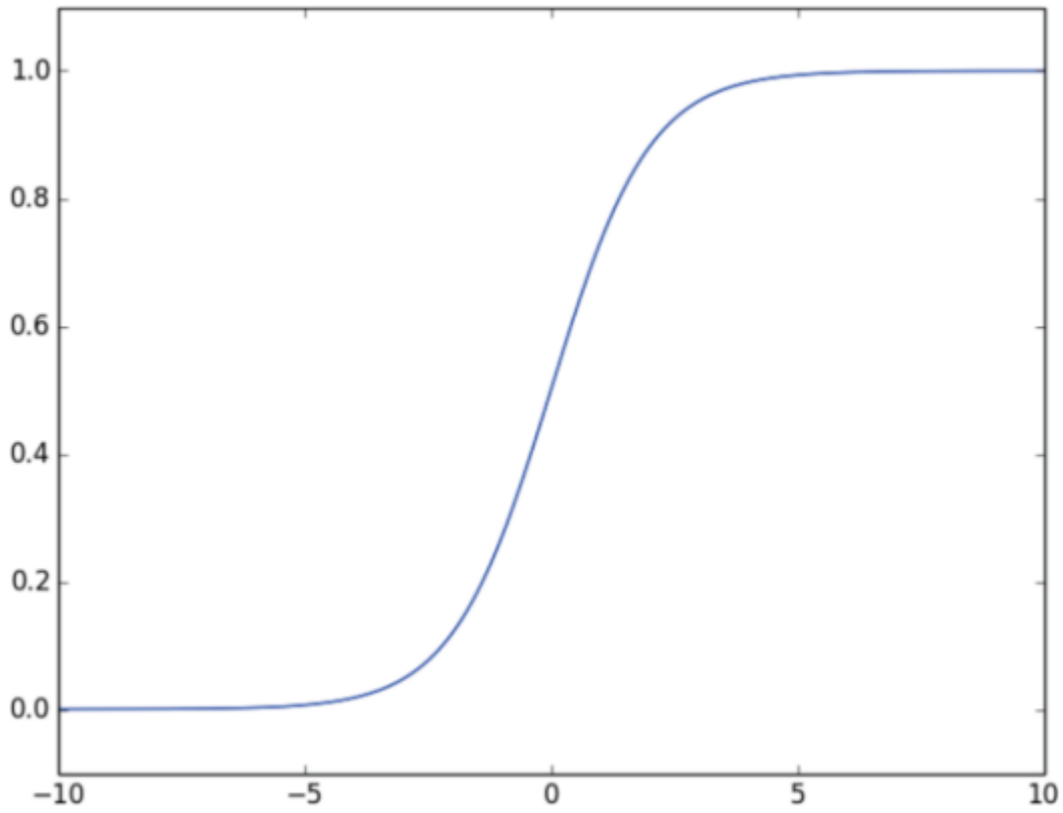
总结一下，**逻辑回归实际上就是在数据服从伯努利分布的假设下，通过极大似然的方法，运用梯度下降算法对参数进行求解，从而达到二分类。**

这里就会有一个问题，既然逻辑回归和线性回归都可以做到分类，那么在推荐系统中为什么用逻辑回归而不是线性回归呢？总的来说，有下面三点原因。

1. 虽然它们都能够处理二分类问题，但线性回归模型一般是处理连续特征的二分类问题，比如身高体重这种连续特征；而推荐系统中的目标是推荐一个物品给用户，这个物品实际上是离散的，因此不能使用线性回归进行预测。
2. 逻辑回归模型具有较高的模型可解释性，可以帮助我们更好地理解推荐系统中不同因素对推荐结果的影响，而线性回归在这一点上无法实现我们的需求。
3. 推荐系统中往往存在很多的噪音，逻辑回归可以更好地处理异常值，避免推荐结果被干扰。

通过上面我们知道，线性回归的输出实际上是一个连续值。如果我想把它用作分类问题，比如说二分类，应该怎么办呢？

很简单，**在线性回归的基础上，把它的输出通过另一个函数映射到[0, 1]这个区间范围内就能解决这个问题。**这个映射函数我们一般用 Sigmoid 函数，而映射之后的函数就是一个逻辑回归模型，它对应的逻辑回归图像如下。



其函数原型为 $y = \frac{1}{(1+e^{-z})}$。

简单说下它的推导过程。刚刚说过，逻辑回归就是线性回归加上 Sigmoid 函数，线性回归模型的公式为 $f(x) = \omega^T x$；值域为 $(-\infty, \infty)$。

我们不能直接基于线性模型来进行建模，那么中间的映射函数就选择 Sigmoid 函数，也就是 $y = \frac{1}{1+e^{-z}}$。

从上面对应的逻辑回归图像可以看出，Sigmoid 函数的值域在（0,1）之间，这样实际上就完美地解决了分类的问题。

在推荐系统中，逻辑回归对于特征处理有着非常强的优势，你可以对照下面这个表格了解一下。

逻辑回归对特征处理的优势	
特征选择	可以使用正则化技术来选择最重要的特征，提高模型效率和准确性
处理非线性特征	可以通过引入多项式和交互特征来处理非线性特征，增强模型的表现力
处理处理缺失值和异常值	可以处理缺失值或者异常值，使得模型更加健壮，能够预测更为准确的结果
训练速度方面	训练速度相对较快，在大规模推荐系统中可以快速地处理大量数据

一般来讲，使用逻辑回归处理特征会经过下面五个步骤。

1. 特征选择：逻辑回归可以使用正则化技术（如 L1、L2 正则化）来选择最重要的特征，从而降低维度并去除无关特征。选择相关的特征有助于提高模型的稳定性和准确性。
2. 处理缺失值和异常值：逻辑回归可以使用缺失值插补和异常值检测来处理缺失值和异常值，从而避免对模型产生影响，提高模型的健壮性。
3. 处理非线性特征：逻辑回归可以通过引入多项式和交互特征来处理非线性特征，从而增强模型的表现力。以多项式模型为例，逻辑回归可以使用幂函数或指数函数对特征进行转换，从而处理非线性变量。
4. 特征标准化：逻辑回归可以使用特征标准化来消除特征数据值的量纲影响，避免数值范围大的特征对模型产生很大影响。
5. 特征工程：逻辑回归也可以使用特征工程来创建新的特征，例如聚合或拆分现有特征、提取信号等。这有助于发现与目标变量相关的新信息，从而改进对数据的理解。

总的来说，逻辑回归依靠特征处理来提高模型的准确性、稳定性和表现能力，并且可以使用各种技术来处理特征，以便更好地利用数据和建立强大的预测模型。

什么是 GBDT 算法

讲完逻辑回归模型后，我们来讲 GBDT 算法。

GBDT（Gradient Boosting Decision Tree）算法是一种基于决策树的集成学习算法，它通过不断训练决策树来提高模型的准确性。GBDT 在每一次训练中都利用当前的模型进行预测，并将预测误差作为新的样本权重，然后训练下一棵决策树模型来拟合加权后的新数据。

> 决策树模型是一种基于树形结构的机器学习模型，用于解决分类与回归问题。它把所有的训练数据样本不断划分成更小的子集，以构建一棵树结构。每个节点代表一个特征属性，节点的分支代表该属性的不同取值。根据每个节点的属性值，将数据划分到不同的子节点里去，直到叶子节点表示最终的输出结果。在这里，分类树用于分类标签的值，比如用户的性别、文章是体育还是财经还是政治等；归回树用于预测实际的数值，例如温度、年龄、相关度等。

GBDT 中的 B 代表 Boosting。Boosting 算法的基本思想是通过将多个弱分类器线性组合形成一个强分类器，达到优化训练误差和测试误差的目的。具体应用时，每一轮将上一轮分类错误的样本重新赋予更高的权重，这样一来，下一轮学习就容易重点关注错分样本，提高被错分样本的分类准确率。

GBDT 由多棵 CART 树组成，本质是多颗回归树组成的森林。每一个节点按贪心分裂，最终生成的树包含多层，这就相当于一个特征组合的过程。

在推荐系统中，我们使用 GBDT 算法来优化和提高个性化推荐的准确性。通过 GBDT 算法对用户历史行为数据进行建模和学习，可以很容易地学习到学习用户的隐式特征（例如品味、购买能力、口味偏好等）。另外，GBDT 算法可以自动选择重要的特征，对离散型和连续型特征进行处理（如缺失值填充、离散化等），为特征工程提供更好的支持。

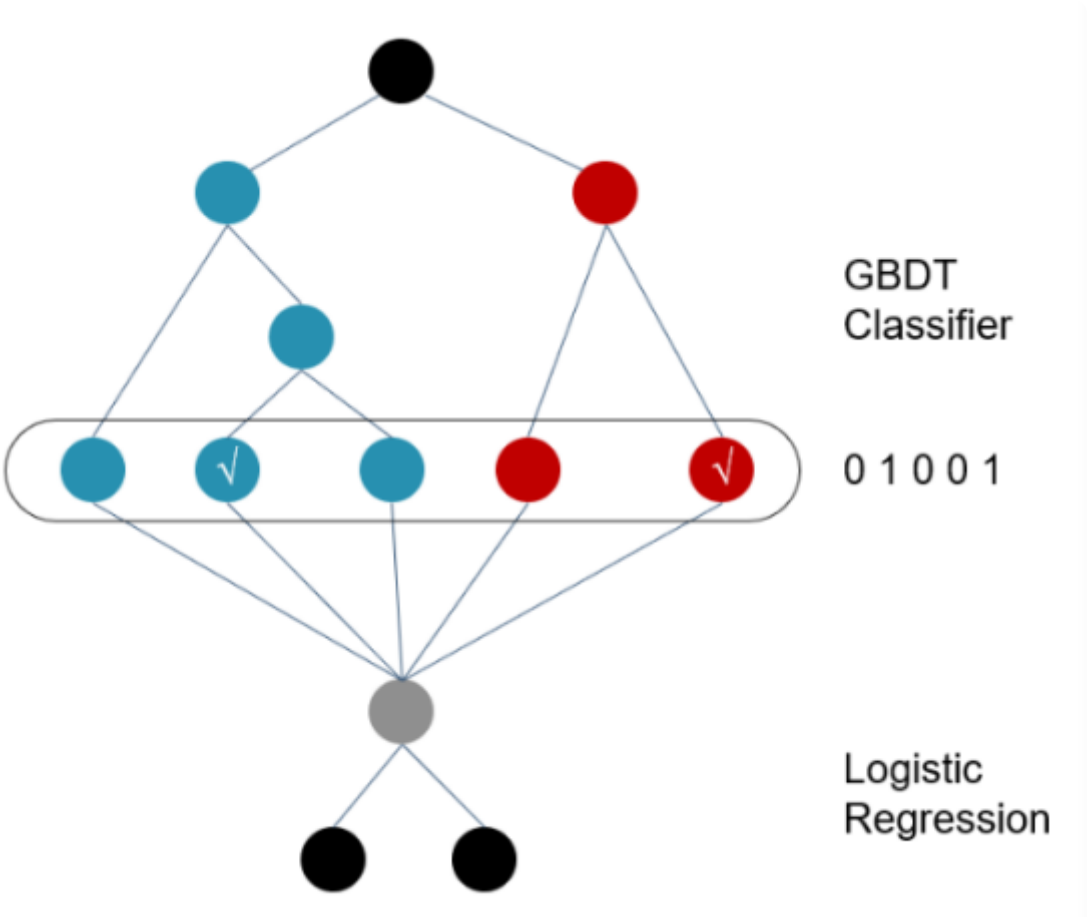
总的来说，**GBDT 算法可以帮助我们更好地理解用户行为和需求，提高推荐精度和用户满意度，让推荐系统更加智能、准确和可靠。**

GBDT 与 LR 的结合

对 LR 和 GBDT 有了简单的了解之后，接下来就是如何将 GBDT 与 LR 结合，进行推荐系统的排序。

在推荐系统中，GBDT+LR 使用最广泛的场景就是点击率预估，然后根据点击率预估的结果进行排序，因此 GBDT+LR 一般被应用于排序层中。

先来看一下 GBDT+LR 的模型结构。



可以看到，整个模型实际上被分成两个部分，下面是 LR 上面是 GBDT。从上往下看，整个模型的训练可以分成下面五个步骤。

1. GBDT 训练：使用 GBDT 对原始数据进行训练并生成特征。在训练过程中，每棵树都是基于前一棵树的残差进行构建。这样，GBDT 可以逐步减少残差，生成最终的目标值。

2. 特征转换：使用 GBDT 生成的特征进行转换。这些特征是树节点的输出，每个特征都对应于一个叶子节点。在转换过程中，每个叶子节点都会被转换为一个新的特征向量，代表这个叶子节点与其他节点的相对位置，并将这些特征向量连接起来形成新的训练集。
3. 特征归一化：对生成的特征进行归一化处理，确保不同维度的特征在训练过程中具有相等的权重。
4. LR 训练：使用 LR 对转换后的特征进行二分类或回归。特征向量被送入逻辑回归模型中进行训练，以获得最终的分类模型。在训练过程中，使用梯度下降法来更新模型参数，以最小化损失函数，损失函数的选择取决于分类问题的具体情况。
5. 模型预测：训练完成后，使用 LR 模型对新的数据进行预测。GBDT+LR 模型将根据特征生成函数和逻辑回归模型预测新数据的类别或值。

总之，整个过程实际上就是采用 GBDT 进行特征筛选后，再使用 LR 模型进行预测，从而得到最终的结果。

使用 GBDT+LR 结合的形式进行点击率预测最大的好处在于，它既可以利用 GBDT 对复杂数据进行非线性特征提取和降维，又可以运用 LR 对特征进行加权和融合，提高模型的预测精度。

另外，GBDT+LR 不需要手动选择特征（通过 GBDT 自动选择），使得模型更具有鲁棒性和可扩展性。

同时，GBDT+LR 具有良好的可解释性，可以通过分析 GBDT 中的特征重要度和 LR 中的权重，得到每个特征在模型中的贡献程度，从而更好地理解模型的预测结果。

虽然 GBDT+LR 来进行点击率预测有很多的好处，但是同时也有很多的问题，比如下面三点。

1. GBDT+LR 建模复杂度较高，需要调节多个超参数，如 GBDT 中的树深度、叶子节点数量、学习率等，LR 中的正则化参数等，增加了模型调优的难度。
2. GBDT+LR 需要大量的数据和计算资源进行训练，尤其是对于大规模的数据集，训练时间和计算成本都较高。

3. GBDT+LR 对异常值和噪声数据敏感，需要进行数据预处理和异常值处理，以提高模型的稳定性和鲁棒性。

GBDT+LR	
优势	问题
提高模型的预测精度	建模复杂度高
使得模型更具有鲁棒性和可扩展性	训练时间和计算成本高
具有良好的可解释性	对异常值和噪声数据敏感

实现一个 GBDT+LR 算法

接下来我们来看一下怎么实现一个 GBDT+LR 算法。由于目前没有比较好的生产数据，暂时使用 Movielens 1MB 数据集，后续你可以根据推荐系统实际的数据来做数据集。

我们直接看代码。

复制代码

```
import pandas as pd
import numpy as np
import lightgbm as lgb
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_sco


# 读取数据集
ratings = pd.read_csv("ml-1m/ratings.dat", sep="::", header=None, names=["user_id
movies = pd.read_csv("ml-1m/movies.dat", sep="::", header=None, names=["movie_id"
data = pd.merge(ratings, movies, on="movie_id").drop(columns=["timestamp", "title
```

```
12
13 # 将电影的genres字段转换为多个二值型变量（使用pandas的get_dummies函数）
14 genres_df = data.genres.str.get_dummies(sep="|")
15 data = pd.concat([data, genres_df], axis=1).drop(columns=["genres"])
16
17 # 提取出用于训练 GBDT 模型和 LR 模型的特征和标签
18 features = data.drop(columns=["user_id", "movie_id", "rating"])
19 label = data["rating"]
20
21 # 划分训练集、测试集
22 split_index = int(len(data) * 0.8)
23 train_x, train_y = features[:split_index], label[:split_index]
24 test_x, test_y = features[split_index:], label[split_index:]
25
26 # 训练 GBDT 模型
27 gbdt_model = lgb.LGBMRegressor(n_estimators=100, max_depth=5, learning_rate=0.1)
28 gbdt_model.fit(train_x, train_y)
29 gbdt_train_leaves = gbdt_model.predict(train_x, pred_leaf=True)
30 gbdt_test_leaves = gbdt_model.predict(test_x, pred_leaf=True)
31
32 # 将 GBDT 输出的叶子节点 ID 转换为 one-hot 编码的特征
33 train_new_feats = lgb.Dataset(gbdt_train_leaves)
34 test_new_feats = lgb.Dataset(gbdt_test_leaves)
35 one_hot_train = train_new_feats.construct_feature_matrix(train_new_feats.data).to
36 one_hot_test = test_new_feats.construct_feature_matrix(test_new_feats.data).toarr
37
38 # 训练 LR 模型
39 lr_model = LogisticRegression()
40 lr_model.fit(one_hot_train, train_y)
41
42 # 在测试集上评估模型性能
43 y_pred = lr_model.predict(one_hot_test)
44 print(f"Accuracy: {accuracy_score(test_y, y_pred)}")
45 print(f"Precision: {precision_score(test_y, y_pred, average='macro')}")
46 print(f"Recall: {recall_score(test_y, y_pred, average='macro')}")
47 print(f"F1-Score (macro): {f1_score(test_y, y_pred, average='macro')}")
```


解释下上面的代码。

1. 从文件中读取电影评级数据集和电影数据集，然后将它们按电影 ID 进行合并。
2. 将电影的 genres 字段转换为多个二值型变量，得到包含二进制特征的数据集。
3. 将数据集按照 8:2 的比例分为训练集和测试集，并分别提取特征和标签。

4. 使用 LightGBM 训练 GBDT 模型，得到训练集和测试集输出的叶节点 ID。
5. 将训练集和测试集节点 ID 转换成 One-Hot 编码的特征矩阵。
6. 使用 LogisticRegression 训练 LR 模型，得到预测结果。
7. 采用准确度，精确度，召回率和 F1 得分作为评价指标进行模型性能评估，并显示评估结果。

相关数据集你可以在这个网址中找到。

总结

到这里这节课就学完了，下面我们来总结一下本节课的内容，这节课主要讲了下面五个点。

1. 逻辑回归是一种广义线性模型，属于分类算法。它通常用于对二元分类问题建模，即将数据分为两个相互排斥的类别。
2. GBDT 是一种集成学习算法，基本思路是通过迭代地训练多个弱学习器（例如决策树），每次训练都会调整样本的权重和残差，使得前一轮弱学习器的误差被后续模型进行更好地拟合。
3. GBDT+LR 模型在推荐系统中应用最为广泛的就是点击率预估（CTR，Click-through rate prediction）问题，即根据用户的历史行为（比如浏览历史、购买记录等）和当前的环境（比如时间、地点、设备等），预测用户是否会对推荐的商品产生兴趣并进行点击的概率。
4. GBDT+LR 模型的优点在于能够充分利用 GBDT 模型的自适应学习特性，自动刻画大量的特征组合。同时，对于 TF-IDF 特征或者 One-Hot 编码的稀疏特征等处理方式，GBDT+LR 能够更好地处理。
5. 最后我们还用一个例子讲解了如何使用 Python 在公开数据集上实现一个 GBDT+LR。

思考题

学完本节课，我给你留两道思考题。

1. 你认为 GBDT 和 LR 分别用于 GBDT+LR 算法的哪些方面？

2. 什么情况下使用 GBDT+LR 算法?

期待在留言区看到你的想法，如果你觉得这门课对你有帮助，也欢迎分享给朋友们一起学习！

精选留言 (2)

麦克范儿

2023-06-11 来自加拿大

老师您好，感谢这讲非常详细的解释。有个关于算法使用的问题想请教一下，GBDT + LR 这类机器学习算法和协同过滤这类的传统算法的区别在于说机器学习将推荐问题从相似度计算和后续的用户 - 物品匹配问题转化为传统的机器学习的二分类问题吗？另外我看有些网站上面没有将这些算法按照召回和排序进行分类，而是都进行了平行分类，因此想了解下是否咱们介绍过的协同过滤和GBDT + LR这些算法都有召回和排序并推荐的作用呢？谢谢您指点！


peter

2023-06-10 来自北京

Q1：模型训练的时候需要大量机器，模型确定以后用于计算则并不需要大量机器，是这样吗?

Q2：GBDT+LR，这种算法用的多吗？实际效果怎么样？其消耗的硬件资源多吗?