推荐系统（5）基于LR+XGBoost预估电信客户流失

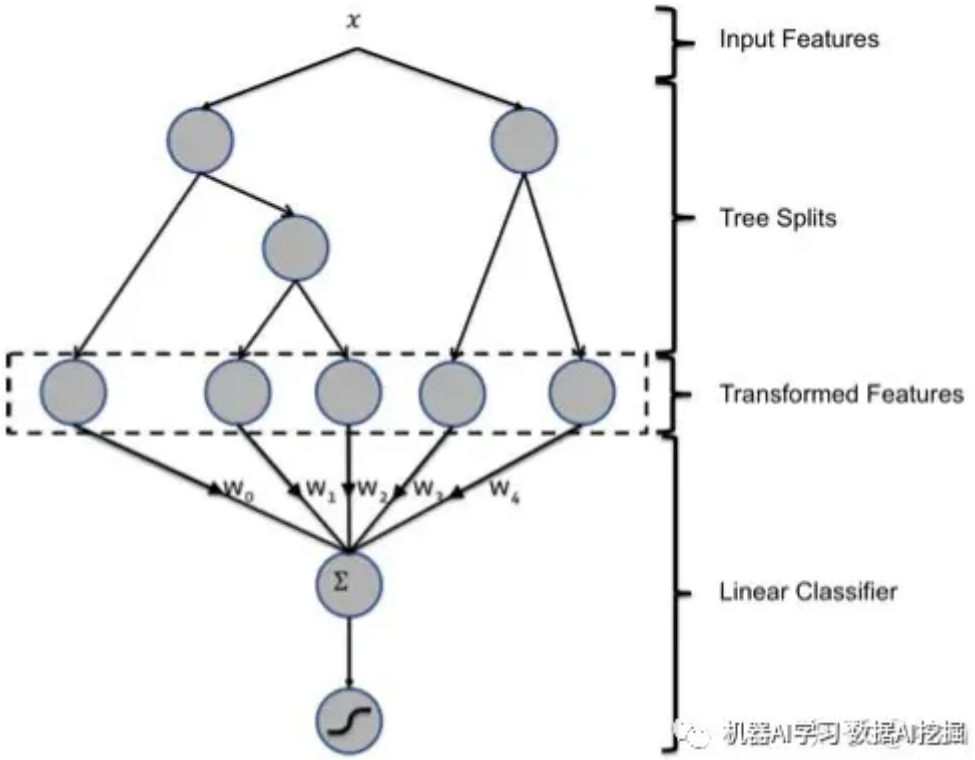
推荐算法 机器AI学习 数据AI挖掘 8月3日

想了解更多好玩的人工智能应用，请关注公众号“机器AI学习 数据AI挖掘”，”智能应用"菜单中包括：颜值检测、植物花卉识别、文字识别、人脸美妆等有趣的智能应用。。

一、算法背景

推荐算法CTR的问题，首先LR属于线性模型，需要大量的特征工程增强模型学习能力，GBDT是非线性模型，具有区分特性及特征组合。因此，我们通过GBDT生成多棵树选取新的特征，然后加入到LR特征的方式来建模学习，来综合彼此的优点，业界（谷歌、百度、kaggle等）取得了不错的效果。

GBDT和LR的融合方案，FaceBook的公开的paper中有个例子：



二、LR+GBDT的特点与方案

1、为什么建树采用ensemble决策树?

一棵树的表达能力很弱，不足以表达多个有区分性的特征组合，多棵树的表达能力更强一些。GBDT每棵树都在学习前面棵树尚存的不足，迭代多少次就会生成多少颗树。按paper以及Kaggle竞赛中的GBDT+LR融合方式，多棵树正好满足LR每条训练样本可以通过GBDT映射成多个特征的需求。

2、为什么建树采用GBDT而非RF?

RF也是多棵树，但从效果上有实践证明不如GBDT。且GBDT前面的树，特征分裂主要体现对多数样本有区分度的特征；后面的树，主要体现的是经过前N颗树，残差仍然较大的少数样本。优先选用在整体上有区分度的特征，再选用针对少数样本有区分度的特征，思路更加合理，这应该也是用GBDT的原因。

然而，Facebook和Kaggle竞赛的思路是否能直接满足现在CTR预估场景呢?

按照Facebook、Kaggle竞赛的思路，不加入广告侧的AD ID特征？但是现CTR预估中，AD ID类特征是很重要的特征，故建树时需要考虑AD ID。直接将AD ID加入到建树的feature中？但是AD ID过多，直接将AD ID作为feature进行建树不可行。下面第三部分将介绍针对现有CTR预估场景GBDT+LR的融合方案。

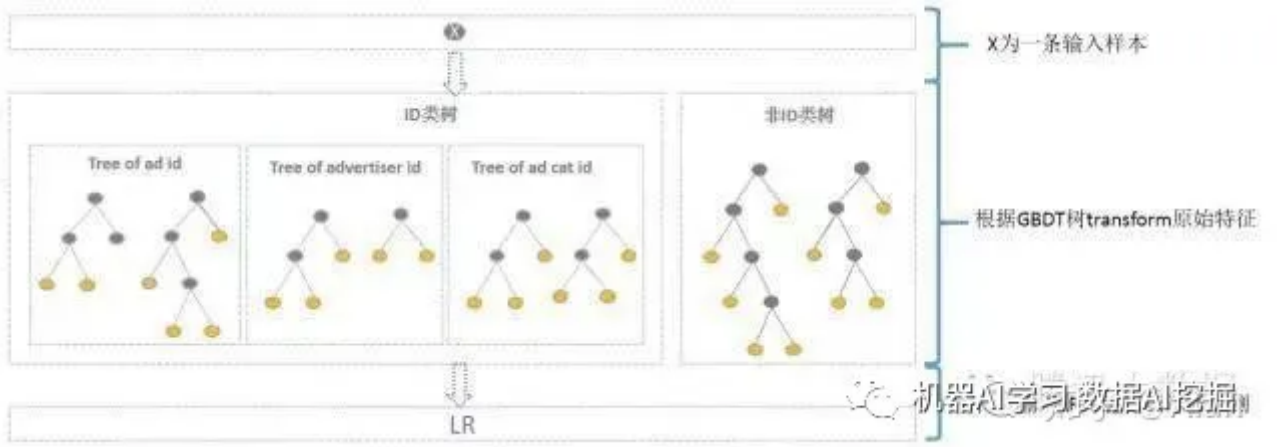
3、GBDT与LR融合方案

AD ID类特征在CTR预估中是非常重要的特征，直接将AD ID作为feature进行建树不可行，故考虑为每个AD ID建GBDT树。但互联网时代长尾数据现象非常显著，广告也存在长尾现象，

为了提升广告整体投放效果，不得不考虑长尾广告。在GBDT建树方案中，对于曝光充分训练样本充足的广告，可以单独建树，发掘对单个广告有区分度的特征，但对于曝光不充分样本不充足的长尾广告，无法单独建树，需要一种方案来解决长尾广告的问题。

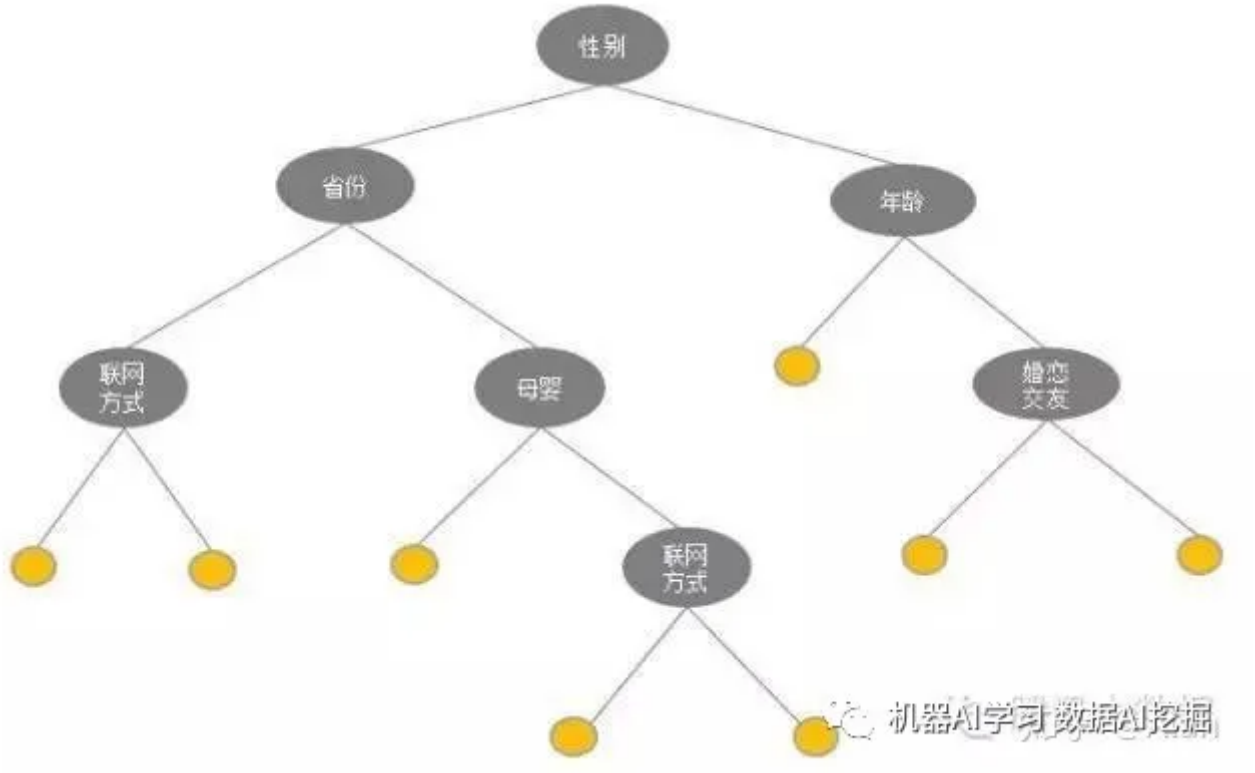
综合考虑方案如下，使用GBDT建两类树，非ID建一类树，ID建一类树。

> 非ID类树(gbdt的输入特征不含id类)：不以细粒度的ID建树，此类树作为base，即便曝光少的广告、广告主，仍可以通过此类树得到有区分性的特征、特征组合。
> ID类树：以细粒度 的ID建一类树，用于发现曝光充分的ID对应有区分性的特征、特征组合。如何根据GBDT建的两类树，对原始特征进行映射？以如下图3为例，当一条样本x进来之后，遍历两类树到叶子节点，得到的特征作为LR的输入。当AD曝光不充分不足以训练树时，其它树恰好作为补充。



通过GBDT 映射得到的特征空间维度如何？

GBDT树有多少个叶子节点，通过GBDT得到的特征空间就有多大。如下图4一颗树，一个叶子节点对应一种有区分性的特征、特征组合，对应LR的一维特征。这颗树有8个叶子节点（黄色点），即对应LR 的8维特征。估算一下，通过GBDT转换得到的特征空间较低，Base树、ID树各N颗，特征空间维度最高为N+N*广告数+N*广告主数+ N*广告类目数。其中广告数、广告主数、广告类目数都是有限的，同时参考Kaggle竞赛中树的数目N最多为30，则估算通过GBDT映射得到的特征空间维度并不高，且并不是每个ID训练样本都足以训练多颗树，实际上通过GBDT 映射得到的特征空间维度更低。



如何使用GBDT 映射得到的特征?

通过GBDT生成的特征，可直接作为LR的特征使用，省去人工处理分析特征的环节，LR的输入特征完全依赖于通过GBDT得到的特征。此证明，通过实验发现GBDT+LR在曝光充分的广告上确实有效果，但整体效果需要权衡优化各类树的使用。

三、案例分析

链接:

https://pan.baidu.com/s/14Zd2zNlCPFOHSr0rNATzPQ
pan.baidu.com

密码:eoxr

```
# -*-coding:utf-8-*-
"""
    Author: Alan
    Desc:
        GBDT+LR模型 电信客户流失预测
```

```
"""
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

class ChurnPredWithGBDTAndLR:
    def __init__(self):
        self.file = "data/new_churn.csv"
        self.data = self.load_data()
        self.train, self.test = self.split()

    # 加载数据
    def load_data(self):
        return pd.read_csv(self.file)

    # 拆分数据集
    def split(self):
        train, test = train_test_split(self.data, test_size=0.1, random_sta
        return train, test

    # 模型训练
    def train_model(self):
        lable = "Churn"
        ID = "customerID"
        x_columns = [x for x in self.train.columns if x not in [lable, ID]]
        x_train = self.train[x_columns]
        y_train = self.train[lable]

        # 创建gbdt模型 并训练
        gbdt = GradientBoostingClassifier()
        gbdt.fit(x_train, y_train)

        # 模型融合
        gbdt_lr = LogisticRegression()
        enc = OneHotEncoder()
        print(gbdt.apply(x_train).shape)
        print(gbdt.apply(x_train).reshape(-1,100).shape)

        # 100为n_estimators，迭代次数
```

```
        enc.fit(gbdt.apply(x_train).reshape(-1,100))
        gbdt_lr.fit(enc.transform(gbdt.apply(x_train).reshape(-1,100)),y_tr

        return enc, gbdt, gbdt_lr

    # 效果评估
    def evaluate(self,enc,gbdt,gbdt_lr):
        lable = "Churn"
        ID = "customerID"
        x_columns = [x for x in self.test.columns if x not in [lable, ID]]
        x_test = self.test[x_columns]
        y_test = self.test[lable]

        # gbdt 模型效果评估
        gbdt_y_pred = gbdt.predict_proba(x_test)
        new_gbdt_y_pred = list()
        for y in gbdt_y_pred:
            # y[0] 表示样本label=0的概率 y[1]表示样本label=1的概率
            new_gbdt_y_pred.append(1 if y[1] > 0.5 else 0)
        print("GBDT-MSE: %.4f" % mean_squared_error(y_test, new_gbdt_y_pred
        print("GBDT-Accuracy : %.4g" % metrics.accuracy_score(y_test.values
        print("GBDT-AUC Score : %.4g" % metrics.roc_auc_score(y_test.values

        gbdt_lr_y_pred = gbdt_lr.predict_proba(enc.transform(gbdt.apply(x_t
        new_gbdt_lr_y_pred = list()
        for y in gbdt_lr_y_pred:
            # y[0] 表示样本label=0的概率 y[1]表示样本label=1的概率
            new_gbdt_lr_y_pred.append(1 if y[1] > 0.5 else 0)
        print("GBDT_LR-MSE: %.4f" % mean_squared_error(y_test, new_gbdt_lr_
        print("GBDT_LR-Accuracy : %.4g" % metrics.accuracy_score(y_test.val
        print("GBDT_LR-AUC Score : %.4g" % metrics.roc_auc_score(y_test.val

if __name__ == "__main__":
    pred = ChurnPredWithGBDTAndLR()
    enc, gbdt, gbdt_lr = pred.train_model()
    pred.evaluate(enc, gbdt,gbdt_lr)
```

四、总结与展望

点击率预估模型涉及的训练样本一般是上亿级别，样本量大，模型采用速度较快的LR+GBDT减少特征工程的人力成本，且业界现在已有实践，在实际落地是可行的。但LR+GBDT已也存在缺点，离线处理和在线处理都复杂，上线测试效果、调参工程复杂耗时（人生苦短却必须）。现目前FM、DeepFM、DIEN等深度学习推荐系统等优秀算法思路还等待我们去挖掘使用应用落地。最后，不同场景下GBDT+LR的思路可能会略有不同，请根据实际情况多种角度尝试。