GBDT+LR算法进行特征扩增

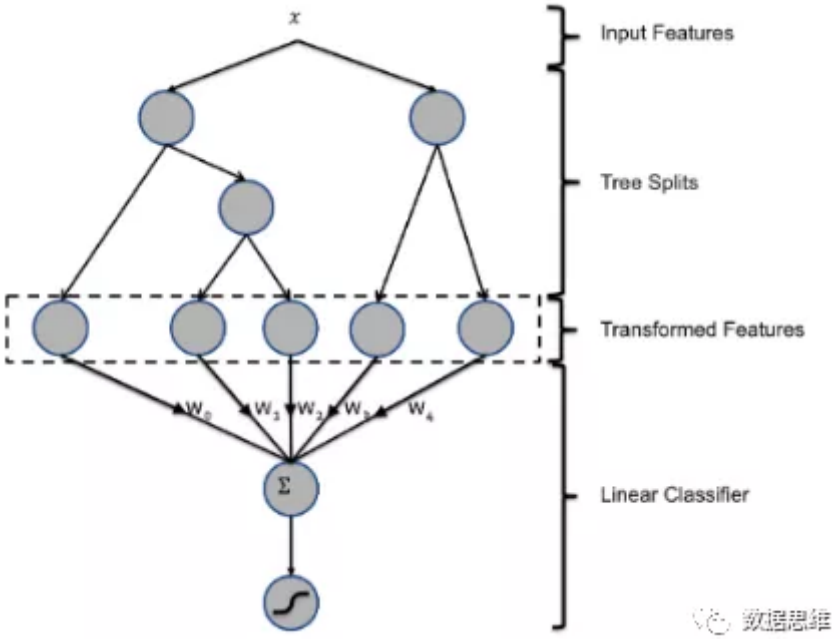
数据思维 2019-10-02

from:http://blog.csdn.net/TwT520Ly https://blog.csdn.net/TwT520Ly/article/details/79769705
参考文献:
https://blog.csdn.net/lilyth_lilyth/article/details/48032119
https://blog.csdn.net/asdfghjkl1993/article/details/78606268

0.简介

CTR估计也就是广告点击率预估，计算广告训练与平滑思想说明了是用LR算法对于预测的有效性。 `LR（Logistic Regression）` 是广义线性模型，与传统线性模型相比，LR通过 `Logit` 变换将函数值映射到0~1区间，映射后的函数就是CTR的预估值。LR模型十分适合并行化，因此对于大数据的训练十分有效。但是对于线性模型而言，学习能力是有限的，因此需要大量的特征工程预先分析出有效的特征或者是特征组合，从而去间接的增强LR的非线性学习能力。

特征组合，是通过特征的一些线性叠加或者非线性叠加得到一个新的特征，可以有效的提高分类效果。常见的特征组合方式有笛卡尔积方式。为了降低人工组合特征的工作量，FaceBook提出了一个自动特征提取的方式 `GBDT+LR` 。



GBDT是梯度提升决策树，首先会构造一个决策树，首先在已有的模型和实际样本输出的残差上再构造一颗决策树，不断地进行迭代。每一次迭代都会产生一个增益较大的分类特征，因此GBDT树有多少个叶子节点，得到的特征空间就有多大，并将该特征作为LR模型的输入。

1.核心问题

（1）建树采用ensemble决策树?

一棵树的区分性是具有一定的限制的，但是多棵树可以获取多个具有区分度的特征组合，而且GBDT的每一棵树都会学习前面的树的不足。

（2）建树算法为什么采用GBDT而不是RF?

对于GBDT而言，前面的树，特征分裂主要体现在对多数样本的具有区分度的特征；后面的树，主要体现的是经过前面n棵树，残差依然比较大的少数样本。优先选用在整体上具有区分度的特征，再选用针对少数样本有区分度的特征。

2.代码实现

导入包

```
import numpy as np
import random
import xgboost as xgb
import matplotlib
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifierfrom sklearn.dat
asets import make_classification
from sklearn.linear_model import LogisticRegressionfrom sklearn.metrics import roc_curve, roc_a
uc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoderfrom xgboost.sklearn import XGBClassifier
```

生成随机数据

```
np.random.seed(10)X, Y = make_classification(n_samples=1000, n_features=30)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=233, test_size=0.5)
X_train, X_train_lr, Y_train, Y_train_lr = train_test_split(X_train, Y_train, random_state=233,
test_size=0.2)
```

RandomForest + LogisticRegression

```
def RandomForestLR():
    RF = RandomForestClassifier(n_estimators=100, max_depth=4)
    RF.fit(X_train, Y_train)
    OHE = OneHotEncoder()
    OHE.fit(RF.apply(X_train))
    LR = LogisticRegression()
    LR.fit(OHE.transform(RF.apply(X_train_lr)), Y_train_lr)
    Y_pred = LR.predict_proba(OHE.transform(RF.apply(X_test)))[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, Y_pred)
    auc = roc_auc_score(Y_test, Y_pred)
    print('RandomForest + LogisticRegression: ', auc)
    return fpr, tpr
```

Xgboost + LogisticRegression

```
def XGBoostLR():
    XGB = xgb.XGBClassifier(nthread=4, learning_rate=0.08, n_estimators=100, colsample_bytree=
0.5)
    XGB.fit(X_train, Y_train)
    OHE = OneHotEncoder()
    OHE.fit(XGB.apply(X_train))
    LR = LogisticRegression(n_jobs=4, C=0.1, penalty='l1')
    LR.fit(OHE.transform(XGB.apply(X_train_lr)), Y_train_lr)
    Y_pred = LR.predict_proba(OHE.transform(XGB.apply(X_test)))[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, Y_pred)
    auc = roc_auc_score(Y_test, Y_pred)
    print('XGBoost + LogisticRegression: ', auc)
    return fpr, tpr
```

GradientBoosting + LogisticRegression

```
def GBDTLR():
    GBDT = GradientBoostingClassifier(n_estimators=10)
    GBDT.fit(X_train, Y_train)
    OHE = OneHotEncoder()
    OHE.fit(GBDT.apply(X_train)[:, :, 0])
    LR = LogisticRegression()
    LR.fit(OHE.transform(GBDT.apply(X_train_lr)[:, :, 0]), Y_train_lr)
    Y_pred = LR.predict_proba(OHE.transform(GBDT.apply(X_test)[:, :, 0]))[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, Y_pred)
    auc = roc_auc_score(Y_test, Y_pred)
    print('GradientBoosting + LogisticRegression: ', auc)
    return fpr, tpr
```

LogisticRegression

```
def LR():
    LR = LogisticRegression(n_jobs=4, C=0.1, penalty='l1')
    LR.fit(X_train, Y_train)
    Y_pred = LR.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, Y_pred)
```

```
    auc = roc_auc_score(Y_test, Y_pred)
    print('LogisticRegression: ', auc)
    return fpr, tpr
```

XGBoost

```
def XGBoost():
    XGB = xgb.XGBClassifier(nthread=4, learning_rate=0.08, n_estimators=100, colsample_bytree=
0.5)
    XGB.fit(X_train, Y_train)
    Y_pred = XGB.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, Y_pred)
    auc = roc_auc_score(Y_test, Y_pred)
    print('XGBoost: ', auc)
    return fpr, tpr
```

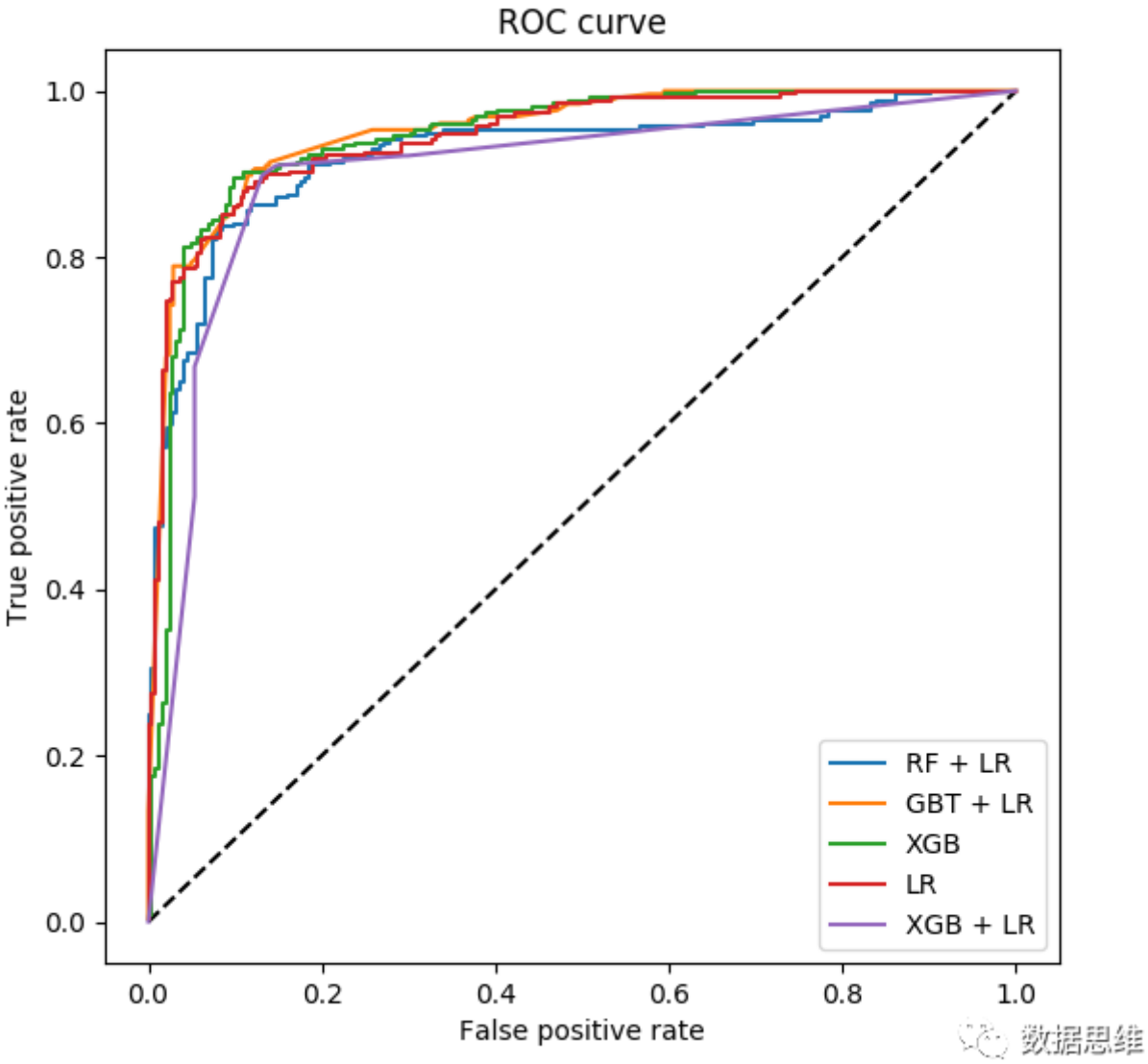
调用并绘制图像

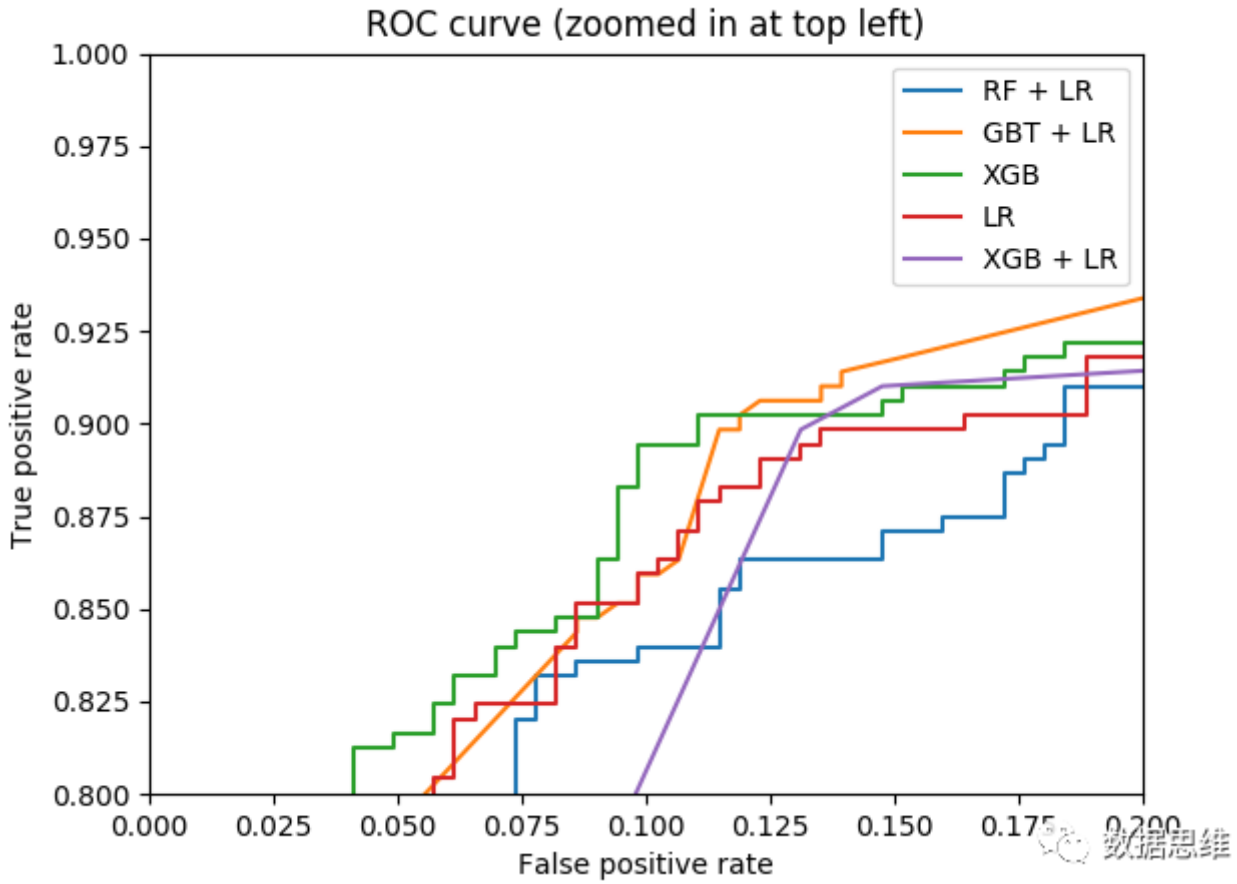
```
if __name__ == '__main__':
    fpr_xgb_lr, tpr_xgb_lr = XGBoostLR()
    fpr_xgb, tpr_xgb = XGBoost()
    fpr_lr, tpr_lr = LR()
    fpr_rf_lr, tpr_rf_lr = RandomForestLR()
    fpr_gbdt_lr, tpr_gbdt_lr = GBDTLR()

    plt.figure(1)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr_rf_lr, tpr_rf_lr, label='RF + LR')
    plt.plot(fpr_gbdt_lr, tpr_gbdt_lr, label='GBT + LR')
    plt.plot(fpr_xgb, tpr_xgb, label='XGB')
    plt.plot(fpr_lr, tpr_lr, label='LR')
    plt.plot(fpr_xgb_lr, tpr_xgb_lr, label='XGB + LR')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(loc='best')
    plt.show()

    plt.figure(2)
    plt.xlim(0, 0.2)
    plt.ylim(0.8, 1)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr_rf_lr, tpr_rf_lr, label='RF + LR')
    plt.plot(fpr_gbdt_lr, tpr_gbdt_lr, label='GBT + LR')
    plt.plot(fpr_xgb, tpr_xgb, label='XGB')
    plt.plot(fpr_lr, tpr_lr, label='LR')
    plt.plot(fpr_xgb_lr, tpr_xgb_lr, label='XGB + LR')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve (zoomed in at top left)')
```

```
    plt.legend(loc='best')
    plt.show()
```





为了更好的服务数据圈内同学，我们需要更多的志愿者，主要协助推广转发，寻找更多更好的内容，有兴趣的同学可以联系L23683716，加志愿者群。

求职招聘，技术交流：