基于AI算法的无线网络邻区关系优化研究与实践

Study and Practice on Optimization of Wireless Network Neighborhood Relationship Based on AI Algorithm

李张铮，陈　锋，董帝烺（中国联通福建省分公司，福建 福州 350000）
Li Zhangzheng，Chen Feng，Dong Dilang（China Unicom Fujian Branch，Fuzhou 350000，China）

摘　要：
针对现有邻区优化方式的不足，基于现网数据引入XGBoost机器学习回归预测算法，通过学习具有自动邻区关系网络的两两小区切换占比建立预测模型，优化非自动邻区关系网络小区邻区关系。研究结果表明，基于AI算法的无线网络邻区关系优化能有效提高邻区优化效率，提升邻区关系的准确性。

关键词：
XGBoost算法；小区切换占比预测；邻区优化；网络优化
doi：10.12045/j.issn.1007-3043.2021.05.015
文章编号：1007-3043（2021）05-0065-07
中图分类号：TN929.5
文献标识码：A
开放科学（资源服务）标识码（OSID）：

Abstract：
Neighborhood optimization is an indispensable and complex part of wireless network optimization，and traditional artificial optimization is a heavy burden on the operation of the operator's huge wireless base station. In view of the deficiency of the existing neighborhood optimization method，according to the existing network data，XGBoost machine learning regression prediction algorithm is introduced，and the neighborhood relationship of non-automatic neighborhood relationship network is optimized by learning the two-cell handover ratio of automatic neighborhood relationship network to establish the prediction model. The results show that the optimization of wireless network neighborhood based on AI algorithm can effectively improve the efficiency of neighborhood relationship optimization and improve the accuracy of neighborhood relationship.

Keywords：
XGBoost algorithm；Cells handover ratio prediction；Neighborhood relationship optimization；Network optimization

引用格式：李张铮，陈锋，董帝烺. 基于AI算法的无线网络邻区关系优化研究与实践[J]. 邮电设计技术，2021(5)：65-71.

0 引言

在无线网络优化工作中，邻区优化是降低掉话率、提升移动网络质量、改善用户感知的最基本且最有效的手段。目前基站邻区优化有2种方式：自动邻区关系（ANR）和非自动邻区关系（下文简称非自动邻区关系网络为传统网络）。具备自动邻区关系的网络可自动优化邻区关系，不需人工干预；传统网络邻区关系需优化人员手工优化。这些网络邻区的数目众多，优化工作量非常大；需要优化邻区的确定与个人优化经验有很大的关系，稍有不慎就可能造成邻区漏配或冗余邻区，存在较大的优化风险。规避上述问题，提高邻区优化的效率和精度已成为网络优化的关键。考虑到在ANR网络中，自动邻区关系已成为小区SON功能的标配，无需人员操作网络便可自动识别和添加邻区，如何利用ANR邻区关系来优化传统网络邻区已成为网络运营智能化的重要课题。

机器学习技术作为人工智能的重要组成部分，是国家发展战略重点扶持的目标和当下各行业关注应用的焦点。为了推动传统网络邻区优化的智能化，提

收稿日期：2021-03-23

升网络运营智能化水平，特开展基于机器学习算法的邻区关系优化的研究。

1 传统无线网络邻区关系优化难点

在传统网络优化中，邻区关系优化一直以来是一个难点。由于邻区关系数量多、影响大、技术要求高、优化手段匮乏等等方面的因素，使得邻区关系优化在传统网络中存在一些挑战。

1.1 邻区关系数量多，导致优化耗时耗力

以福州联通为例，W现网有小区25 000个，用每个小区有20~25条邻区来计算，所配置的邻区个数至少50万条以上，网优人员每周提取MR数据，使用厂家工具进行同频、异频邻区核查，根据核查结果，确定需要优化的邻区，并进行相应操作，邻区优化的工作量非常大。

1.2 邻区关系影响面大，直接关系网络口碑

邻区设置不当，会导致干扰增大、容量下降和性能恶化，严重影响用户感知，引发的掉话等问题会导致用户投诉，给运营商网络口碑带来负面影响，影响NPS得分。邻区设置不当主要有2种表现方式：邻区漏配和冗余邻区。邻区漏配会引起干扰增大，降低用户的通话质量甚至掉话，从而引起容量及覆盖能力下降；冗余邻区一方面将会由于切换的过多会导致信令负荷加重；另一方面由于终端测量能力的限制，会降低测量的精度、增加测量时延。同时信号较多会造成干扰，容易出现掉话，影响速率的提升，从而影响用户感知。

1.3 邻区关系优化手段缺乏，对技术要求高

传统的邻区关系优化手段有2种：基于路测软件分析和基于厂家的邻区核查工具平台。

a）路测软件分析。基于导频的小区切换关系来定位邻区关系合理性，该方法的局限性是路测范围有限，覆盖面不足，无法开展全网的精细邻区优化，且路测方法耗时耗力。

b）基于厂家的邻区核查平台。通过采集UE上报的测量报告、话统呼叫记录、事件进行汇总分析，判断邻区漏配和冗余。该方法有较高的精确性，但受限于厂家License配额和优化人员的技术水平。

2 基于XGBoost算法的小区切换占比预测

随着运营商移动用户数的不断增加，良好的用户网络体验保障对无线网络运营提出了更高的要求。影响无线网络质量的因素很多，其中邻区关系是一个关键因素，它是小区移动性管理的直接承载者。做好邻区关系优化，始终是网优工作的重点。

本文通过利用XGBoost机器学习算法学习ANR现网邻区关系数据，建立小区间切换次数占比模型预测出传统网络小区的邻区关系。该模型可在开站邻区配置、邻区核查、用户投诉分析等网优日常工作中起到积极作用。

2.1 训练集和测试集样本生成

2.1.1 样本的采集

提取某省联通2个行政区LTE网络3天的两两小区切换次数报表，汇总每个小区邻区的切换次数占比降序排列，取每个小区占比前50名的邻区作为样本，切换占比为样本标签，同时关联网络工参相关字段（见表1），形成最终样本。

表1　网络工参字段

字段名称	取值范围	示例
小区经度/°	数值，0~180	119.123 413
小区纬度/°	数值，0~90	26.023 819
小区方位角/°	数值，0~360	30
小区下倾角/°	数值，0~90	10
小区天线挂高/m	数值，大于0	20
小区覆盖类型	字符串，室分或宏站	宏站

2.1.2 样本划分为训练集和测试集

机器学习一般将样本划分为训练集和测试集，训练集用于模型训练，测试集用于测试模型性能。本文利用sciki-learn的train_test_split()函数将样本划分为训练集和测试集，其中参数测试集比例test_size取0.2，即训练集和测试集比例为8∶2。

2.2 数据预处理

数据预处理主要是检查每个特征是否有缺失值或非法字符，对不合理的值进行校正替换。检查样本数据发现，覆盖类型为室分的小区方位角都是0值，这与实际室分小区为全向覆盖不符，故室分小区的方位角需修正。修正方法如下：若室分小区与宏站邻小区同经纬度，则室分小区取宏站邻小区的方位角；若室分小区与室分邻小区同经纬度，则室分小区方位角取值368°；若室分小区与邻小区不同经纬度，则室分小区方位角取室分小区与邻小区连线与正北方向的顺时针夹角r（见图1）。

设室分小区经纬度(X_1, Y_1)，邻小区经纬度(X_2, Y_2)，具体小区连线夹角r计算公式如下：

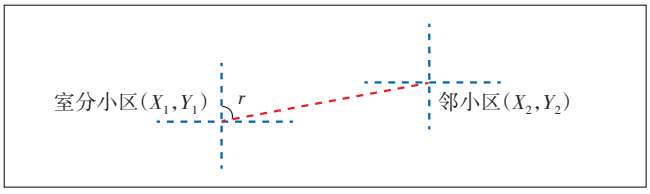


图1　室分小区方位角定义

$$x_1 = X_1 \times \frac{\pi}{180}, y_1 = Y_1 \times \frac{\pi}{180}, x_2 = X_2 \times \frac{\pi}{180}, y_2 = Y_2 \times \frac{\pi}{180} \quad (1)$$

$$r = \begin{cases} 180, & x_1 = x_2, y_1 > y_2 \\ 0, & x_1 = x_2, y_1 < y_2 \end{cases} \quad (2)$$

为了便于书写，令

$$A = \frac{\sqrt{4\left(\sin\frac{y_1 - y_2}{2}\right)^2 - \left[\sin\frac{x_1 - x_2}{2} \times \left(\cos y_1 - \cos y_2\right)^2\right]}}{\sin\frac{|x_1 - x_2|}{2} \times \left(\cos y_1 + \cos y_2\right)} \quad (3)$$

$$r = \begin{cases} 270 - \frac{\tan^{-1}(A)}{\pi} \times 180, & x_1 > x_2, y_1 > y_2 \\ 270 + \frac{\tan^{-1}(A)}{\pi} \times 180, & x_1 > x_2, \text{其他} \end{cases} \quad (4)$$

$$r = \begin{cases} 90 + \frac{\tan^{-1}(A)}{\pi} \times 180, & x_1 < x_2, y_1 > y_2 \\ 90 - \frac{\tan^{-1}(A)}{\pi} \times 180, & x_1 < x_2, \text{其他} \end{cases} \quad (5)$$

图2给出了室分小区方位角特征预处理过程。

2.3 特征工程

特征工程是机器学习过程的重要环节，样本特征的好坏决定了机器学习性能的上限，而模型只是逼近这个上限而已。特征工程的主要内容包括特征构造、特征抽取和特征选择。本文的原始特征包括本地/目标小区经纬度、本地/目标小区方位角度、本地/目标小区合计下倾角度及本地/目标小区天线挂高10个维度。为了满足特征选择的需要，在此基于本地/目标小区的经纬度构造额外的特征，主要包括haversine距离、两经纬度的方位角、经纬度PCA分量，最后进行特征选择。

2.3.1 haversine距离

haversine公式是计算球面两点间距离的一种方法，该方法采用了正弦函数，即使距离很小，也能保持足够的有效数字。haversine距离计算公式如下：

$$\text{haversine}(d/R) = \text{haversine}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\text{haversine}(\Delta\theta) \quad (6)$$

其中，$\text{haversine}(\alpha) = \sin^2(\alpha/2) = (1 - \cos\alpha)/2$，$R$为地球半径，取6 371 km，$\varphi_1$、$\varphi_2$表示2点的纬度，$\Delta\theta$表示2点经度的差值。代码实现如图3所示。

2.3.2 两经纬度的方位角

设本地小区经纬度为(lat_1,lng_1)，目标小区经纬度为(lat_2,lng_2)，两经纬度间的方位角公式计算如下，代码实现如图4所示。

$$\text{bearing} = \tan^{-1}\left(\frac{\sin(\text{lng}_2 - \text{lng}_1)\cos(\text{lat}_2)}{\cos(\text{lat}_1)\sin(\text{lat}_2) - \sin(\text{lat}_1)\cos(\text{lat}_2)\cos(\text{lng}_2 - \text{lng}_1)}\right) \quad (7)$$

```
for idx,df_idx in enumerate(df_indoor_cells.index.to_list()):
    if (long_arc_from[df_idx]==long_arc_to[df_idx])&(lat_arc_from[df_idx]>lat_arc_to[df_idx]):
      df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=180
    elif (long_arc_from[df_idx]==long_arc_to[df_idx])&(lat_arc_from[df_idx]<lat_arc_to[df_idx]):
      df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=0
    elif (long_arc_from[df_idx]==long_arc_to[df_idx])&(lat_arc_from[df_idx]==lat_arc_to[df_idx]):
     if(df_indoor_cells.loc[df_idx,df_indoor_cells.columns[5]]="室外")
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]= df_indoor_cells.loc[df_idx,df_indoor_cells.columns[6]] #若同一
     else:
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]= 368  #若同一点上邻区为室分小区，则用368表示该邻区
    elif (long_arc_from[df_idx]>long_arc_to[df_idx])&(lat_arc_from[df_idx]>lat_arc_to[df_idx]):
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=270- np.arctan((sub1[idx]-sub2[idx])/(mul1[idx]×mul2[idx]))×180/
    elif (long_arc_from[df_idx]>long_arc_to[df_idx])&(lat_arc_from[df_idx]≤lat_arc_to[df_idx]):
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=270+ np.arctan((sub1[idx]-sub2[idx])/(mul1[idx]×mul2[idx]))×180/
    elif (long_arc_from[df_idx]<long_arc_to[df_idx]&(lat_arc_from[df_idx]>lat_arc_to[df_idx]):
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=90+ np.arctan((sub1[idx]-sub2[idx])/(mul1[idx]×mul2[idx]))×180/
    elif (long_arc_from[df_idx]<long_arc_to[df_idx])&(lat_arc_from[df_idx]≤lat_arc_to[df_idx]):
        df_indoor_cells.loc[df_idx,df_indoor_cells.columns[4]]=90- np.arctan((sub1[idx]-sub2[idx])/(mul1[idx]×mul2[idx]))×180/
```

图2　室分小区方位角特征预处理

```
def haversine_distance(df_long_lat):
    lat1=df_long_lat.iloc[:,0].values
    lng1=df_long_lat.iloc[:,1].values
    lat2=df_long-lat.iloc[:,2].values
    lng2=df_long_lat.iloc[:,3].values
    lat1,lng1,lat2,lng2=map(np.radians,(lat1,lng1,lat2,lng2))
    AVG_EARTH_Radius=6371 # in km
    lat=lat2-lat1
    lng=lng2-lng1
    d=np.sin(lat×0.5)××2+np.cos(lat1)×np.cos(lat2)×np.sin(lng×0.5)×2
    h=2×AVG_EARTH_Radius×np.arcsin(np.sqrt(d))

    return h
```

图3　本地/目标小区haversine距离计算

```
def bearing_degree(df_long_lat):
    lat1=df_long_lat.iloc[:,0].values
    lng1=df_long_lat.iloc[:,1].values
    lat2=df-long-lat.iloc[:,2].values
    lng2=df-long-lat.iloc[:,3].values
    AVG_EARTH_Radius=6371 # in km
    lng_delta_rad=np.radians(lng2-lng1)
    lat1,lng1,lat2,lng2=map(np.radians,(lat1,lng1,lat2,lng2))
    y=np.sin(lng_delta_red)×np.cos(lat2)
    x=np.cos(lat1)×np.sin(lat2)-np.sin(lat1)×np.cos(lat2)×
                                        np.cos(lng_delta_rad)
    result=np.degrees(np.arctan2(y,x))
    result[result<0]=result[result<0]+360

    return result
```

图4　2个经纬度间的方位角计算

2.3.3 经纬度的PCA分量

主成分分析(PCA——Principal Component Analysis)是最广泛的数据压缩算法，主要通过降维可以生成更便于人理解的新特征，加快对样本有价值信息的处理速度。此处对本地/目标小区经纬度4个特征采用PCA进行变换，默认降维后的特征数仍为4(见图5)。

2.3.4 特征/目标相关性分析

特征选择不仅具有减少特征数量(降维)、减少过拟合、提高模型泛化能力等优点，而且还可以使模型获得更好的解释性，增强对特征和特征值、特征和目标之间关系的理解，加快模型的训练速度获得更好的预测性能。此处采用pandas的相关系数计算函数corr()来分析特征和目标间的相关性(见图6和表2)。

从热力图上可以发现，部分特征间的相关性过高，这将造成特征间的多重共线性，影响模型效果，这里剔除相关系数大于0.8的特征(包括本地小区LATITUDE，本地小区LATITUDE_pca_0，本地小区LONGITUDE)，保留与目标相关性最大的特征。

2.3.5 特征标准化

特征标准化就是将某列特征的值缩放到均值为0，方差为1的状态，计算公式为$z=(x-\mu)/\sigma$。标准化的好处是提升模型精度和加快收敛速度。此处使用scikit-learn自带的StandardScaler()类进行转换。

2.4 模型训练

2.4.1 基于交叉验证的回归预测模型选择

机器学习中常用的回归预测模型有线性回归、KNN、随机森林、GBDT和XGBoost等。这里分别使用这几个模型进行交叉验证打分，选出最好的模型。这些模型的参数都取默认值，交叉验证参数取5，评估标准为平均绝对误差MAE。实验结果表明，最好的模型为XGBoost，平均cross_val_score得分最高为-0.03(见图7)。下面就使用XGBoost模型进行建模训练。

2.4.2 XGBoost算法原理概述

XGBoost算法近年来在工业界和各类数据挖掘竞赛中大放异彩，取得良好的预测效果。与传统的Boosting算法如GBDT比较，XGBoost算法优点在于：GBDT只利用了一阶导数的信息，而XGBoost对损失函数做了二阶泰勒展开，并且在目标函数中加入了正则项，用来权衡目标函数和模型的复杂程度，防止过拟

```
def long_lat_pca_transform(df_long_lat,pca):
    arr_coords=df_long_lat.values
#     pca=PCA().fit(arr_coords)
    df_long_lat[df_long_lat.columns[0]+"_pca_0"]=pca.transform(df_long_lat[[df_long_lat.columns[0],df_long_lat.columns[1],
                                                                            df_long_lat.columns[2],df_long_lat.columns[3]]])[:,0]
    df_long_lat[df_long_lat.columns[1]+"_pca_1"]=pca.transform(df_long_lat[[df_long_lat.columns[0],df_long_lat.columns[1],
                                                                            df_long_lat.columns[2],df_long_lat.columns[3]]])[:,1]
    df_long_lat[df_long_lat.columns[2]+"_pca_2"]=pca.transform(df_long_lat[[df_long_lat.columns[0],df_long_lat.columns[1],
                                                                            df_long_lat.columns[2],df_long_lat.columns[3]]])[:,2]
    df_long_lat[df_long_lat.columns[3]+"_pca_3"]=pca.transform(df_long_lat[[df_long_lat.columns[0],df_long_lat.columns[1],
                                                                            df_long_lat.columns[2],df_long_lat.columns[3]]])[:,3]
    return df_long_lat.loc[:,[df_long_lat.columns[0]+"_pca_0",df_long_lat.columns[1]+"_pca_1",
                              df_long_lat.columns[2]+"_pca_2",df_long_lat.columns[3]+"_pca_3"]]
```

图5　本地/目标小区经纬度的PCA变换

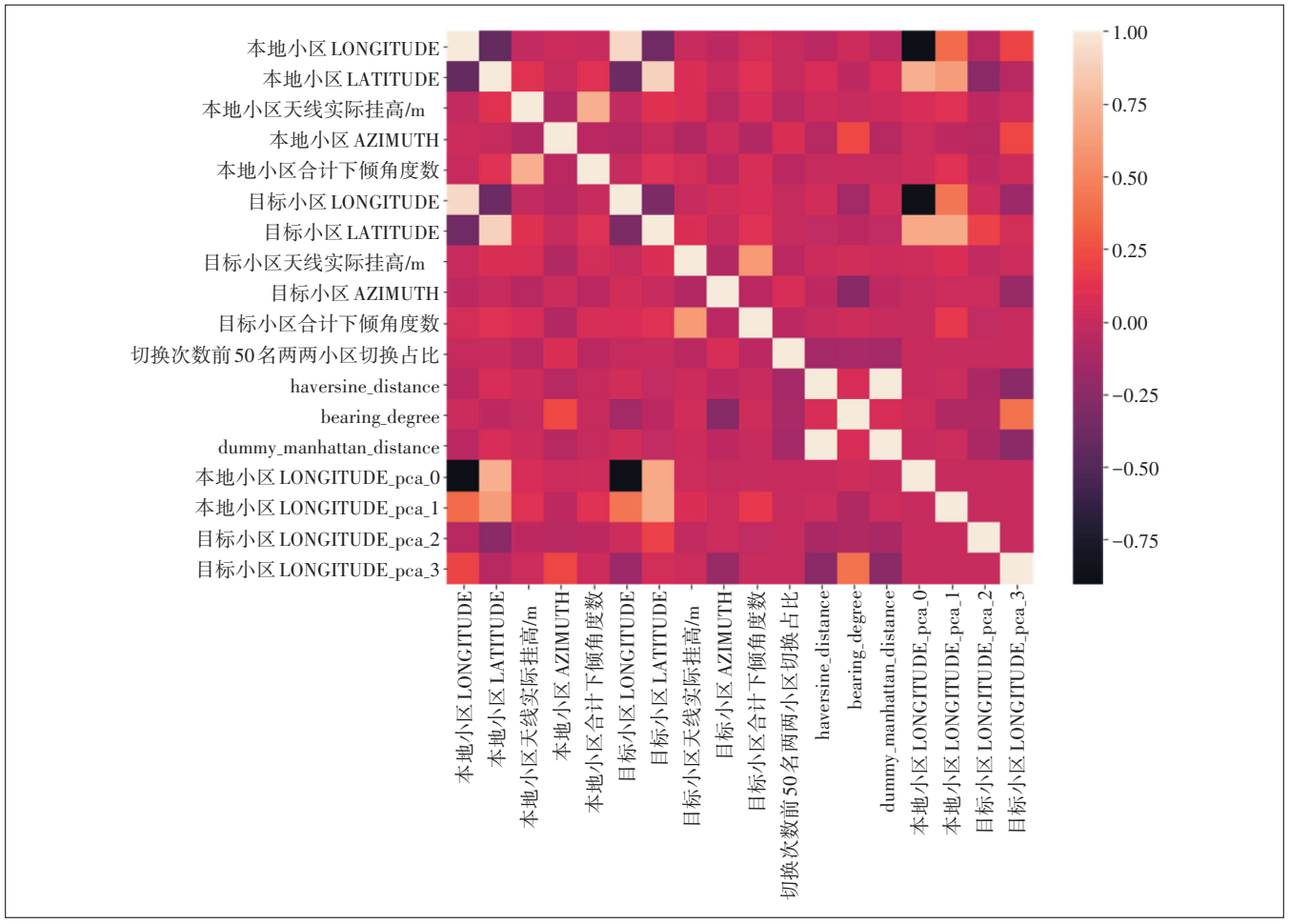


图6　特征和目标间的相关性热力图

表2　特征和目标间的相关系数值

特征/目标	相关系数	特征/目标	相关系数
切换次数前50名两两小区切换占比	1	本地小区 LONGITUDE	−0.003 803
本地小区 AZIMUTH	0.079 165	目标小区 LONGITUDE	−0.006 654
目标小区 AZIMUTH	0.065 884	目标小区合计下倾角度数	−0.037 084
目标小区 LONGITUDE_pca_3	0.007 913	目标小区天线实际挂高/m	−0.046 067
本地小区 LATITUDE_pca_0	0.004 453	本地小区合计下倾角度数	−0.047 737
目标小区 LATITUDE_pca_2	0.002 534	本地小区天线实际挂高/m	−0.048 235
目标小区 LATITUDE	0.001 642	bearing_degree	−0.128 856
本地小区 LATITUDE	0.000 144	haversine_distance	−0.135 591
本地小区 LONGITUDE_pca_1	−0.003 323		

合；Boosting是串行过程，不能并行化且计算复杂度较高，也不适合高维稀疏特征，而XGBoost在特征粒度上可进行并行化计算且考虑了训练数据为稀疏值的情况。该算法原理如下：

XGBoost算法的目标函数包含损失函数L和正则化项Ω：

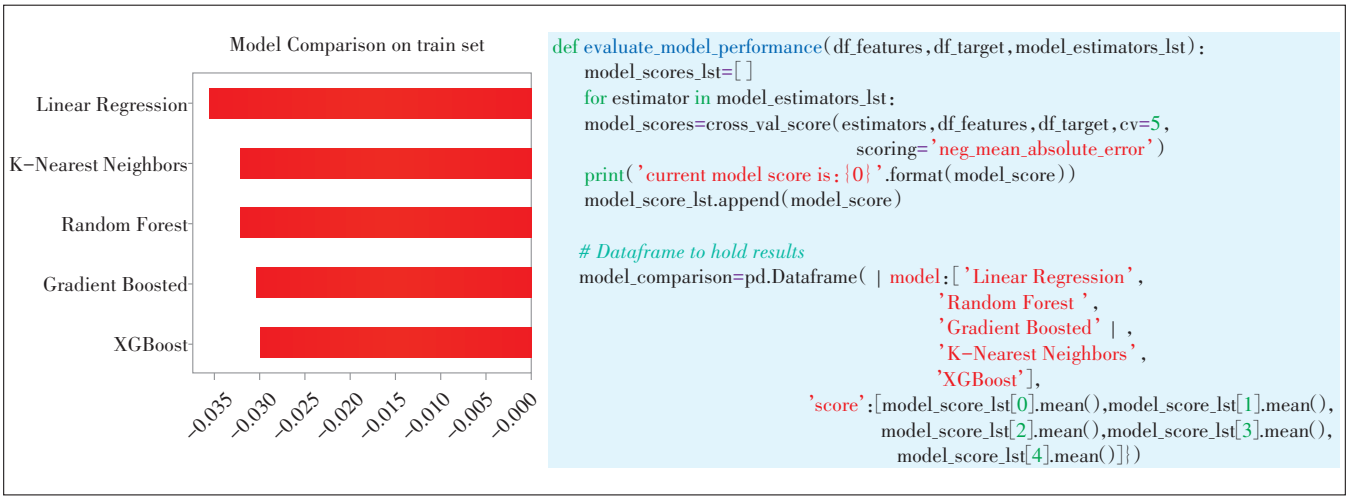
$$\mathrm{Obj}=\sum_{i=1}^{n}l\left(\hat{y}_i,y_i\right)+\sum_{t=1}^{k}\Omega\left(f_t\right) \tag{8}$$

根据第t步的新模型的预测值$f_t\left(x_i\right)$，此时的目标函数可写成：

$$\mathrm{Obj}^{(t)}=\sum_{i=1}^{n}l\left(y_i,\hat{y}_i^{t}\right)+\sum_{i=1}^{t}\Omega\left(f_i\right)=$$
$$\sum_{i=1}^{n}l\left(y_i,\hat{y}_i^{t-1}+f_t\left(x_i\right)\right)+\sum_{i=1}^{t}\Omega\left(f_i\right) \tag{9}$$

利用泰勒公式将目标函数进行泰勒二阶展开，得

$$\mathrm{Obj}^{(t)}=\sum_{i=1}^{n}\left[l\left(y_i,\hat{y}_i^{t-1}\right)+g_if_t\left(x_i\right)+\frac{1}{2}h_if_t^2\left(x_i\right)\right]+$$
$$\sum_{i=1}^{t}\Omega\left(f_i\right) \tag{10}$$




```
def evaluate_model_performance(df_features,df_target,model_estimators_lst):
    model_scores_lst=[]
    for estimator in model_estimators_lst:
    model_scores=cross_val_score(estimators,df_features,df_target,cv=5,
                                 scoring='neg_mean_absolute_error')
    print('current model score is:{0}'.format(model_score))
    model_score_lst.append(model_score)

  # Dataframe to hold results
  model_comparison=pd.Dataframe( | model:['Linear Regression',
                                          'Random Forest ',
                                          'Gradient Boosted' | ,
                                          'K-Nearest Neighbors',
                                          'XGBoost'],
                                 'score':[model_score_lst[0].mean(),model_score_lst[1].mean(),
                                          model_score_lst[2].mean(),model_score_lst[3].mean(),
                                          model_score_lst[4].mean()]})
```

图7　基于交叉验证的回归模型选择

其中g_i为损失函数一阶导数，h_i为损失函数的二阶导数。当损失函数取平方损失时，目标函数近似为：

$$\mathrm{Obj}^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\sum_{i=1}^{t}\Omega(f_i) \quad (11)$$

进一步地，基函数取为决策树模型$f_t(x)=\omega_{q(x)}$，$q(x)$表示样本x所在的叶子节点，同时设决策树叶子节点数为T，该值决定了决策树的复杂度，值越大模型越复杂，此时目标函数的正则项表示为：

$$\Omega(f_t)=\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2 \quad (12)$$

由于每个样本x_i最终都是落在叶子节点上，且每个叶子节点都会包含多个样本，因此遍历所有样本x_i求损失函数等价于遍历所有叶子节点求损失函数，设第j个叶子节点包含的样本集合为$I_j=\{i\}$，则损失函数为：

$$\begin{aligned}Obj^{(t)} &\approx \sum_{i=1}^{n}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\Omega(f_t)=\\ &\sum_{i=1}^{n}\left[g_i\omega_{q(x_i)}+\frac{1}{2}h_i\omega_{q(x_i)}^2\right]+\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2=\\ &\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j}g_i\right)\omega_j+\frac{1}{2}\left(\sum_{i\in I_j}h_i+\lambda\right)\omega_j^2\right]+\gamma T \quad (13)\end{aligned}$$

为简化公式，定义$G_j=\sum_{i\in I_j}g_i$，$H_j=\sum_{i\in I_j}h_i$，目标函数为：

$$\mathrm{Obj}^{(t)}=\sum_{j=1}^{T}\left[G_j\omega_j+\frac{1}{2}(H_j+\lambda)\omega_j^2\right]+\gamma T \quad (14)$$

接着对ω_j求一阶导数，并使之为0，得叶子节点j对应的权值和最优目标函数为：

$$\omega_j^*=-\frac{G_j}{H_j+\lambda} \quad (15)$$

$$\mathrm{Obj}=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda}+\gamma T \quad (16)$$

2.4.3　基于网格搜索的XGBoost模型超参数调整

XGBoost模型的超参数分2类：第1类负责控制模型的复杂度，第2类用于增加随机性，从而使得模型在训练时对噪声不敏感。下面介绍调参重点关注的超参数：

a）eta，学习率，默认为0.3，范围为[0，1]。

b）gamma，最小划分损失，它是对于一个叶子节点，当对它采取划分之后，损失函数的降低值的阈值，默认为0。

c）max_depth，每棵子树的最大深度。其取值范围为[0，∞]，0表示没有限制，默认值为6。该值越大，则子树越复杂；值越小，则子树越简单。

d）min_child_weight，子节点的权重阈值。表示对于一个叶子节点，当对它进行划分之后，它的所有子节点的权重之和的阈值。该值越大，则算法越保守。默认值为1。

e）subsample，对训练样本的采样比例。取值范围为(0，1]，默认值为1。

f）colsample_bytree，构建子树时，对特征的采样比例。取值范围为(0，1]，默认值为1。

g）lambda，正则化系数（基于weights的正则化），默认为1。该值越大则模型越简单。

h）alpha，正则化系数（基于weights的正则化），默认为0。该值越大则模型越简单。

本文利用scikit-learn库自带的GridSearchCV网格搜索算法来调整XGBoost算法超参数（见图8），候选超

```
732 grid_search=GridsearchCV(xgb_model,param_grid,cv=5,vebose=2,n-jobs=-1,scoring='neg_mean_absolute_error')
733 grid_search.fit(X_model_evaluate,targets_2)
734 print('测试集最佳得分:%f' % grid_search.best_score_)

432 {'alpha':0.85,'colsample_bytree':0.7,'eta':0.05,'gamma':0,:'lambda':5,'max_depgh':18,'min_child_weight':1,
    'n_estimators':200,'subsample':1}
433 XGBRegressor(alpha=0.85,base_score=0.5,booster=None,colsample_bylevel=1,)
434               colsample_bynode=1,colsample_bytree=0.7,eta=0.05,gamma=0,
435               gpu_id=-1,importance_type='gain',interaction_constraints=None,
436               lambda=5,learning_rate=0.050 000 000 7,max_delta_step=0,
437               max_depth=18,min_child_weight=1,missing=nan,
438               monotone_constraints=None,n_estimators=200,n_jobs=0,
439               num_parallel_tree=1,objective='reg:squarederror',random_state=60
440               reg_alpha=0.850 000 024,reg_lambda=5,scale_pos_weight=1,
441               subsample=1,tree_method=None,validate_parameters=False,
442               verbosity=None)
```

图8　基于GridSearchCV的XGBoost模型超参数调整

参数值集合如下：

param_grid = {'eta':[0.01,0.015,0.025,0.05,0.1],
'n_estimators':[100,150,200,250,300],
'gamma':[0,0.05,0.1,0.15,0.2,0.3,0.35],
'max_depth':[16,17,18,19,20,21,22,23,24],
'min_child_weight':[1,2,3,4,5,6,7],
'subsample':[0.5,0.6,0.7,0.8,0.9,1],
'colsample_bytree':[0.5,0.55,0.6,0.65,0.7,0.75],
'lambda':[1.3,1.4,1.5,1,3,5],
'alpha':[0.35,0.45,0.55,0.65,0.75,0.85]}

最终得到的最佳超参数组合是：{'alpha': 0.85, 'colsample_bytree': 0.7, 'eta': 0.05, 'gamma': 0, 'lambda': 5, 'max_depth': 18, 'min_child_weight': 1, 'n_estimators': 200, 'subsample': 1}。在测试集上进行评估，平均绝对误差MAE为0.005 10。

2.4.4　基于ANR网络切换占比模型的传统网络小区邻区关系预测

对需优化邻区关系的传统网络小区选取5 km范围内的周边小区，根据切换占比模型特征采集数据，构成样本输入模型进行预测。实验结果表明，对现网真实邻区关系的命中率为60%，即60%的现网邻区出现在预测出的占比前50名小区中。

3　总结

传统网络小区邻区优化是网优工作的重点和难点，人工优化方法费时费力。通过引入机器学习算法学习ANR网络的邻区关系建立切换次数占比模型可模拟真实的传统现网邻区关系情况，极大程度地提高了邻区优化效率和用户网络口碑。

参考文献：

[1] 工业和信息化部. 促进新一代人工智能产业发展三年行动计划(2018-2020)[EB/OL].[2020-10-21]. http://www.cac.gov.cn/2017-12/26/c_1122166495.htm.
[2] 周晓鹏，李程峻. WCDMA同频邻区漏配多配的测试方法分析[J/OL].[2020-10-21]. http://d.wanfangdata.com.cn/conference/7320426.
[3] 聂嘉文，宋昭辉，王栋，等. 基于拓扑结构进行邻区规划优化的方法:CN201610245047.3[P]. 2016-07-27.
[4] czhhh. XGBoost算法原理[EB/OL].[2020-10-21]. https://zhuanlan.zhihu.com/p/88966999.
[5] 阿泽. 终于有人把XGBoost和LightGBM讲明白了，项目中最主流的集成算法![EB/OL].[2020-10-21]. https://mp.weixin.qq.com/s/LoX987dypDg8jbeTJMpEPQ.
[6] 王硕然，林华乐，陈爽. 基于MR大数据的邻区优化算法应用[J]. 中国新通信，2016，18(020):95-96.
[7] 陈奇. 移动通信邻区优化系统的设计与实现[D]. 大连:大连理工大学，2014.
[8] 赵春阳，赵春雷，王蔚. 一种基于MRE数据的LTE邻区优化方法研究[J]. 移动通信，2016，40(9):60-64.
[9] 郑亚平，贾磊，方路成，等. 基于AI的突发人流聚集区域识别与预警方法[J]. 电信工程技术与标准化，2020(9).
[10] 王明君. TD-LTE高铁专网优化方法研究[J]. 移动通信，2014(19):67-71.
[11] 张功国. 基于X2接口TD-LTE系统邻区自优化流程设计[J]. 数字技术与应用，2013(3):174-175.
[12] 王新楼，郭超，纪国强. 基于MR位置指纹定位的优化算法分析和实践[J]. 邮电设计技术，2014(11):52-56.

作者简介：
李张铮，毕业于大连理工大学，工程师，主要从事无线网络优化工作；陈锋，毕业于福州农林大学，高级工程师，主要从事无线网络优化工作；董帝烺，毕业于厦门大学，高级工程师，主要从事WCDMA、LTE的网络优化和NR新技术的研究工作。