中国研究生创新实践系列大赛
“华为杯”第十六届中国研究生数学建模竞赛

学　　校　中国科学技术大学/软件学院

参赛队号　**No.19103580023**

队员姓名	**1. 高泽洲**
	2. 朱俊
	3. 刘剑

中国研究生创新实践系列大赛
“华为杯”第十六届中国研究生数学建模竞赛

题　　目　无线智能传播模型

摘　　要:

针对无线传播模型并准确预测出新环境下平均信号接收功率的问题，本文通过 stacking 的方式融合三大模型，建立起一套评价小区平均信号接收功率的体系。该模型的创新之处在于，通过使用树模型、传统信道模型和深度学习模型的 stacking，合理的利用了传统信道模型的知识、树模型对特征空间的合理划分、以及深度学习模型的学习能力，从而对 RSRP 进行合理准确的预测。此模型首先以传统信道模型为基础，通过修正 hata 模型中的系数，建立 cost231 hata 模型；其次，通过梯度提升树算法求解小区平均信号接收功率与基站属性、小区属性之间的关系；最后对小区特征进行含有修正因子的归一化，通过神经网络算法进一步拟合小区平均信号接收功率的函数。

问题一中，我们总共使用了四类特征：部分原始特征、几何特征、信道特征以及模型输出特征。原始特征是去除标签的数据的输入，几何特征中我们考虑了基站与小区之间的距离、栅格与信号线的相对高度、基站信号发射下倾角、基站信号发射下倾角的正切值；信号模型特征我们计算了基站天线有效高度、用户天线有效高度、对数化的基站发射频率、场景纠正系数、用户天线高度纠正项、用户天线有效高度与链路距离的乘积; 模型输出特征是指传统信道模型和树模型的输出值。

问题二中，基于问题一中构造的特征，我们计算这些特征与目标之间的相关性以及判断特征是否发散，对特征进行筛选。

问题三中，通过原始特征、几何特征和信道特征计算 cost-231 模型，然后通过几何特征、信道特征、cost231 模型输出拟合梯度提升树，最后通过几何特征、信道特征、cost231 模型输出、梯度提升树输出拟合神经网络模型。图 1 是我们解决方案的工作流：

通过 stacking 方式融合 cost231-hata 模型、梯度提升树和神经网络模型，取得了超过任意单一模型的表现，在华为 ModelArts 平台上部署发布我们的模型，在测试集上计算得到的 rmse 等于 9.670。

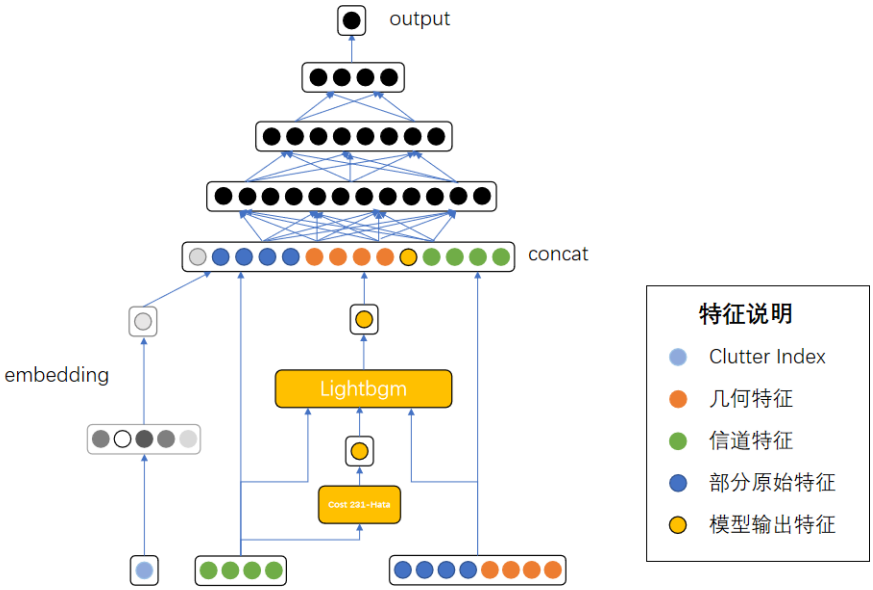

图 **1**　模型结构图

关键字： RSRP　传统信道模型　树模型　神经网络　stacking

目录

1. 问题重述

随着 5G NR 技术的发展，5G 在全球范围内的应用也在不断地扩大。运营商在部署 5G 网络的过程中，需要合理地选择覆盖区域内的基站站址，进而通过部署基站来满足用户的通信需求。在整个无线网络规划流程中，高效的网络估算对于精确的 5G 网络部署有着非常重要的意义。

现有的无线传播模型可以按照研究方法进行区分，一般分为：经验模型、理论模型和改进型经验模型。在实际传播模型建模中，为了获得符合目标地区实际环境的传播模型，需要收集大量额外的实测数据、工程参数以及电子地图用来对传播模型进行校正。此外无线 LTE 网络已在全球普及，全球几十亿用户，每时每刻都会产生大量数据。如何合理地运用这些数据来辅助无线网络建设就成为了一个重要的课题。

与传统经验模型需要额外人力物力进行校正相比，是否可以利用采集的历史数据并利用机器学习技术，得到一套合适的机器学习模型用以对不同场景下信道传播路径损耗进行准确预测，成为一个非常有价值的研究方向。

所以本题的目的是利用采集的历史数据并利用机器学习技术，得到一套合适的机器学习模型用以对不同场景下信道传播路径损耗进行准确预测。即通过寻找工程参数、地理环境等因素与平均信号接收功率（RSRP）之间的映射模型（理论与实践表明 RSRP 是工程参数、地理环境等因素的随机函数），从而能够在新的环境中快速预测特定地理位置的 RSRP 值。

2. 模型假设

在问题的求解过程中，考虑到实际情况的需要，采用了以下的相关假设：

(1) 假设栅格左上角的坐标表示栅格的坐标;

(2) 假设相邻小区在 RSRP 测量时不会相互干扰，同一个位置对两个小区进行测量时，两个小区的 RSRP 值不会叠加；

(3) 假设基站发射高度值 Height 是基站发射机到地面的高度，不考虑 Cell Building Height; (4) 假设用于评分的数据集与训练数据集来自统一数据集的随机划分。

3. 符号说明

符号	意义
θ_D	垂直总倾度
θ_{MD}	垂直电下倾角
θ_{ED}	垂直机械下倾角
θ_P	栅格与发射塔水平夹角
θ_A	发射线水平夹角
Sim	θ_P 与 θ_A 角度差的 cos 值
X_{CELL}	小区所属站点的栅格位置，X 坐标
Y_{CELL}	小区所属站点的栅格位置，Y 坐标
X	栅格位置，X 坐标
Y	栅格位置，Y 坐标
hr	用户天线有效高度
hb	基站天线有效高度
h_b	发射机高度
Δh	栅格与信号线相对高度
f	发射机中心频率
α_{urban}	城市用户天线高度纠正项
$\alpha_{suburban}$	非城市用户天线高度纠正项
C_m	场景纠正常数
PL	传播路径损耗
RSRP	栅格的平均信号接收功率
$pred_{lgb}$	lgb 模型的输出

表 1　符号说明表

4. 问题的分析

4.1 问题一特征工程

4.1.1 问题描述和分析

赛题提供的训练数据集包含多个小区的工程参数数据、地图数据和 RSRP 标签数据, 数据集各种参数说明如下：

字段名称	含义	单位
Cell Index	小区唯一标识	-
Cell X	小区所属站点的栅格位置，X 坐标	-
Cell Y	小区所属站点的栅格位置，Y 坐标	-
Height	小区发射机相对地面的高度	m
Azimuth	小区发射机水平方向角	Deg
Electrical Downtilt	小区发射机垂直电下倾角	Deg
Mechanical Downtilt	小区发射机垂直机械下倾角	Deg
Frequency Band	小区发射机中心频率	MHz
RS Power	小区发射机发射功率	dBm

表 2　工程参数数据

字段名称	含义	单位
Cell Index	小区唯一标识	-
Cell X	小区所属站点的栅格位置，X 坐标	-
Cell Y	小区所属站点的栅格位置，Y 坐标	-
Height	小区发射机相对地面的高度	m
Azimuth	小区发射机水平方向角	Deg
Electrical Downtilt	小区发射机垂直电下倾角	Deg
Mechanical Downtilt	小区发射机垂直机械下倾角	Deg
Frequency Band	小区发射机中心频率	MHz
RS Power	小区发射机发射功率	dBm

表 3　地图数据

字段名称	含义	单位
RSRP	栅格的平均信号接收功率	dBm

表 4　RSRP 标签数据

高效的机器学习模型建立依赖于输入变量与问题目标的强相关性，因此输入变量也称为“特征”。特征工程的本质是从原始数据中转换得到能够最好表征目标问题的参数，并使得各个参数的动态范围在一个相对稳定的范围内，从而提高机器学习模型训练的效率。一般特征工程的典型技术有：

- 剔除失真、低质量数据；数据插值补齐；去除异常点
- 连续数据离散化；数据去均值；幅度限制；方差限制

在本次信道传播模型问题，我们需要充分利用几何位置特征以及传统经验信道模型中涉及的参数。

4.1.2 问题求解

(1) 原数据集的特征参数

对于原始数据集的特征参数，首先，我们去除了基站特征：对于同一个小区的用户点，这些特征都是相同的，方差为 0，对 target 不会产生影响。最终我们维持以下参数不变作为我们的特征参数：

- 栅格位置 X 坐标，X
- 栅格位置 Y 坐标，Y
- 栅格上的建筑物高度 Building Height
- 栅格上的海拔高度 Altitude

(2) 几何特征

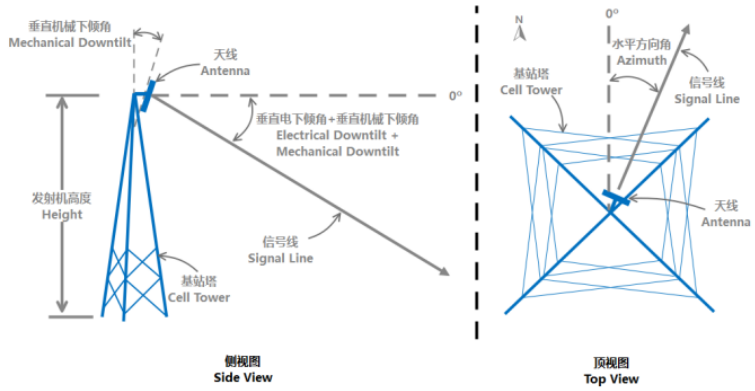

图 **2**　几何特征图

如图所示，几何特征中，链路距离、栅格与信号线相对高度、水平角与两地之间夹角的相似度很明显跟接受的强度有关系，所以我们在几何特征中增加以下特征参数:

- 增加总倾角即垂直电下倾角与垂直机械倾角的和 θ

$$\theta_D = \theta_{MD} + \theta_{ED} \tag{1}$$

- 增加链路距离 d

$$d = \sqrt{(X_{CELL} - X)^2 + (Y_{CELL} - Y)^2} \tag{2}$$

- 增加总倾角的正切值 t_θ

$$t\_\theta = tan(\theta_D) \tag{3}$$

- 增加栅格与信号线相对高度 Δh

$$\Delta h = \ h_b - tan(\theta_D)\ h_b \tag{4}$$

其中:

$$h_b = Height + Cell\ Altitude - Altitude \tag{5}$$

- 增加栅格与信号塔水平倾角 θ_P

$$\theta_P = \text{arctan}(|Y_{CELL} - Y| \,/\, |X_{CELL} - X|) \tag{6}$$

- 增加栅格与信号塔水平倾角 θ_P 与信号线水平倾角 θ_A 角度差的 cos 值

$$sim = \cos|\theta_P - \theta_A| \tag{7}$$

(3) 信道特征

同时我们也构造了若干信道特征参数。其定义如下：

- 增加参数 $\log f$
- 增加参数 $\log h_b$
- 增加参数 $\log d$
- 增加用户天线高度纠正项 α

$$\alpha = \begin{cases} 8.29(\log_{10}(1.54h_R))^2 - 1.1 & ,\text{if } 150 \le f \le 200 \\ 3.2\,(\log_{10}(11.75h_R))^2 - 4.97 & ,\text{if } 200 < f \le 1500 \end{cases}$$

其中:

$$h_r = Building\ Height + Altitude \tag{8}$$

- 增加场景纠正常数 C_m

1) 当 Clutter Index ∈ { 10,11,12,13,16,20 } 时，即小区处于树木密度适中的中等城市和郊区的中心时, 取 C_m=3dB;

2) 当 Clutter Index ∉ { 10,11,12,13,16,20 } 时，即小区处于大城市中心时，取 C_m=0dB.

(4) 模型特征

首先，我们将传统经验信道模型 Cost231-Hata 模型得到的输出结果 PL 纳入特征参数。其定义如下：

$$PL = 46.3 + 33.9lgf - 13.82lg(h_b) - \alpha + (44.9 - 6.55lg(h_b))lgd + C_m \tag{9}$$

其中 PL 定义为传播路径损耗（dB）、为载波频率（MHz）、基站天线有效高度（m）、用户天线有效高度（m）、用户天线高度纠正项 (dB)、链路距离（km）以及为场景纠正常数（dB）

其次，我们将 lgb 模型的输出作为神经网络的输入特征参数，为了下一步中使用树模型、传统信道模型和深度学习模型的 stacking。

4.2 问题二特征相关性

4.2.1 问题描述和分析

完成特征设计后，通常需要选择有意义的特征输入机器学习模型进行训练。对于不同方法构造出来的特征，需要从多个层面来判断这个特征是否合适。通常来说，可以从以下两个方面来选择特征：

- 特征是否发散：如果一个特征不发散，例如方差接近于 0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
- 特征与目标的相关性：这点比较显见，与目标相关性高的特征，应当优先选择

基于提供的各小区数据集，设计多个合适的特征，计算这些特征与目标的相关性，并将结果量化、排序，形成表格。

4.2.2 问题求解

我们通过皮尔森相关系数的绝对值来反映特征和 RSRP 线性相关的程度。皮尔森相关系数求解公式如下：

$$r=\frac{\sum_{i=1}^{n}(X_i-\overline{X})(Y_i-\overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\overline{Y})^2}}$$

图 **3**

其中 n 为样本量，X,Y,$\bar{X}$,$\bar{Y}$ 分别为两个变量的观测值和均值。r 描述的是两个变量间线性相关强弱的程度。r 的取值在-1 与 +1 之间，若 r>0，表明两个变量是正相关，即一个变量的值越大，另一个变量的值也会越大；若 r<0，表明两个变量是负相关，即一个变量的值越大另一个变量的值反而会越小。r 的绝对值越大表明相关性越强。

绘制各特征与 RSRP 皮尔森相关系数绝对值的柱状图如下：

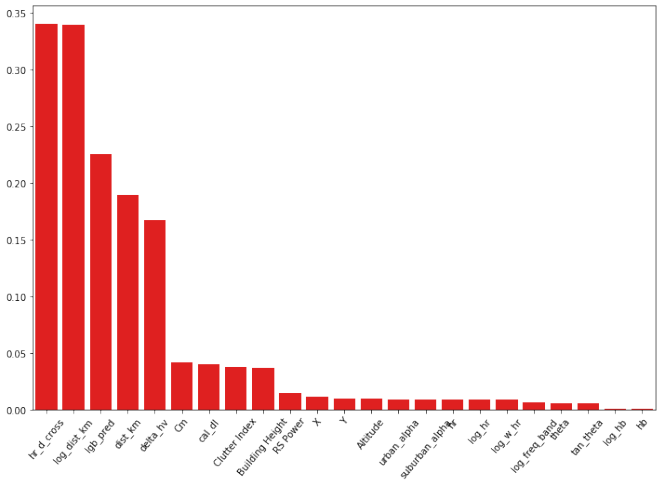

图 **4**　各特征与 **RSPR** 皮尔森系数

4.2.3 特征与目标相关性表格

排序	特征名称	该特征与目标的相关性
0	hr · d	0.339872
1	$logd$	0.339790
2	$pred_{lgb}$	0.225452
3	d	0.189281
4	Δh	0.167327
5	Cm	0.041435
6	Clutter Index	0.037404
7	Building Height	0.036822
8	RS Power	0.014241
9	X	0.011341
10	Y	0.009749
11	Altitude	0.009698
12	α_{urban}	0.008791
13	$\alpha_{suburban}$	0.008789
14	hr	0.008714
15	$\log hr$	0.008572
16	$\log whr$	0.008401
17	$\log f$	0.006383
18	θ	0.005583
19	$\tan(\theta)$	0.005546
20	$\log h_b$	0.000672
21	h_b	0.000482

4.3 问题三 RSRP 预测

4.3.1 问题描述和分析

在设计和选择了有效的特征之后，就可以通过建立预测模型来进行 RSRP 的预测了。队根据自己建立的特征集以及赛题提供的训练数据集，建立基于 AI 的无线传播模型来对不同地理位置的 RSRP 进行预测。

4.3.2 模型建立与求解

问题三中，通过原始特征、几何特征和信道特征计算 cost-231 模型，然后通过几何特征、信道特征、cost231 模型输出拟合梯度提升树，最后通过几何特征、信道特征、cost231 模型输出、梯度提升树输出拟合神经网络模型。以下是我们解决方案的工作流：

在得到 cost231 模型特征后，我们拟合了 500 棵梯度提升树，每棵树的学习

率是 0.3，最大叶子结点数是 64，不限制树深，为了防止模型过拟合，在学习每棵树时，采样 80 % 的样本建树，每个样本采样 80 % 的特征进行划分，同时设置 L1 与 L2 的系数为 1，有效地防止了过拟合。

在获得了 Clutter Index 特征、部分原始特征、信道特征、几何特征和树模型特征后，我们首先考虑如何处理 Clutter Index 特征。在深度学习模型中，网络并不能对用数值表示的类别型特征进行有效的学习。对于维数较高的类别型特征，一般的处理方式是对其进行 embedding。在这里我们将 Clutter Index 特征中的每一个值 embedding 为一个 1 维的向量，并将其与部分原始特征、信道特征、几何特征和树模型特征拼接在一起，作为后续神经网络的输入。在此问题中，我们需要对 RSRP 进行预测，而 RSRP 的定义域可看做是整个实数域，我们把此问题定性为回归问题，我们希望深度学习模型可以学习到一个能准确解决回归问题的模型。

神经网络的设计采用三个全连接的隐藏层的设计。其中第一层的隐藏元个数为 64 个，其目的是希望低阶特征能在尽量减少计算资源消耗的情况下尽可能多的交叉。第二层的隐藏元个数为 32 个，第三层的隐藏元个数为 8 个，其目的是希望得到高阶的特征交叉项。考虑到避免梯度消失的问题，以及尽可能减少计算资源的消耗，每一个隐藏层的激活函数我们均采用 Relu 函数。同时为了避免模型的过拟合问题，每一个隐藏层我们均采用系数为 0.3 的 L2 正则化。最后一个隐藏层输出的 8 维向量进行线性组合得到最终的对 RSRP 的预测值。我们的损失函数选择为 MSE，即均方误差损失函数。在训练的过程中，我们采用 8：2 的比例划分了训练集和验证集，优化方式为自适应学习率的方法 Adam。通过 stacking 方式融合 cost231-hata 模型、提督提升树和神经网络模型，取得了超过任意单一模型的表现. 其结构图如下：

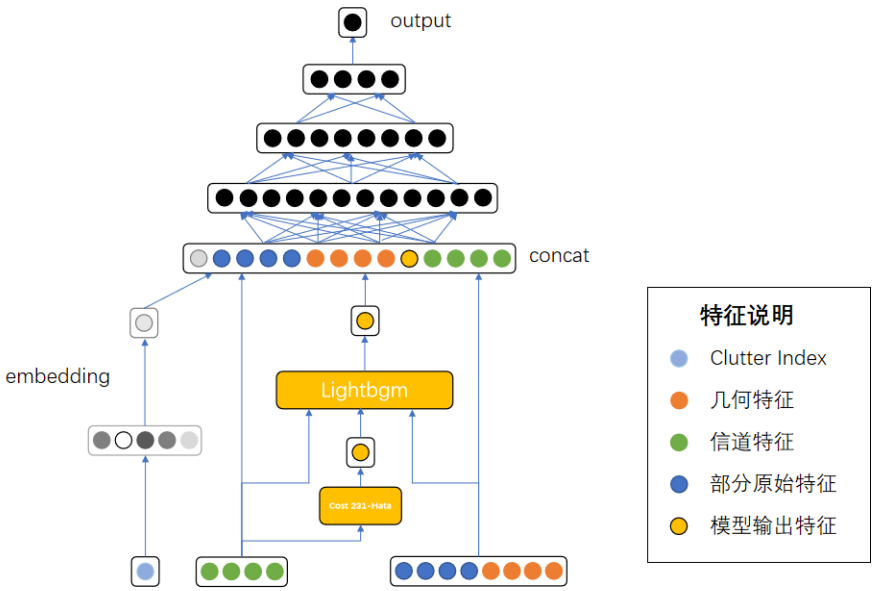

图 **5**　模型结构图

最终在华为 modelarts 平台上部署发布我们的模型，在测试集上计算得到的 rmse 等于 9.670。

5. 模型的评价

该模型使用了 stacking 方式融合 cost231-hata 模型、梯度提升树和神经网络模型，取得了超过任意单一模型的表现。特征中例如坐标、Cluster Index 等特征，深度学习模型并不能很好的利用。而在梯度提升树模型中，建树的过程实则是在对特征空间进行划分，因此梯度提升树模型的训练可以很好运用坐标、Cluster Index 等特征，在一定程度上弥补了深度学习模型的不足。而使用传统信道模型的过程则是弥补了深度学习模型不能很好的由简单的低阶特征拟合出更为有效的高阶特征的不足。在实验中，通过经验的方式由传统信道模型得出的特征在模型中的重要度也很高，与 RSRP 的相关度也很高，因此在模型中加入传统信道模型因素也是直观且合理的选择。深度学习模型是目前非常热门的机器学习算法，神经网络理论上可以对任意函数进行有效的拟合。我们综合了以上三点的考虑，构建了我们的 stacking 模型，取得了超过任意单一模型的表现。

6. 参考文献

[1] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. arXiv preprint arXiv:1207.0580, 2012.

[2] Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.

[3] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[C]//Advances in Neural Information Processing Systems. 2017: 3146-3154.

[4] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

[5] Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations[C]//Proceedings of the 10th ACM conference on recommender systems. ACM, 2016: 191-198.

[6] Guo H, Tang R, Ye Y, et al. DeepFM: a factorization-machine based neural network for CTR prediction[J]. arXiv preprint arXiv:1703.04247, 2017.

[7] Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st workshop on deep learning for recommender systems. ACM, 2016: 7-10.

[8] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[9] Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.

[10] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.

[11] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[12] Baltazar Espinoza,Ricardo A. Sáenz. Restrictions of harmonic functions and Dirichlet eigenfunctions of the Hata set to the interval[J]. Analysis,2016,36(3).

[13] Castro, B. S. L., Gomes, I. R., Ribeiro, F. C. J., Cavalcante, G. P. S.. COST231-Hata and SUI Models performance using a LMS tuning algorithm on 5.8GHz in Amazon Region cities[P]. Antennas and Propagation (EuCAP), 2010 Proceedings of the Fourth European Conference on,2010.

[14] Hinton G E. A practical guide to training restricted Boltzmann machines[M]//Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012: 599-619.

[15] Nuno Cota,António Serrador,Pedro Vieira,Ana Rita Beire,António Rodrigues. On the Use of Okumura‐Hata Propagation Model on Railway Communications[J]. Wireless Personal Communications,2017,93(3).

附录 A 程序代码

(1) 问题一：特征工程

```
import numpy as np
import pandas as pd
import lightgbm as lgb
import math
from sklearn.model_selection import train_test_split

path = 'train_total.csv'
train_data = pd.read_csv(path, index_col=0)
train_data = train_data.reset_index(drop=True)
assert len(set(train_data.index)) == len(train_data.index)

#distance
train_data['distance'] = (train_data['Cell X'] -
    train_data['X'])**2 + (train_data['Cell Y'] -
    train_data['Y'])**2
train_data['distance'] = train_data['distance'].apply(lambda x:
    5*math.sqrt(x))
print('distance finished.')

#delta_hv
train_data['theta'] = (train_data['Electrical Downtilt'] +
    train_data['Mechanical Downtilt']) / 180
train_data['tan_theta'] = train_data['theta'].apply(lambda x:
    math.tan(x))
train_data['delta_hv'] = train_data['Height'] -
    train_data['tan_theta'] * train_data['distance']
print('delta_hv finished.')

#log_freq
train_data['log_freq_band'] = train_data['Frequency
    Band'].apply(lambda x: math.log(x))
print('log_freq_band finished.')

# hb: 基站天线有效高度
train_data['hb'] = train_data['Height'] + train_data['Cell
    Altitude'] - train_data['Altitude']
train_data['log_hb'] = train_data['hb'].apply(lambda x:
    math.log(abs(x)+0.083))
print('log_hb finished.')

# hr: 用户天线有效高度
train_data['hr'] = train_data['Building Height'] +
    train_data['Altitude']
train_data['log_hr'] = train_data['hr'].apply(lambda x:
    math.log(abs(x)+0.083))
print('log_hr finished.')

# dist: km
train_data['dist_km'] = train_data['distance'] / 1000
train_data['log_dist_km'] = train_data['dist_km'].apply(lambda
```

```
    x: math.log(abs(x)+0.083))
print('log_dist finished.')

# dl and hr_d_cross
train_data['dl'] = train_data['RS Power'] - train_data['RSRP']
train_data['hr_d_cross'] = train_data['log_hr'] *
    train_data['log_dist_km']
print('dl, hr_d_cross finished.')

print(train_data.columns)
def get_cm(x):
   target = {10: 1, 11: 1, 12: 1, 13: 1, 16: 1, 20: 1}
   if x['Cell Clutter Index'] in target or x['Clutter Index'] in
      target:
     return 3
   else:
     return 0

train_data['Cm'] = train_data.apply(get_cm, axis=1)
print('Cm finished.')
#train_data = train_data.rename(columns={'hue': 'hr', 'log_hue':
    'log_hr'})
train_data['suburban_alpha'] = (1.1*train_data['log_freq_band']
    - 0.7)*train_data['hr'] -
    (1.56*train_data['log_freq_band']-0.8)
train_data['log_w_hr'] = train_data['hr'].apply(lambda x:
    math.log(11.75*x+0.083))
train_data['urban_alpha'] =
    3.2*train_data['log_w_hr']*train_data['log_w_hr'] - 4.97

def get_dl(x):
   res = 0
   if x['Cm'] == 0: # 偏远地区
     res = 46.3 + 33.9*x['log_freq_band'] - 13.82*x['log_hb'] +
        x['log_dist_km']*(44.9-6.55*x['log_freq_band']) + x['Cm']
        - x['suburban_alpha']
   else: # 大城市区
     res = 46.3 + 33.9*x['log_freq_band'] - 13.82*x['log_hb'] +
        x['log_dist_km']*(44.9-6.55*x['log_freq_band']) + x['Cm']
        - x['urban_alpha']
   return res

train_data['cal_dl'] = train_data.apply(get_dl, axis=1)
print('cal_dl finished.')

# h_theta
def get_h_theta(x):
   res = 0
   if x['X'] - x['Cell X'] > 0 and x['Y'] - x['Cell Y'] > 0:
     res = math.atan(abs(x['X'] - x['Cell X']) / abs(x['Y'] -
        x['Cell Y']))
   elif x['X'] - x['Cell X'] < 0 and x['Y'] - x['Cell Y'] > 0:
     res = -math.atan(abs(x['X'] - x['Cell X']) / abs(x['Y'] -
        x['Cell Y']))
```

```
  elif x['X'] - x['Cell X'] < 0 and x['Y'] - x['Cell Y'] < 0:
    res = - math.atan(abs(x['Y'] - x['Cell Y']) / abs(x['X'] -
       x['Cell X'])) - 90
  elif x['X'] - x['Cell X'] > 0 and x['Y'] - x['Cell Y'] < 0:
    res = math.atan(abs(x['Y'] - x['Cell Y']) / abs(x['X'] -
       x['Cell X'])) + 90
  elif x['X'] == x['Cell X'] and x['Y'] > x['Cell Y']:
    res = 0
  elif x['X'] == x['Cell X'] and x['Y'] < x['Cell Y']:
    res = 180
  elif x['Y'] == x['Cell Y'] and x['X'] > x['Cell X']:
    res = 90
  elif x['Y'] == x['Cell Y'] and x['X'] < x['Cell X']:
    res = -90
  else:
    res = x['Azimuth']
  return res

train_data['h_theta'] = train_data.apply(get_h_theta, axis=1)
train_data['cos_h_theta'] = ((train_data['h_theta'] -
   train_data['Azimuth'])*math.pi/180).apply(math.cos)
print('h_theta and cos finished.')

# del wrong rows
del_1 = train_data[(train_data['Clutter Index'] == 10) &
   (train_data['Building Height'] <= 60)].index
del_2 = train_data[(train_data['Clutter Index'] == 14) &
   (train_data['Building Height'] >=20)].index
train_data.drop(del_1, inplace=True)
train_data.drop(del_2, inplace=True)

def get_flag(x):
cluster_maxheight_map = {1:0,2:0,3:0,4:10,5:10,6: 15,7:
   15,8:15,9:15, 10:200,11:60,12: 40,13:20,14:20,15:
   10,16:10,17:20,18:20,19:20,20:20}
  if x['Building Height'] <= cluster_maxheight_map[x['Clutter
     Index']]:
    return True
  else:
    return False
train_data['flag'] = train_data.apply(get_flag, axis=1)
train_data = train_data[train_data['flag']==True]
print('del wrong rows.')
```

(2) 问题二：求解特征相关性

```
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
import os
import lightgbm as lgb
import json
import math
```

```
# read csv
path = 'train_data_featured.csv'
train_data = pd.read_csv(path, index_col=0)
train_data = train_data.reset_index(drop=True)
assert len(set(train_data.index)) == len(train_data.index)

# create cluster_height_map
cluster_height_map = {}
def get_cluster_height_map(x):
  global cluster_height_map
  if x['Clutter Index'] in cluster_height_map:
    if x['Building Height'] in cluster_height_map[x['Clutter
       Index']]:
    pass
    else:
    cluster_height_map[x['Clutter Index']].append(x['Building
       Height'])
  else:
   cluster_height_map[x['Clutter Index']] = [x['Building
      Height']]
train_data.apply(get_cluster_height_map, axis=1)
print('create cluster_height_map finished.')

# del wrong rows
del_1 = train_data[(train_data['Clutter Index'] == 10) &
   (train_data['Building Height'] <= 60)].index
del_2 = train_data[(train_data['Clutter Index'] == 14) &
   (train_data['Building Height'] >=20)].index
train_data.drop(del_1, inplace=True)
train_data.drop(del_2, inplace=True)
def get_flag(x):
  cluster_maxheight_map = {
  1:0,
  2:0,
  3:0,
  4:10,
  5:10,
  6: 15,
  7: 15,
  8:15,
  9:15,
  10:200,
  11:60,
  12: 40,
  13:20,
  14:20,
  15: 10,
  16:10,
  17:20,
  18:20,
  19:20,
  20:20
  }
  if x['Building Height'] <= cluster_maxheight_map[x['Clutter
```

```
      Index']]:
     return True
   else:
    return False
train_data['flag'] = train_data.apply(get_flag, axis=1)
train_data = train_data[train_data['flag']==True]
print('del wrong rows.')
print(train_data.shape)

# dnn
data = train_data
print(data.shape)
model = lgb.Booster(model_file='./model/lgb01.txt')
# 输入lgb的特征
_features = ['X', 'Y', 'Altitude', 'Building Height',
'Clutter Index', 'distance', 'theta', 'delta_hv',
'hb', 'log_hb', 'hr', 'log_hr', 'dist_km',
'log_dist_km', 'hr_d_cross', 'suburban_alpha','cal_dl']
x= data[_features]
y_pred = model.predict(x, num_iteration=model.best_iteration)
data['lgb_pred'] = y_pred
print('lgb pred calculated.')
corr = data.corr()
#print(corr)
corr.to_csv('features_corr.csv')
```

(3) 问题三：模型训练

```
features_all = ['Clutter Index', 'theta', 'tan_theta',
   'delta_hv',
'log_freq_band', 'hb', 'log_hb', 'hr', 'log_hr', 'dist_km',
'log_dist_km', 'hr_d_cross', 'Cm', 'suburban_alpha', 'log_w_hr',
'urban_alpha', 'cal_dl','RS Power','lgb_pred','X', 'Y',
   'Altitude', 'Building Height',
'h_theta', 'cos_h_theta']
features = ['X', 'Y', 'Altitude', 'Building Height', 'delta_hv',
   'hb', 'log_hb', 'hr', 'log_hr', 'dist_km',
'log_dist_km', 'hr_d_cross', 'log_w_hr']
features_dnn = ['X', 'Y', 'Altitude', 'Building Height',
'theta', 'tan_theta', 'delta_hv',
'log_freq_band', 'hb', 'log_hb', 'hr', 'log_hr', 'dist_km',
'log_dist_km', 'hr_d_cross', 'Cm', 'suburban_alpha', 'log_w_hr',
'urban_alpha', 'cal_dl','RS Power','lgb_pred', 'h_theta',
   'cos_h_theta']

target = ['RSRP']
cate_fea = ['Clutter Index']
input = data[features_all]
# bn
for x in features:
  if input[x].max() - input[x].min() == 0:
     div = input[x].max() + 1
  else:
    div = input[x].max() - input[x].min()
```

```
  input[x] = (input[x] - input[x].min()) / div
print('bn finish.')
label = data[target]

batch_size = 1024
x_train, x_test, y_train, y_test = train_test_split(input,
    label, test_size=0.2, random_state=666)

x = tf.placeholder(tf.float64, [None, 24])
cate_x = tf.placeholder(tf.float64, [None, 1])
y = tf.placeholder(tf.float64, [None, 1])


clutter = tf.cast(tf.reshape(cate_x, [-1]), tf.int64)
embed = tf.Variable(tf.random_normal(shape=[20], mean=0,
    stddev=1, dtype=tf.float64), name='clu_embed')
# embed = tf.Print(embed, [embed], "embed: ", summarize=100)
_clutter = tf.reshape(tf.nn.embedding_lookup(embed, clutter),
    [-1, 1])
new_x = tf.concat([x, _clutter], axis=1)
# layers1 = tf.layers.dense(new_x, 64, activation=tf.nn.relu)
# layers2 = tf.layers.dense(layers1,32, activation=tf.nn.relu)
# layers3 = tf.layers.dense(layers2,8, activation=tf.nn.relu)
layers1 = tf.layers.dense(new_x, 64, activation=tf.nn.relu,
    kernel_regularizer=tf.contrib.layers.l2_regularizer(0.3))
layers2 = tf.layers.dense(layers1,32, activation=tf.nn.relu,
    kernel_regularizer=tf.contrib.layers.l2_regularizer(0.3))
layers3 = tf.layers.dense(layers2,8, activation=tf.nn.relu,
    kernel_regularizer=tf.contrib.layers.l2_regularizer(0.3))
output = tf.layers.dense(layers3, 1, activation=None)

loss = tf.reduce_mean(tf.square(output - y))
optimize = tf.train.AdamOptimizer(0.001).minimize(loss)

saver = tf.train.Saver()
# gpu_options = tf.GPUOptions(allow_growth=True)
# with
    tf.Session(config=tf.ConfigProto(gpu_options=gpu_options)) as
    sess:
with tf.Session() as sess:
  init = tf.global_variables_initializer()
  sess.run(init)
  lowest_mse = [-1, float('inf')]
  for i in range(1000):
     for j in range(1 + len(x_train) // batch_size):
        tmpx = x_train[j*batch_size: (j+1)*batch_size]
        fea = tmpx[features_dnn]
        cat = tmpx[cate_fea]
        tmpy = y_train[j*batch_size: (j+1)*batch_size]
        # print(fea.shape)
        # print(cat)
        sess.run(optimize, feed_dict={x: fea, y: tmpy, cate_x:
           cat})
     x_test_fea = input[features_dnn]
```

```
    x_test_cat = input[cate_fea]
    _loss = sess.run(loss, feed_dict={x: x_test_fea, y: label,
        cate_x: x_test_cat})
   print('step:%d MSE:%.4f' % (i, _loss))
   tf.saved_model.simple_save(sess,"./home/admin/jupyter/math_A/dnnv1/{0}"
   .format(i), inputs={"x": x,"cate_X": cate_x, "y": y},
outputs={"output": output})
```