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	清华大学
参赛队号	**19100030004**
队员姓名	**1.** 杨婧
	2. 谢文彬
	3. 彭程

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：

在无线网络规划的过程中，建立高效精确的无线传播模型对目标通信覆盖区域进行信号强度估算对 5G 网络的部署具有重要意义。本文运用机器学习方法针对无线传播模型建模问题展开研究，首先设计信号传播三维场景并完成特征设计过程，然后运用多种特征评价方法对特征与信号强度目标值的相关性进行评分和排序，最后基于构造和筛选的特征，集成前馈神经网络与梯度提升决策树建立组合模型，实现最终信号功率预测。本模型在华为云平台线上部署运行的 RMSE 测试结果为 9.07421，截至论文提交时排名第一。

问题一：本问题在传统无线传播模型 Cost 231-Hata 的基础上进一步引入专家知识，将数学模拟和物理客观相结合，建立了简化的发射站和接收点的信号传播场景模型。然后，基于场景模型构造四类重要特征：1) 观察点与基站点的距离，2) 信号传播线与观察点-基站点连线夹角，3) 信号传播途中遇到的阻挡物数量与高度等信息，4) 小区地物类型频率统计信息等。最终在原有 17 种特征之外共计增加 18 种新特征，并通过可视化数据集中小区内信号强度分布状况与信号发射机信息直观验证构造特征的合理性。

问题二：本问题根据特征的发散性和特征与目标的相关性两个方面来进行特征筛选。特征的发散程度使用变异系数指标进行评价；特征与目标的相关性使用皮尔森相关系数和最大信息系数统计指标，以及嵌入式的特征打分模型随机森林和 CatBoost 进行评价。然后，基于上述五种评价方式分别对原始特征及问题一的构造特征进行特征评分和排名，并使用五种评价指标的分数加权和作为每个特征的最终评分。最后，根据特征得分与排名信息的分析，可以验证问题一中设计的四类关键特征对于观测点信号功率的预测具有非常重要的意义。

问题三：本问题基于前馈神经网络和梯度提升决策树建立无线智能传播模型。首先，基于问题一和问题二构造和筛选的特征建立多层前馈神经网络的预测模型，同时使用并行方式叠加线性回归模型来有效捕捉较为低阶的交叉特征。此外，为弥补神经网络

收敛困难且可解释性较差的缺点，我们将改进的神经网络结构与梯度提升决策树集成构建组合预测模型，从而有效地结合不同模型的优势，得到了更优异的模型性能表现。

关键词： 无线传播　信号预测　神经网络　梯度提升树模型　特征工程

目录

1. 问题重述

1.1 问题背景

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。无线传播模型正是通过对目标通信覆盖区域内的无线电波传播特性进行预测，使得小区覆盖范围、小区间网络干扰以及通信速率等指标的估算成为可能。由于无线电波传播环境复杂，会受到传播路径上各种因素的影响，如平原、山体、建筑物、湖泊、海洋、森林、大气、地球自身曲率等，使电磁波不再以单一的方式和路径传播而产生复杂的透射、绕射、散射、反射、折射等，所以建立一个准确的模型是一项非常艰巨的任务。

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。经验模型的获得是从经验数据中获取固定的拟合公式，典型的模型有 Cost 231-Hata、Okumura 等。理论模型是根据电磁波传播理论，考虑电磁波在空间中的反射、绕射、折射等来进行损耗计算，比较有代表性的是 Volcano 模型。改进型经验模型是通过在拟合公式中引入更多的参数从而可以为更细的分类场景提供计算模型，典型的有 Standard Propagation Model（SPM）。

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。

近年来，大数据驱动的 AI 机器学习技术获得了长足的进步，并且在语言、图像处理领域获得了非常成功的运用。伴随着并行计算架构的发展，机器学习技术也具备了在线运算的能力，其高实时性以及低复杂度使得其与无线通信的紧密结合成为了可能。

传播模型建立本质上是一个函数拟合的过程，即通过调整传播模型的系数，使得利用传播模型计算得到的路径损耗值与实测路径损耗值误差最小。所以当工程参数、地理位置信息、特定地理位置测量点的 RSRP 已知的情况下，该问题可以归类为一个监督学习问题。请通过机器学习方法（不限定只使用这种方法）来建立无线传播模型，并利用模型来预测在新环境下无线信号覆盖强度，从而减少网络建设成本，提高网络建设效率。

1.2 问题提出

问题 1: 特征设计

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。请根据传统经验信道模型 Cost 231-Hata 以及数据集信息设计合适的特征，并阐述原因。

问题 2: 特征选择

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。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，整理成表格，并阐明设计这些特征的原因和用于排序的量化数值的计算方法。

问题 3: 建立模型预测 RSRP

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。请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。模型的评价包括如下两个指标。

1. 弱覆盖识别率 (PCRR : Poor coverage recognition rate)
2. 均方根误差 (RMSE : Root mean squared error)

2. 模型的假设

1. 信号接收功率 (RSRP) 与工程参数、地理位置之间的未知函数关系可以通过机器学习模型来拟合；
2. 只考虑数据集中提供的信息，不考虑额外因素对信号的影响，比如天气因素等；
3. 假定不同地形环境下信号的衰减强度可以由模型根据数据中地物类型 Clutter Index 信息学习出来，不需要单独为每种地物类型设置场景纠正常数；
4. 假设发射机一直正常工作，不考虑故障情况。
5. 假设接收点信号功率只受一个发射基站的影响。

表 1　论文中用到的符号定义

符号	意义
h_{ca}	小区站点所在栅格 (Cell X, Cell Y) 的海拔高度
h_{cb}	小区站点所在栅格 (Cell X, Cell Y) 的建筑物高度
h_a	栅格 (Cell X, Cell Y) 的海拔高度
h_b	栅格 (Cell X, Cell Y) 的建筑物高度
h_c	小区发射机相对于建筑物的高度
Δh	栅格观察点与基站天线之间的高度差
d_h	栅格观察点与发射机间的水平面欧式距离
d	栅格观察点与发射机天线间的三维欧式距离
α	小区发射机的水平方向角 Azimuth 信息
β_h	发射机信号线与观察点-基站点连线间的水平面夹角
β	发射机信号线与观察点-基站点连线间的水平面夹角
θ	小区发射机垂直机械下倾角和垂直电下倾角之和
CV	特征变异系数 (Coefficient of Variation)
PCC	皮尔森相关系数 (Pearson Correlation Coefficient)
MIC	最大信息系数 (Maximal Information Coefficient)
RF	随机森林模型 (Random Forest)

3. 符号说明

4. 问题分析与求解

4.1 问题一：特征设计

4.1.1 问题分析

问题一的目标是参考 Cost 231-Hata 模型 [1] 和分析数据集中的各类信息完成特征工程的特征设计部分。

传统的 Cost 231-Hata 模型定义如下:

$$\mathrm{RSRP} = \mathrm{P}_t - \mathrm{PL} \tag{1}$$

$$\mathrm{PL} = 46.3 + 33.9\log_{10} f - 13.82\log_{10} h_b - \alpha(h_r) + (44.9 - 6.55\log_{10} h_b)\log_{10}\ d + C_m \tag{2}$$

该模型的核心思路是用发射机的总功率 P_t 减去估计出的传播路径上的损耗值 PL(公式1)。PL 的估计方式基于经验公式 (2)，其中 f 为基站载波频率 (MHz)、h_b 为基站天线有效高度 (m)、h_r 为用户天线有效高度 (m)、$\alpha(h_r)$ 为用户天线高度纠正项 (dB)、d 为链

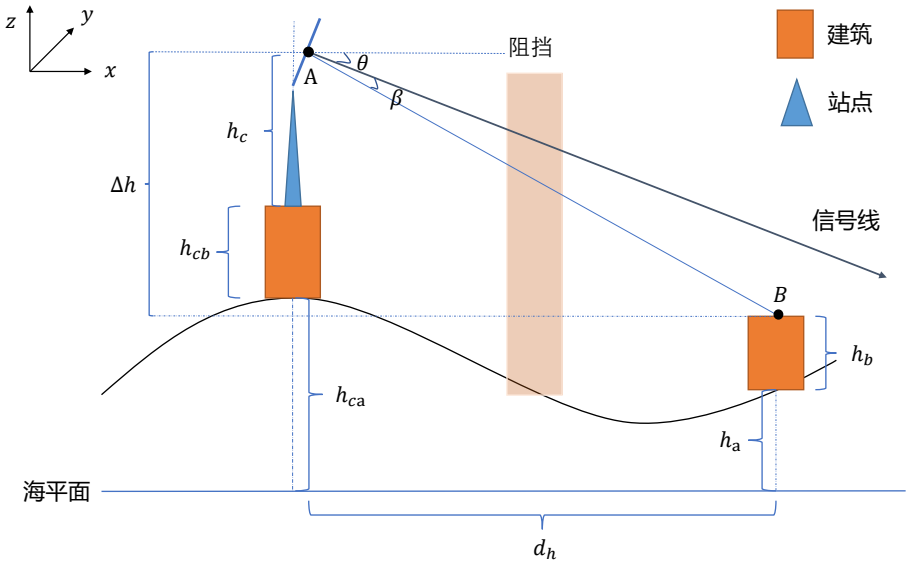

图 1　信号发射机与接收点间信号传播的三维场景模型图示

路距离 (km) 以及 C_m 为场景纠正常数 (dB)。该公式 1) 考虑了用户和基站间的距离，天线的有效高度，并用 log 函数来对这些信息做变换；2) 考虑了基站载波频率，同样使用 log 变换；3) 针对不同场景下信号的损耗强度用不同的场景纠正值来处理。

为便于构造合适的特征，我们在本节设计了考虑信号基站与接收点三维空间位置信息的无线信号传播场景模型，然后通过场景分析构造特征。此外，还利用对数据的可视化分析来直观验证了一些构造特征的有效性。

4.1.2 场景建模

首先对基站的发射天线做如下数学简化：认为基站的发射天线板是一个厚度为 0 的矩形，其底边和地面平行，信号从矩形的正面中点发出（正面为背向发射塔的一面），并且信号具有一定的集中度。因此，我们可以过矩形的中点，沿矩形的中垂线向正面方向画一条射线，此射线即为该基站的信号线。

在图1中，基于上述简化画出一般情况下目标点和基站点的信号传播场景模型。图中海平面所在面为 $\{(x,y,z)|z=0\}$。图中 θ 是机械下倾角 Mechanical Downtilt 与电子下倾角 Electrical Downtilt 之和，$\theta=\theta_{MD}+\theta_{ED}$，即信号线的实际发射角度；$h_{ca}$ 表示小区站点所在栅格 (Cell X, Cell Y) 的海拔高度；h_{cb} 表示小区站点所在栅格 (Cell X, Cell Y) 的建筑物高度；h_c 表示小区发射机相对于建筑物的高度；Δh 表示栅格观察点与基站天线之间的高度差；d_h 表示当前栅格单元与发射机间的水平距离;d 表示当前栅格点 B 与信号天线 A 间的 3D 欧式距离；h_a 表示栅格 (Cell X, Cell Y) 的海拔高度；h_b 表示栅格 (Cell X, Cell Y) 的建筑物高度；α 表示发射机的水平方向角；β 表示信号线与 AB 连线的夹角，记为信号偏转角。

在图1场景模型中，可以定义 A 的坐标为 (x_0, y_0, h_0), B 的坐标为 (x_1, y_1, h_1)，其中 x_0, y_0 为基站点的水平坐标，$h_0 = h_c + h_{cb} + h_{ca}$ 为其绝对高度，x_1, y_1 为目标点的水平坐标，$h_1 = h_b + h_a$ 为其绝对高度。

此外，需要明确因为信号线水平方向角 Azimuth 的存在，图1应该是三维的。因为信号线和直线 AB 构成的平面并不一定垂直于地面，我们无法如同题目中给定的 Cost 231-Hata 模型那般严格定义点 B 到发射线的相对高度，严格来说，是我们无法依据简单的 z 坐标的相对大小给出该相对高度的定义。

4.1.3 特征设计

我们基于上述三维信号传播场景模型来完成特征设计，除了数据集中原先给定的特征列外，主要设计了 4 类特征：基于空间位置、基于信号偏转角、基于阻挡物、基于小区地物类型整体环境等的特征。

1. 基于位置与距离的特征：

基于物理知识，无线电信号的强弱与传播距离存在较大关系。而 A、B 间的距离 d 实际上由 A 和 B 的相对高度差 Δh 以及 a 和 b 的水平距离 d_h 共同决定，因此我们可以把 d，Δh，d_h 都做为模型的特征。

$$d_h = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}, \tag{3}$$

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (h_1 - h_0)} \tag{4}$$

通过 A 和 B 的三维坐标以及天线的水平转角和竖直转角等基本信息，可以计算出距离与各种角度的高级特征，所以我们可以添加基站点的三维坐标 (x_0, y_0, h_0)、观察点的三维坐标 (x_1, y_1, h_1)、天线的水平转角 (α) 和竖直转角 (θ) 做为模型的特征。

2. 基于信号偏转角的特征：

基站天线发射的信号具有一定集中度，其合理性较为明显，比如天线背面的信号通常比正面相同角度相同距离的信号弱，即信号的强度与信号线和 AB 连线的夹角大小有关。当其它条件相同时，AB 连线与信号线夹角 β 越小，B 点的信号越强。在示意图2中，以基站为圆心做圆，在圆周上的点与基站连线与信号线夹角越小，则信号越强。我们定义信号偏转角来对这种特征进行建模。

由于数据中给定的发射机水平方向角 Azimuth，即 α 是从 Y 轴正方向开始顺时针旋转不断增大的，我们先将其变为二维平面常用的从 X 轴正方向逆时针开始增大的 α':

$$\alpha' = (2\pi - \alpha) + \pi/2 \tag{5}$$

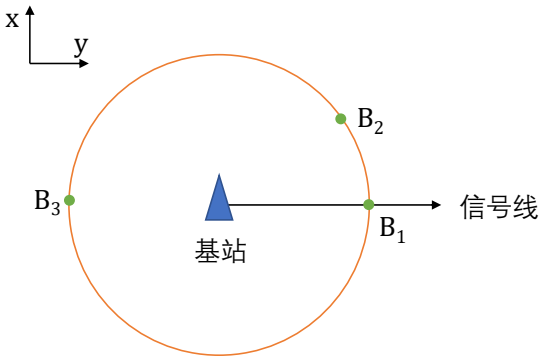

图 2 基站信号线直接指向的方向上的观测点的信号接收功率较强。在基站点的等半径圆上，观测点-基站点连线与信号线的夹角越大的观测点信号越弱 ($B_1 > B_2 > B_3$)。

则信号线的向量 $\vec{v}_1$ 和 AB 的向量 $\vec{v}_2$ 可以表示为:

$$\vec{v}_1 = (\cos\alpha', \sin\alpha', -\tan\theta), \tag{6}$$

$$\vec{v}_2 = (x_1 - x_0, y_1 - y_0, h_1 - h_0) \tag{7}$$

接下来，信号线 $\vec{v}_1$ 与 AB 对应的 $\vec{v}_2$ 这两个向量在 3D 环境里夹角 β 可以通过其余弦来表示如下：

$$\cos\beta = \frac{\vec{v}_1\vec{v}_2}{||v_1|| \cdot ||v_2||}$$

此外，$\vec{v}_1$ 与 $\vec{v}_1$ 在水平面上的夹角 β_h 可以通过两向量的 x, y 分量用上述相同方式算出。

将 3D 下的 β 与水平面的 β_h 都记为信号的水平偏转角，作为重要特征加入数据训练过程。

3. 基于阻挡物的特征：

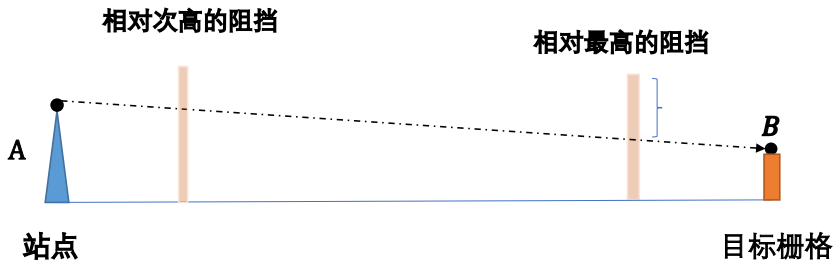

图 3 观测点的信号强度会受到观测点-基站点连线上阻碍物状况的影响。阻碍物的数量，高于信号传播线的高度，与基站点之间的距离等都可以作为特征。

为便于对传播路径的损耗进行建模，上述无线信号传播模型增加了阻挡物所带来

的信号衰减影响。当信号传播的时候遇到阻挡物，会有部分信号被反射、吸收，造成信号强度降低。因此传播过程中阻挡物的数量，阻挡物的种类，阻挡物的高度，阻挡物超出 AB 部分高度占阻挡物高度的比例等都会影响信号的传播。我们对这些特征做了建模以便于学习模型对观测点的功率预测。

图3直观地画出了信号由基站到目标栅格的连线上有阻挡物时信号会受影响的情况。更为严格地，我们定义阻挡物如下：

一个栅格为站点 A 和目标栅格 B 的阻挡物，当且仅当，水平面上它位于 AB 的之间并且它的绝对高度超过了 AB 所在直线在该点的高度。

阻挡物的数目可能是任意的。当有多个阻挡物的时候，我们只关心相对最高的那个阻挡物，而不是绝对最高的那个，比如当阻挡物恰好位于目标栅格正前方，它的阻挡效果很可能比其他更高，但离目标栅格更远的阻挡物更明显。更为严格地定义，给定 A,B 的两个阻挡物 a, b，令 $r(x)$ 表示 a 到 AB 所在直线的距离除以 a 的高度，如果 $r(a) > r(b)$, 那么 a 相对高于 b。我们也称那个相对最高的阻挡物为最坏阻挡。所以最后的特征包括阻挡物数目，最坏阻挡的种类，最坏阻挡的坐标，最坏阻挡离 AB 直线的高度等。

4. 小区地物类型等其余特征：

小区内观测点和基站点的地形种类环境，即 Clutter Index 信息也会影响信号强弱，比如开阔地带的信号往往比建筑密集区较好，这些地形信息也应该作为重要的特征供机器学习模型学习，因此，我们统计了小区内的各种地物类型出现频率来作为备选特征。

可以注意到 Cost 231-Hata 模型中使用了的距离信息都以 10 为底取了对数，因此我们也可以将 $\log_{10} d$，$\log_{10} d_h$ 等作为额外的特征。此外，特征中存在不少角度，而几何上角度的作用往往体现在三角函数中，由于机器学习模型其实比较难拟合三角函数，因此我们还可以把角度信息的正弦、余弦值作为新特征。

表 2　数据集中给定的所有特征与标签列以及本文构造出的新特征

特征来源	字段名称	含义	单位
原有	Cell Index	小区唯一标识	-
	Cell X	小区所属站点的栅格位置，X 坐标	m
	Cell Y	小区所属站点的栅格位置，Y 坐标	m
	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
	Cell Building Height	小区站点所在栅格 (Cell X, Cell Y) 的建筑物高度	m
	Cell Altitude	小区站点所在栅格 (Cell X, Cell Y) 的海拔高度	m
	Cell Clutter Index	小区站点所在栅格 (Cell X, Cell Y) 的地物类型索引	-
	X	栅格位置，X 坐标	m
	Y	栅格位置，Y 坐标	m
	Building Height	栅格 (X,Y) 上的建筑物高度	m
	Altitude	栅格 (X,Y) 上的海拔高度	m
	Clutter Index	栅格 (X,Y) 上的地物类型索引	-
	RSRP	栅格 (X, Y) 的平均信号接收功率，标签列	dBm
构造	dis_2d	小区观察点与基站点的水平面欧式距离	m
	dis_3d	小区观察点与基站点的三维欧式距离	m
	dis_2d_log	对 dis_2d 做 log 变换	Deg
	dis_3d_log	对 dis_3d 做 log 变换	Deg
	hori_beta	基站信号线与观察点-基站点连线间水平夹角，信号偏转角	Deg
	beta	基站信号线与观察点-基站点连线间三维夹角，信号偏转角	Deg
	obstacle_num	观察点-基站点连线上的阻挡物数量	-
	worst_obstacle_height	观察点-基站点连线上最高阻挡物的高度	m
	worst_obstacle_delta_d	观察点-基站点连线上最高阻挡物与基站点的距离	m
	worst_obstacle_height_rate	观察点-基站点连线上最高阻挡物超过连线部分高度占总高度比	-
	worst_obstacle_y	观察点-基站点连线上最高阻挡物 X 坐标	m
	worst_obstacle_x	观察点-基站点连线上最高阻挡物 Y 坐标	m
	clutter_frequency	小区内各种地物类型的出现频率	-
	delta_height	观察点与基站点间的高度差	m
	abs_height	观察点的绝对高度	m
	cell_abs_height	基站点的绝对高度	m
	downtilt	信号线下倾角，垂直电下倾角与机械下倾角之和	Deg
	frequency_band_log	对小区发射机中心频率做 *log* 变换	-

表中列出了数据集中原有的 17 个特征与 1 个标签的信息，以及新设计的 18 个特征的描述。为便于区分，新设计的特征的名字中字母开头为小写，且用下划线'_' 间隔单词，不同于原始特征中首字母大写，以空格' ' 间隔单词的方式。

4.1.4 特征设计结果

原始数据中的特征与标签列和本论文中构建的特征汇总于表2。

原始数据集中包含 17 个特征和一个标签，可以分为两类：1) 小区基站点的信息，包括基站位置与高度、基站的发射功率、频率、基站天线的下倾角与水平方向角等；2) 栅格观测点的信息，包括栅格点的位置与高度、地物类型等。标签为 RSRP，即当前栅

格观测点的平均信号接收功率，为模型最终需要预测的值。

新构造的特征主要有 18 个，包括：1) 距离与位置特征，包括栅格观测点与小区基站点间的水平面距离、三维距离、高度差等；2) 信号偏转角特征，包括基站信号线与观测点-基站点连线的水平面夹角，三维夹角；3) 信号传播路径上遮挡物特征，包括遮挡物的个数、最高遮挡物高度与位置等；4) 小区内各种地物类型出现频率，以及观测点与基站点间距离、基站信号频率的 log 变换特征，信号线两下倾角之和，计算观测点和基站点绝对高度等特征。

新构造的特征中考虑了观测点接收信号的距离、信号线与观测点-基站点连线夹角、信号传播路径上的阻挡物等信息。信号传播距离越远，信号肯定越弱；信号线与观测点-基站点连线的夹角 (0～180°) 越大，则信号越弱；信号传播路径上阻碍物越多，则信号损耗越大。这些特征考虑的因素和信号传播过程密切相关，符合物理规律。

4.1.5 特征设计合理性验证

我们通过数据集中信息的可视化来验证设计特征的合理性。在图4 中以小区 1021701 和 1030301 为例画出了小区所有栅格点上的信号接收功率 RSPR 的分布状况以及小区基站发射机的信号线角度。暖色调 (偏红) 代表 RSPR 值较强，冷色调 (偏蓝) 代表 RSPR 较弱。基站位置处标了黑点，基站信号线的水平方向角 α 以黑色的向量箭头来表示。

通过观察小区 1021701 和 1030301 的 RSPR 值分布情况，可以发现如下两点信息：

1. 离基站发射机位置越近的地方信号越强。与发射机位置越远，信号功率越偏向冷色调，即信号越弱。
2. 信号线指向的方向上信号较强。以图中小区 1021701 和 1030301 为例，黑色箭头即信号线指向的地方信号强；观察点-基站点连线与信号线方向之间夹角越大，色调越偏冷，即信号功率越小。

以上两个信息分别验证了特征设计时两类特征：1) 基站点-观测点距离 dis_2d、dis_3d 等特征；2) 信号线与观测点-基站点连线夹角，即 beta、hori_beta 的构造是非常有价值的。

4.2 问题二：特征选择

4.2.1 问题分析

基于问题一中构造的特征，本问题需要对它们进行评价和选择。特征的评价通常包括如下两种基本指标：

1. 特征自身发散程度。
2. 特征与预测目标的相关性。

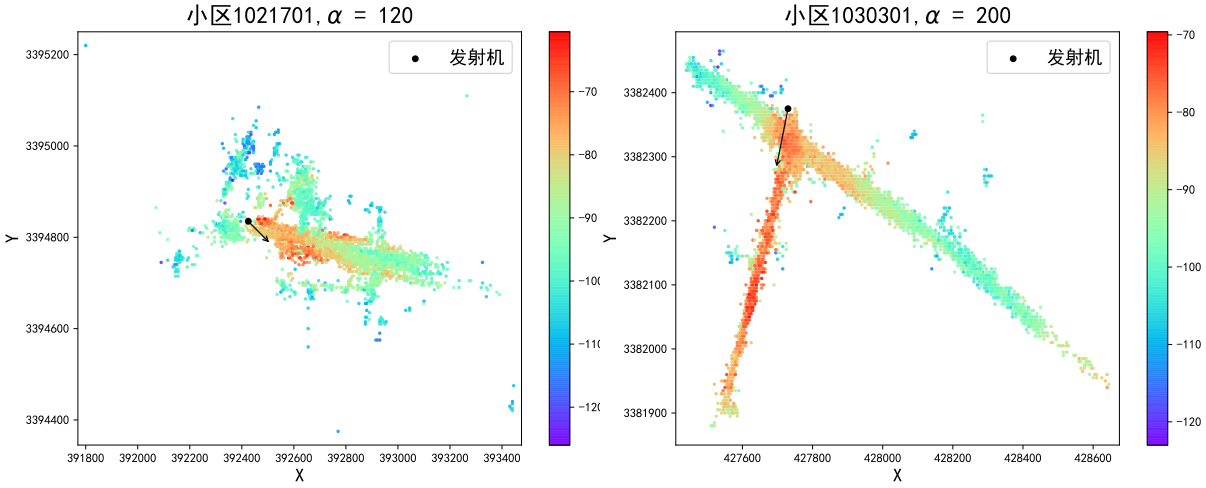

图 4　小区不同位置处信号接收功率 RSRP 的分布与发射机位置及水平方向角 α(黑色箭头) 的图示。a. 在信号线直接指向的方向上 RSRP 较大；b. 距离基站点较近的地方 RSPR 较大。

常见的特征筛选方法大致可以分为三类：过滤式 (filter)、包裹式 (wrapper) 和嵌入式 (embedding) [2]。

1. 过滤式特征筛选直接用一些统计方式来对特征进行打分并排名，排名越靠前的特征价值越大。由于在过滤式的特征筛选过程中不涉及到学习模型，因此其计算速度很快。常见的过滤式方法有 T 检验 (*t*-test) [3], 卡方检验 (χ^2 test) [4], 皮尔森相关系数 (Pearson Correlation Coefficient，PCC)，最大信息系数 (Maximal Information Coefficient, MIC) [5] 等。
2. 包裹式的特征评价方法通过在某些特征上做训练的机器学习模型的预测效果来对这些特征做打分，以此来评价和筛选不同的特征组合。通常使用遗传算法 (GA) [6]，粒子群算法 (PSO) [7] 等群智能算法来不断迭代优化生成的特征子集，但由于学习模型要不断重新训练及做预测，因此时间开销较大。
3. 嵌入式特征筛选方式将机器学习模型的训练过程与特征筛选过程融为一体，在模型的训练过程中自动进行了不同特征的重要性比较来完成特征的打分与筛选。

过滤式特征评价方式的优点是方便地使用传统统计学度量方式来快速地对特征重要性进行判断，结果的质量有统计学的理论分析保证，但是其缺点是独立地考察每一个特征与目标变量的相关性，忽略了不同特征之间的关联信息与组合效果。包裹式和嵌入式的特征评价方法可以考虑到不同特征组合产生的效果来更好地评价特征重要性，但是计算时时间开销往往较大。

表 3 不同特征在不同评价指标下的得分大小。每个评价指标的绝对值总分为 100，然后分到各个特征上。

特征	PCC 评分	MIC 评分	RF 评分	CatBoost 评分	CV 评分
dis_2d_log	-18.692	11.356	9.637	6.399	0.055
dis_2d	-10.244	11.287	9.635	6.923	0.464
dis_3d_log	-18.559	3.214	9.556	3.023	0.052
hori_beta	7.162	6.872	3.233	7.829	0.543
dis_3d	-10.211	3.204	9.556	2.037	0.461
cell_clutter_frequency	3.416	5.344	4.605	10.146	3.689
beta	7.405	3.914	2.395	3.976	0.503
worst_obstacle_height_rate	0.078	5.957	2.590	4.702	59.559
worst_obstacle_height	3.409	3.942	3.475	3.079	0.128
obstacle_num	-2.828	4.737	2.921	3.200	0.655
Clutter Index	0.449	1.044	3.646	4.974	29.410
Building Height	-2.565	2.221	1.864	3.917	0.940
Y	-0.702	3.237	1.294	4.317	0.229
X	-0.608	3.355	1.251	4.129	0.226
Height	-0.098	2.122	1.215	5.451	0.108
Cell Altitude	0.440	1.719	1.199	4.800	0.006
worst_obstacle_delta_d	-1.448	2.830	3.936	0.000	0.392
Cell Y	-1.198	1.940	1.930	2.920	0.215
Cell Index	-0.070	3.193	4.481	0.181	0.151
Cell X	-0.837	1.945	1.895	2.887	0.215
delta_height	-1.721	2.626	1.219	0.991	0.227
RS Power	-0.785	1.242	1.207	3.130	0.058
worst_obstacle_y	-1.193	2.477	2.170	0.000	0.238
Cell Clutter Index	-0.975	0.960	1.204	1.928	0.121
worst_obstacle_x	-0.776	2.347	1.918	0.000	0.231
Azimuth	0.172	1.522	1.196	1.968	0.157
abs_height	-1.838	1.069	1.341	0.412	0.009
Altitude	0.387	1.288	1.219	1.366	0.006
cell_abs_height	0.202	1.069	1.173	1.600	0.009
Cell Building Height	-0.007	0.629	1.234	1.406	0.524
Mechanical Downtilt	0.255	0.421	1.241	0.724	0.184
Electrical Downtilt	-0.491	0.218	1.285	0.582	0.122
downtilt	-0.146	0.568	1.262	0.562	0.115
frequency_band_log	-0.316	0.065	1.008	0.222	0.000
Frequency Band	-0.316	0.066	1.008	0.219	0.001

4.2.2 评价方式

针对特征自身的发散程度的评价，我们采用了变异系数 (Coefficient of Variation , CV) 指标来完成。CV 的定义方式是数据 X 的标准差 σ_X 和均值 μ_X 的商 (公式8)，是数据离散程度的一种非常有效的归一化量度方式 [8]。

$$cv = \frac{\sigma_X}{\mu_X} \tag{8}$$

表 4 不同特征在各个评价指标下的排名高低

特征	PCC 排名	MIC 排名	RF 排名	CatBoost 排名	CV 排名
dis_2d_log	1	1	1	4	28
dis_2d	3	2	2	3	9
dis_3d_log	2	11	4	16	29
hori_beta	6	3	10	2	6
dis_3d	4	12	3	19	10
cell_clutter_frequency	7	5	5	1	3
beta	5	8	13	11	8
worst_obstacle_height_rate	33	4	12	8	1
worst_obstacle_height	8	7	9	15	22
obstacle_num	9	6	11	13	5
Clutter Index	23	28	8	6	2
Building Height	10	18	18	12	4
Y	20	10	20	9	14
X	21	9	23	10	16
Height	32	19	28	5	26
Cell Altitude	24	22	31	7	32
worst_obstacle_delta_d	13	14	7	33	11
Cell Y	14	21	15	17	18
Cell Index	34	13	6	32	21
Cell X	17	20	17	18	17
delta_height	12	15	26	25	15
RS Power	18	25	29	14	27
worst_obstacle_y	15	16	14	34	12
Cell Clutter Index	16	29	30	21	24
worst_obstacle_x	19	17	16	35	13
Azimuth	30	23	32	20	20
abs_height	11	26	19	29	30
Altitude	25	24	27	24	33
cell_abs_height	29	27	33	22	31
Cell Building Height	35	30	25	23	7
Mechanical Downtilt	28	32	24	26	19
Electrical Downtilt	22	33	21	27	23
downtilt	31	31	22	28	25
frequency_band_log	27	35	35	30	35
Frequency Band	26	34	34	31	34

针对特征与目标相关性，我们采用了两种过滤式方法与两种嵌入式方法来评价特征。

两种过滤式特征评价方法：皮尔森相关系数 PCC 和最大信息系数 MIC。PCC 是用来反映两个变量线性相关程度的经典统计量，相关系数用 ρ 表示，为两个变量 X, Y 的协方差 $cov(X, Y)$ 除以它们的标准差 σ_X 和 σ_Y，结果处于-1 到 1 之间，ρ 的绝对值越大表明相关性越强 (公式9)。MIC 是 2011 年发表在 Science 期刊上的一个基于信息论来计算不同变量间关联关系强弱的方法，可以检测出两个变量间各种线性或者非线性关系，在很多大型数据集上都有优秀表现 [9, 10]。

表 5　不同特征在关于目标的综合相关性分数。相关性总分为 100，分到了各个特征上。

排名	特征	与目标相关性
1	dis_2d_log	11.406
2	dis_2d	9.432
3	dis_3d_log	8.503
4	hori_beta	6.217
5	dis_3d	6.194
6	cell_clutter_frequency	5.856
7	beta	4.383
8	worst_obstacle_height_rate	3.894
9	worst_obstacle_height	3.443
10	obstacle_num	3.394
11	Clutter Index	2.797
12	Building Height	2.625
13	Y	2.366
14	X	2.315
15	Height	2.201
16	worst_obstacle_delta_d	2.037
17	Cell Altitude	2.019
18	Cell Y	1.979
19	Cell Index	1.963
20	Cell X	1.874
21	delta_height	1.625
22	RS Power	1.576
23	worst_obstacle_y	1.448
24	Cell Clutter Index	1.255
25	worst_obstacle_x	1.250
26	Azimuth	1.204
27	abs_height	1.154
28	Altitude	1.054
29	cell_abs_height	1.001
30	Cell Building Height	0.816
31	Mechanical Downtilt	0.655
32	Electrical Downtilt	0.639
33	downtilt	0.629
34	frequency_band_log	0.399
35	Frequency Band	0.398

$$\rho = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{9}$$

两种嵌入式特征评价方法：随机森林 (Random Forest, RF) [11] 和 CatBoost 集成树模型 [12]。随机森林是经典的基于 bagging 方式做集成来提升基本决策树模型性能的集成模型；CatBoost 是 2017 年被发布的基于 boosting 方式逐步迭代决策树模型来提高拟合效果的并行计算模型，能很好地处理类别型特征。我们在数据集上完成随机森林和

CatBoost 模型的训练之后，就可以直接输出模型对各个特征的相对重要性打分。

关于特征与目标之间的相关性，我们定义为 5 种评价指标打分的加权评价 (权重总和为 1)，通过整合不同评价方式的结果来对特征给出一个综合性的目标相关评分。CV 评分反映了特征的发散度，即信息量，但是是在一个特征数据列内部计算的，而且不同特征在差异系数 CV 的评价方式下得分相差较大，所以我们对 CV 得分只分配了 0.01 的权重 (w_5)，将剩下的权重平均分配给了 PCC、MIC、RF 和 CatBoost。

$$Score = w_1 \cdot S_{\text{PCC}} + w_2 \cdot S_{\text{MIC}} + w_3 \cdot S_{\text{RF}} + w_4 \cdot S_{\text{CatBoost}} + w_5 \cdot S_{\text{CV}} \tag{10}$$

4.2.3 评价结果

使用上述 5 种评价指标，对所有特征进行打分的结果见表3。已经对每个评价指标输出的得分进行了缩放来便于观察。每个评价指标的绝对值总分为 100，将这个总分分配到各个特征上来完成评分。特征获得的评分绝对值越大，则这一特征越重要。

表3中 PCC 给出评分的正负号代表某个特征与预测目标值是正相关还是负相关，绝对值反映相关性强弱。比如 dis_2d_log、dis_2d、dis_3d_log 和 dis_3d 这 4 个特征与目标值信号接收功率 RSRP 成很强的负相关，即符合离基站距离越远信号越弱的规律，而信号水平偏转角 beta 和 hori_beta 与目标成正相关，是因为这两个角我们用的是其 cos 余弦值，角度在 0 到 180 度范围内越大，则对应余弦值越小，此时对应的信号的强度也越小。

在表4中列出了所有特征在不同评价指标下的排名高低。可以发现前 4 列主要用来针对特征和目标相关性进行评价的指标 PCC、MIC、RF、CatBoost 对不同特征的排名结果具有一定的一致性。重要的特征在不同指标的排名结果里都会比较靠前，而不重要或信息量少的特征在不同指标的排名结果里都比较低。比如观测点与基站点间 2D 平面距离 dis_2d 在四种指标里的排名分别为 3、2、2 和 3，具有高度的一致性。与此相反，对基站的发射机信号频率 Frequency Band 与其 log 变换下的 frequency_band_log 这两个特征，我们查表3可以看出其 CV 评分非常低，几乎为 0，即它们的发散度非常小，在该特征中的信息很少，因此在表4的特征排名里被几种指标都排在了最靠后的位置。

使用公式10对 5 种评价方式的得分加权计算特征与目标综合相关性，并依据相关性高低给出的特征排名列在了表5里。

从最终排名表5可以看出在特征设计阶段很多精心设计的特征与目标的相关性非常强。排在最前面的是发射点-基站点 2D 与 3D 距离：dis_2d_log、dis_2d、dis_3d_log 和 dis_3d 即距离远近会对信号强度影响非常强；接下来，是信号偏转角 beta 和 hori_beta，即信号线与观测点-基站点连线的夹角对信号强度的影响也很重要；然后，统计小区内

不同地物类型如开阔区，建筑密集区出现频率的特征 cell_clutter_frequency 也很重要，说明不同小区地理环境的类型会影响信号传输过程中的损耗强度，并影响接受信号功率；最后，关于信号传播过程中阻挡物 obstacle 相关的特征排名也很靠前，即阻挡物数量 obstacle_num、阻挡物高度 worst_obstacle_height 等特征对于信号传播过程影响很大，与信号功率预测有较大的帮助。

4.3 问题三：模型建立与 RSRP 预测

4.3.1 问题分析

本问题需要基于上述问题一特征的设计构造和问题二的特征选择，建立预测模型来对不同地理位置的 RSRP 进行预测。

预测问题的本质是构造或学习获得问题输入与输出间的映射函数 $f:\mathbb{R}^n \to \mathbb{R}$，常见解决思路有回归分析、灰色预测模型、神经网络等多种方法。而无线电波的传播由于受到传播路径上建筑物、山体、海洋、森林、大气、地球自身曲率等多种因素的影响，使得上述映射变为十分复杂的函数，实际上很难使用无线传播的理论和经验模型或传统的预测方法准确拟合，同时由于受限于华为平台的要求，本问题中我们主要使用基于 Tensoflow 实现的多层前馈神经网络作为问题求解的主要模型。

前馈神经网络的来源于受生物和神经学科启发的一系列数学模型。早期的 McCulloch 和 Walter Pitts 神经元理论 [13] 通过检验函数的正负来识别不同类型的输入。神经元单元模型如图5所示，其接收 n 个输入信号，通过权重连接传递，生成最终输入 z 并将其与神经元的阈值相比较，最终经过激活函数 $a=f(x)$ 处理生成输出。

然而，神经元模型正确的输出值严重依赖于有效的模型权重，而这时的模型权重只能人工设定。随后出现的二元分类感知机 [14] 是该领域最早能够根据不同类别的输入样本来学习权重的模型，但感知机模型由于仅具有单层的网络结构，本质上仍是线性模型，只能求解线性映射，存在很大局限性。1986 年 Rumelhart 等人重新提出 Werbos 的反向传播算法 [15]，使得多隐层神经网络的训练实现成为可能。1989 年 Yann LeCun 使用美国邮政局数据建立的手写数字识别模型展现出使用神经网络解决复杂实际问题的能力 [16]。这一时期各种不同类型的神经网络模型相继出现，例如径向基网络、竞争型 ART 网络、自组织映射网络、Elman 网络以及玻尔兹曼机等。然而，由于硬件和理论等条件的限制，这一时期的网络深度较浅，神经网络的优势并不显著。2012 年 AlexNet 网络相较之前的网络实现深度和参数数量的突破，并提出 Dropout 防止网络出现过拟合的技术，为神经网络深度增加方向的发展带来突破。2014 年，Oxford VGG research group 提出 VGG16/19[17] 进一步增加网络深度，同年 Google 提出 GoogLeNet 及其延伸版本 Inception[18] 引入稀疏网络结构设计，2016 年微软提出 ResNet 引入残差，有效解决了

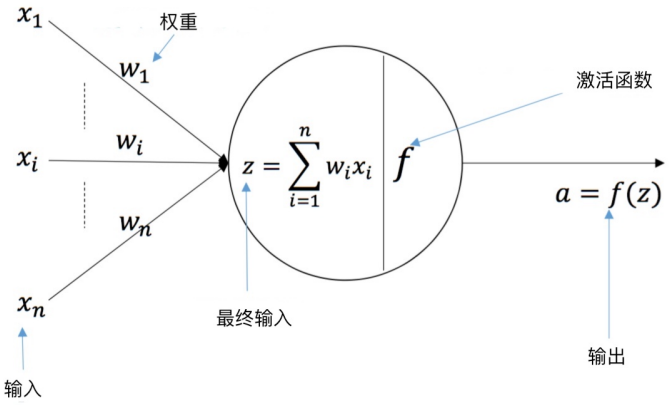

图 5　神经元模型图示

随着网络深度增加而产生的梯度消失问题 [19]。

反向传播算法的本质是将最小均方误差算法推广应用于非线性可微激活函数组成的多层感知机。Hornik 等人证明，对于包含足够多神经元的隐层，多层前馈网络能够以任意精度逼近任意复杂度的连续函数 [20]。因此，通过应用非线性激活函数的基本神经元的组合和多层网络中大量神经元间的相互连接作用，深层的前馈神经网络可看作高度非线性的模型，从而可以确保实现对本问题中无线传播预测的复杂非线性映射的准确拟合。

然而，神经网络也有局限性：随着神经网络模型网络深度增加，模型容量与复杂度随之提升，模型参数增多导致调优复杂且模型可解释性降低，同时更容易发生过拟合。而机器学习中另一大类算法——树模型，则可以通过学习数据的分层结构实现根据数据集的性质调整模型的复杂度，在数据量有限且特征质量一般的情况下相较于神经网络更具优势，并且具有更好的模型可解释性。这类算法的经典模型是 1963 年提出的决策树模型，其通过计算节点分裂时的纯度变化和剪枝可自动实现特征的选择。但单颗决策树往往存在稳定性差、不同预测样本结果方差高的问题。因此实际应用通常使用随机森林或梯度提升决策树等集成树模型，两者分别通过并行和串行的模式集成多颗决策树以有效降低模型的方差。

综上所述，我们使用具有复杂非线性拟合能力的前馈神经网络模型和梯度提升决策树的模型，并将两者组合作为我们本问题的预测模型。

4.3.2 模型建立

基于问题一和问题二中构造和选择的特征，我们首先使用深层的多层前馈神经网络模型并自行设计和优化网络的整体架构以及激活单元、层数和学习率等超参数，应用基于梯度下降的反向传播算法学习得到 RSRP 的前馈神经网络预测模型，同时与梯度

提升决策树模型的预测结果集成建立最终的组合预测模型。

实验数据设计

本问题采用题目提供的所有数据进行实验和测评。数据包含训练数据部分的 4000 个小区数据和测试数据部分的 2 个小区数据。原始的数据特征除预测量实际平均信号接收功率外，按特征类别可以分为工程参数数据和地理位置信息；按特征主体可以分为小区所属站点的特征和栅格的特征信息，包含索引值共计 17 维。此外，我们还使用问题一和问题二构造筛选后的特征进行实验，经过预处理转换后维度共计 79 维。

不同小区的训练数据样本合并后共 12011833 份，我们将数据集按照均匀分布随机分为 100 份，使用一份数据作为本地验证集，其余数据作为训练集。在训练集上建立 RSRP 预测模型并在验证集上对 RSRP 进行预测，统计 RMSE 评测指标衡量模型效果。为避免预测模型的过拟合，我们进行多次试验并在每次试验使用不同的验证数据集，将多次实验的 RMSE 平均值作为最终的评测指标。

数据预处理

在模型算法学习前需要对数据的特征进行标准化、类别特征编码、缺失值和异常值处理四部分预处理工作。数据标准化能够有效加速模型学习时的收敛速度，因此我们对数据集中的坐标特征按下列公式进行标准化化处理：

$$X = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{11}$$

数据中的类别特征 Cell Clutter Index, Clutter Index 等在传入模型前需要进行哑编码，本问题中我们对其手动进行 one-hot 编码，将每个特征变为 20 维特征的稀疏编码。

经过缺失值和异常值检查可以发现，在 12011833 份数据样本中不存在缺失值，但存在部分异常数据需要剔除。为确保训练数据的准确性，我们通过对实际平均信号接收功率 (RSRP) 进行统计分析来完成数据合理性的判断，RSRP 的分布图如图6所示。

由于与基站发射机相距太远的数据失真较大，不适合作为训练样本，根据上图中的 RSRP 数据分布，对于 RSRP $< -130dB$ 的数据，我们将其作为异常样本剔除。此外，我们还通过对信号功率强度与距离基站距离以及建筑物高度和地物类型标识等关联特征的负相关程度进行检查，实现数据样本的进一步异常检查和清洗。

模型设计

我们多层前馈神经网络的基础网络结构设计四层的前馈神经网络，包括输入层，三层隐藏层和输出层。激活函数选择非线性的整流线性单元（ReLU）：

$$g(z) = \max(0, z)$$

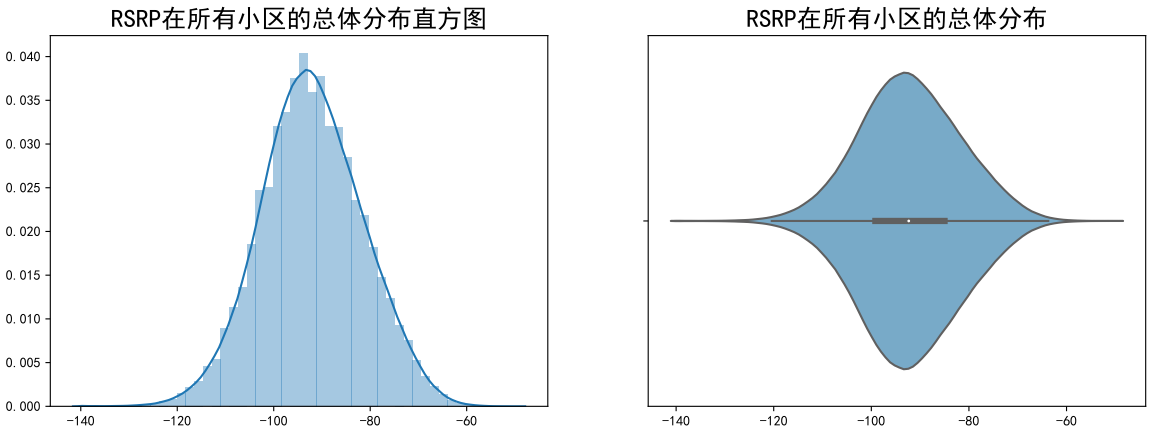

图 6　所有小区信号接收功率 RSRP 的分布示意图。

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$

$$\mathrm{PReLU}_i(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma_i x & \text{if } x \leq 0 \end{cases}$$

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

$$= \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

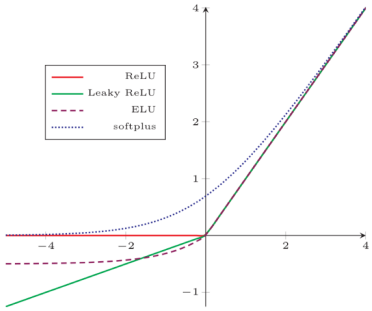

图 7　ReLU 激活函数及其变种

整流线性单元及其改进函数的图像如图7所示。虽然 ReLU 在 $z = 0$ 处不可微，但其二阶导数几乎处处为零，且在处于激活状态是一阶导数恒为 1，因此相比于 sigmoid，tanh 等虽然可微的但引入二阶效应的非线性激活函数而言，它的梯度方向更具价值，在实际使用的多数网络中都具有更好的效果表现。

网络超参数的设置对于模型预测的准确度具有重要影响。对于网络隐层节点个数的配置，我们基于以下公式 (11)，（12），（13）和（14）确定不同类型节点数的初始设置。

$$m < N - 1 \tag{12}$$

$$m = \log_2 n \tag{13}$$

$$m = \sqrt{n + l} + a \tag{14}$$

$$m = \sqrt{nl} \tag{15}$$

其中，N 为训练样本数，m 为隐层节点数，n 为输入层节点数，l 为输出层节点数，a 表示到 1 到 10 之间的常数。然后我们尝试采用贝叶斯优化尝试对除输入层以外每层的节点数进行优化，并使用网格搜索进行超参数调优，最终确定使用如表6和表7所示的网络节点配置和超参数初始值设置。对于超参数中的学习率（learning rate），我们设置了学习率的指数衰减以帮助网络收敛，对于超参数中的批处理大小（batch size）和训练轮次（epoch num）我们采用早停技术避免网络的过拟合。

表 6　网络节点配置

层数	节点数	激活函数
input layer	79	ReLU
layer1	2048	ReLU
layer2	1024	ReLU
layer3	256	ReLU
output layer	1	ReLU

表 7　网络超参数设置

参数名称	参数值
learning rate	2e-5
batch size	1280
epoch num	100

虽然多层的前馈神经网络具有很强的非线性表征和学习能力，但它所学习到的高阶特征通常隐含地体现在隐藏层中，此时泛化到隐式的特征交互往往是十分困难的，而我们认为特征的交叉组合对模型的表示能力具有重要的意义。因此，在上述的网络结构中，我们希望能够对低阶的特征组合进行单独的显式建模。基于这一想法，我们借鉴 Wide and Deep 模型 [21] 和 DCN[22] 中的思想对基础的多层前馈神经网络进行改进，使用并行的方式叠加一个 LR 网络，为其输入类别特征中 Clutter Index 和 Cell Clutter Index 的点积交叉特征，并将输出结果与上述基础网络的输出叠加作为最终输出，使用联合学习对模型进行训练。改进后的网络结构能够有效利用一阶特征和低阶的交叉特征并避免了低阶交叉特征的手工特征工程过程和穷举搜索。最终的网络整体架构图如图8所示。

学习算法

模型的损失函数定义为均方误差函数 MSE（Mean Squared Error），其公式如下：

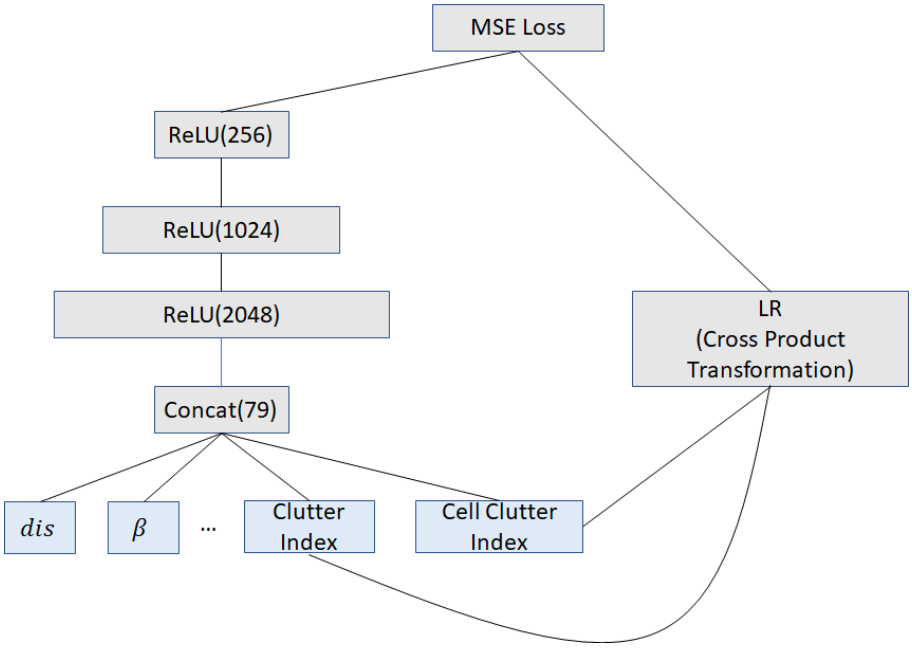

图 8　神经网络模型的网络整体架构

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

MSE 定义参数估计值和参数真实值间差值的平方的数学期望，用于衡量数据的偏离程度。损失函数取最小值时所对应的参数值即所需求解的模型参数的最优值，然而参数寻优的过程通常存在局部最优解和全局最优解两种最优情况。为避免参数寻优陷入局部最优解，模型的优化算法通常会选取模拟退火、遗传算法等启发式方法或随机梯度下降以及带动量的随机梯度下降等算法。本问题的模型实现中，我们选择使用带动量的随机梯度下降方法，并使用反向传播算法实现多层前馈神经网络的误差反向传播，算法 4.1 给出模型使用的基于随机梯度下降进行误差反向传播的算法流程。

除上述前馈神经网络的模型外，我们还建立了一个梯度提升决策树的模型并使用 Catboost 的实现以便对类别型特征变量提供更好的编码和特征表示方式，Catboost 模型的参数设计如表8所示。

最后，我们将神经网络模型和集成决策树的模型通过线性融合方式集成得到本问题最终的组合预测模型。

算法 4.1 1 反向传播算法

输入: 训练集 $D=\{(\mathbf{x}_n,\mathbf{y}_n)\}_{n=1}^{N}$，验证集 $\mathbb{V}$，学习率 α，正则化系数 λ，网络层数 $\mathbb{L}$，神经元数量 $m^{(l)}$，$1\leq l\leq L$

输出: W，b

随机初始化 W，b

repeat 对训练集中所有样本随机重排序

　　for all $(\mathbf{x}^{(n)},\mathbf{y}^{(n)})\in D$ **do**

　　前馈计算当前样本每一层的输出和激活值 $\hat{\mathbf{y}}_k=f(\beta_j-\theta_j)$

　　反向传播计算每一层的误差 $\delta^{(l)}$

　　计算每一层的参数导数

$$\frac{\partial\mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial W^{(l)}}=\delta^{(l)}(a^{(l-1)})^{\mathrm{T}}$$

$$\frac{\partial\mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial b^{(l)}}=\delta^{(l)}$$

　　更新权值

$$W^{(l)}\leftarrow W^{(l)}-\alpha(\delta^{(l)}(a^{(l-1)})^{\mathrm{T}}+\lambda W^{(l)})$$

$$b^{(l)}\leftarrow b^{(l)}-\alpha\delta^{(l)}$$

　　end for

until 神经网络模型在验证 $\mathbb{V}$ 上的错误率不再降低

表 8　Catboost 模型参数设置

参数名称	参数值
iterations	10000
depth	10
learning rate	0.02
verbose	100
loss function	RMSE
eval metric	RMSE
early stopping rounds	100
random seed	2019
bootstrap type	Poisson
devices	3

4.3.3 模型结果

模型的评价指标使用均方根误差 (RMSE : Root mean squared error)，其计算公式如下：

$$\mathrm{RMSE}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(P^{(i)}-\hat{P}^{(i)}\right)^2}$$

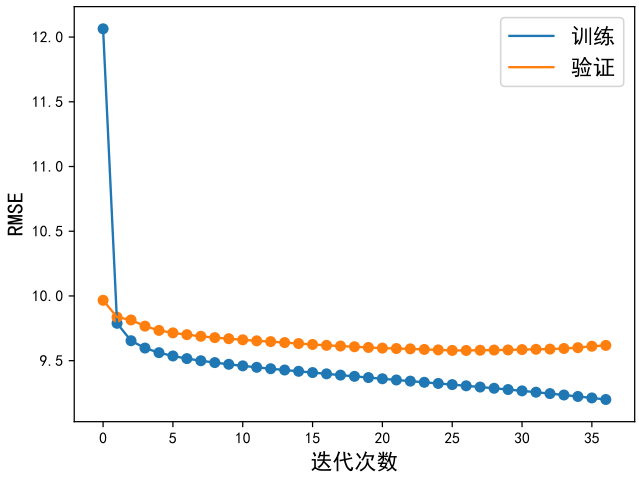

图 9　神经网络模型本地训练及验证结果

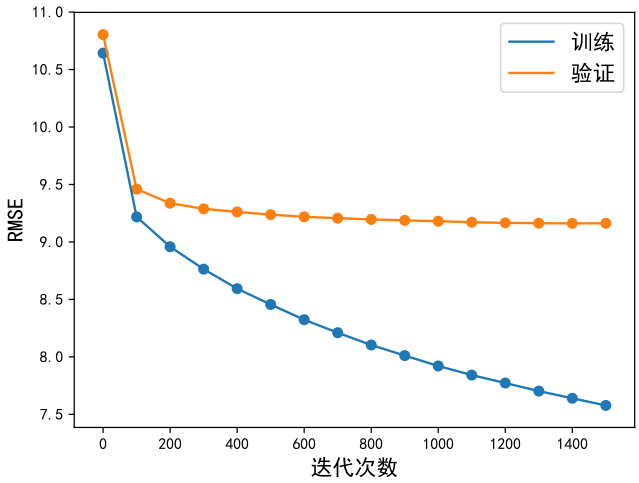

图 10　Catboost 模型本地训练及验证结果

本节对模型的本地训练验证结果和模型的线上结果进行阐述。针对上述模型设计部分提出的不同模型，我们使用相同的特征数据进行模型建立，以便于不同模型间的对比分析。图9和10分别给出上述神经网络模型和集成决策树模型的训练和本地验证的RMSE 指标结果。其中神经网络的本地验证 RMSE 分数为 9.57，梯度提升决策树的本地验证 RMSE 分数为 9.35。

将两个模型集成构建得到的组合预测模型，上线部署后预测的 RMSE 分数为 9.07421，截至提交论文时线上排行榜排名第一位。

5. 模型评价

5.1 模型的优点

问题一设计特征时，在经验模型的基础上进一步引入专家知识，将数学模拟和物理客观相结合，建立了简化的发射站和接收点的物理位置模型。然后基于此模型设计出接收点相对于发射点的 2D、3D 距离，接收点发射点连线与信号线之间夹角、最坏阻挡率等重要特征。此外，通过从模型拟合表征能力分析目前大部分机器学习方法缺少对样本频次分布建模能力的原因，构造其他合理的人为特征以解决该问题，最终通过数据探索对新构造特征的合理性进行可视化验证。

问题二评价特征时，针对特征发散度，使用了变异系数指标；针对特征与目标的相关性，使用了过滤式的统计评价方法包括皮尔森相关系数和最大信息系数，以及嵌入式的基于随机森林和 CatBoost 的特征评价方式。用这 5 种指标对所有特征都分别进行了打分以及排名，然后将 5 种指标的打分加权求和来得到每个特征的综合排名。对特征的评价方式比较多样化，能够尽可能地捕捉特征与目标值的相关性。此外，最后得到的特征综合排名结果有效地验证了问题一中精心设计的各种特征的合理性和重要性。

问题三建立模型时，首先建立多层前馈神经网络的预测模型，同时并行叠加 LR 模型以有效捕捉较为低阶的交叉特征。前馈神经网络的优点是具有较强的自学习自组织和泛化能力，能够有效拟合本题所需要构建的预测模型中复杂的非线性映射关系，但随着模型复杂度的提高，神经网络模型的参数增多，调优复杂且模型可解释性降低，同时更容易发生过拟合。而梯度提升决策模型具有可解释性强、数据适应性高的特点并可以通过决策树的串行集成为模型稳定性提供有效保证。因此，本问题最终使用改进的神经网络结构与梯度提升决策树集成构建组合模型，从而有效地结合不同模型的优势，获得较高的线上和线下评测指标得分。

5.2 模型的缺点

问题一设计特征时，我们手工设计的部分特征由于捕捉了复杂的特征函数关系，例如遮挡特征，计算较为耗时，对数据处理和模型建立阶段的效率带来了一定程度的影响。

问题二评价特征时，考虑到一千两百万的数据样本规模没有使用计算耗时较大的包裹式特征评价方法。使用 one hot 编码方式的类型特征在训练时转换为多个子特征列，对这类特征的评分目前是基于其中权重最大的子特征列来简化考虑的，会造成一定程度的精度损失。

问题三建立模型时，我们充分结合了神经网络模型和决策树模型的不同优势，并对

类别特征的部分低阶交叉特征进行显式建模，但由于比赛时间的限制，我们未能在网络整体架构和参数的优化上进行更为充分的实验和改进，之后可以继续对特征进行聚类等分析挖掘更多特征交叉间的信息，以便对交叉特征进行进一步探索和模型捕捉。

参考文献

[1] 党俊肖, 武丽梅, 马金辉, 基于 GIS 的移动通信信号传播预测及可视化仿真研究, 无线通信技术, 24(04):1-6+12, 2015.

[2] 周志华, 机器学习, Qing hua da xue chu ban she, 2016.

[3] Zhou N, Wang L, A modified T-test feature selection method and its application on the HapMap genotype data, Genomics, proteomics & bioinformatics, 5(3):242-249, 2007.

[4] Liu H, Setiono R, Chi2: Feature selection and discretization of numeric attributes, Tools with artificial intelligence, 1995. proceedings., seventh international conference on, IEEE, 388-391, 1995.

[5] Lin C, Miller T, Dligach D, et al., Maximal information coefficient for feature selection for clinical document classification, ICML Workshop on Machine Learning for Clinical Data. Edingburgh, UK, 2012.

[6] Oreski S, Oreski G, Genetic algorithm-based heuristic for feature selection in credit risk assessment, Expert systems with applications, 41(4):2052-2064, 2014.

[7] Jin C, Jin S W, Qin L N, Attribute selection method based on a hybrid BPNN and PSO algorithms, Applied Soft Computing, 12(8):2147-2155, 2012.

[8] Coefficient of Variation, New York, NY: Springer New York, 95-96, 2008.

[9] Reshef D N, Reshef Y A, Finucane H K, et al., Detecting novel associations in large data sets, science, 334(6062):1518-1524, 2011.

[10] Ge R, Zhou M, Luo Y, et al., McTwo: a two-step feature selection algorithm based on maximal information coefficient, BMC bioinformatics, 17(1):142, 2016.

[11] Breiman L, Random forests, Machine learning, 45(1):5-32, 2001.

[12] Dorogush A V, Gulin A, Gusev G, et al., Fighting biases with dynamic boosting, CoRR, abs/1706.09516, 2017.

[13] McCulloch W S, Pitts W, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics, 5(4):115-133, 1943.

[14] Rosenblatt F, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review, 65(6):386, 1958.

[15] Rumelhart D E, Hinton G E, Williams R J, Learning internal representations by error propagation, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[16] LeCun Y, Boser B, Denker J S, et al., Backpropagation applied to handwritten zip code recognition, Neural computation, 1(4):541-551, 1989.

[17] Simonyan K, Zisserman A, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs], 2014.

[18] Szegedy C, Wei Liu, Yangqing Jia, et al., Going deeper with convolutions, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA: IEEE, 1-9, 2015.

[19] He K, Zhang X, Ren S, et al., Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 770-778, 2016.

[20] Hornik K, Approximation Capabilities of Muitilayer Feedforward Networks. :7.

[21] Cheng H T, Koc L, Harmsen J, et al., Wide & Deep Learning for Recommender Systems, arXiv:1606.07792 [cs, stat], 2016.

[22] Wang R, Fu B, Fu G, et al., Deep & Cross Network for Ad Click Predictions, arXiv:1708.05123 [cs, stat], 2017.

附录 A PYTHON 源程序

特征计算关键代码：

```
import pandas as pd
import numpy as np
import math

file_path = './train_set'
eps = 1.

def parse_file(file_name, train_flag=True, save_h5py=True):
    data = pd.read_csv(os.path.join(file_name))
    del data['Cell Index']

    x_min = min(data['X'].min(), data['Cell X'][0])
    y_min = min(data['Y'].min(), data['Cell Y'][0])
    data['X'] -= x_min; data['Cell X'] -= x_min;
    data['Y'] -= y_min; data['Cell Y'] -= y_min;

    cell_x, cell_y = data['Cell X'][0], data['Cell Y'][0]

    data['abs_height'] = data['Building Height'] + data['Altitude']
    data['cell_abs_height'] = data['Height'] + data['Cell Building Height'] + data['Cell Altitude']
    cell_height = data['cell_abs_height'][0]
    data['delta_height'] = data['abs_height'] - data['cell_abs_height']

    # log height
    data['abs_height_log'] = np.log10(data['abs_height'] + eps)
    data['cell_abs_height_log'] = np.log10(data['cell_abs_height'])

    # angle to pie
    data['Electrical Downtilt'] = data['Electrical Downtilt'] / 180 * np.pi
    data['Mechanical Downtilt'] = data['Mechanical Downtilt'] / 180 * np.pi
    data['Azimuth'] = data['Azimuth'] / 180 * np.pi

    data['downtilt'] = data['Electrical Downtilt'] + data['Mechanical Downtilt']
    data['downtilt_pi_cos'] = np.cos(data['downtilt'])
    data['downtilt_pi_sin'] = np.sin(data['downtilt'])

    data['elec_down_cos'] = np.cos(data['Electrical Downtilt'])
    data['elec_down_sin'] = np.sin(data['Electrical Downtilt'])

    data['mech_down_cos'] = np.cos(data['Mechanical Downtilt'])
```

```
data['mech_down_sin'] = np.sin(data['Mechanical Downtilt'])

data['azimuth_cos'] = np.cos(data['Azimuth'])
data['azimuth_sin'] = np.sin(data['Azimuth'])

# distance
data['dis_3d'] = np.sqrt(
(data['X'] - cell_x) ** 2 + (data['Y'] - cell_y) ** 2 + (data['abs_height'] - cell_height) ** 2
)
data['dis_2d'] = np.sqrt(
(data['X'] - cell_x) ** 2 + (data['Y'] - cell_y) ** 2
)

# log distance
data['dis_3d_log'] = np.log10(data['dis_3d'] + eps)
data['dis_2d_log'] = np.log10(data['dis_2d'] + eps)

# log frequency
data['frequency_band_log'] = np.log10(data['Frequency Band'])

cell_emission_angel = (2.0*np.pi - data['Azimuth']) + np.pi/2.0
p1_0, p1_1 = np.cos(cell_emission_angel), np.sin(cell_emission_angel)
p2_0 = data['X'] - data['Cell X']
p2_1 = data['Y'] - data['Cell Y']
data['hori_belta'] = (p1_0 * p2_0 + p1_1 * p2_1) / np.sqrt(p2_0 ** 2 + p2_1 ** 2)
data['hori_belta'] = data['hori_belta'].fillna(1)

# belta
p1_0, p1_1, p1_2 = p1_0, p1_1, -1 * np.tan(data['downtilt'])
p2_0, p2_1, p2_2 = p2_0, p2_1, data['delta_height']
data['beta'] = (p1_0 * p2_0 + p1_1 * p2_1 + p1_2 * p2_2) / np.sqrt(p1_0 ** 2 + p1_1 ** 2 + p1_2 ** 2) /
    np.sqrt(p2_0 ** 2 + p2_1 ** 2 + p2_2 ** 2)
data['beta'] = data['beta'].fillna(1)

#delta x, delta y
data['delta_x'] = p2_0
data['delta_y'] = p2_1

# obstacle
obstacle_num = []
worst_obstacle_height = []
worst_obstacle_height_rate = []
worst_obstacle_delta_d = []
worst_obstacle_type = []
```

```
    worst_obstacle_x, worst_obstacle_y = [], []
    cell_x, cell_y, cell_height = data['Cell X'][0], data['Cell Y'][0], data['cell_abs_height'][0]
    for j in range(len(data)):
        x_0, y_0 = data['X'][j], data['Y'][j]
        abs_height_0 = data['abs_height'][j]
        dx_0, dy_0 = x_0 - cell_x, y_0 - cell_y

        x_1, y_1 = data['X'], data['Y']
        dx_1, dy_1 = x_1 - cell_x, y_1 - cell_y

        the_cos = (dx_0 * dx_1 + dy_0 * dy_1) / np.sqrt(dx_0 ** 2 + dy_0 ** 2) / np.sqrt(dx_1 ** 2 + dy_1 **
            2)
        cos_simi = (the_cos >= 0.99)

        between = ((x_1 - x_0) * (x_1 - cell_x)) <= 0
        height = np.sqrt(dx_1 ** 2 + dy_1 ** 2) / np.sqrt(dx_0 ** 2 + dy_0 ** 2) * (abs_height_0 -
            cell_height) + cell_height
        height = height.fillna(cell_height)

        too_height = data['abs_height'] >= height
        too_height_rate = (data['abs_height'] - height) / height

        num = np.sum(cos_simi & between & too_height)
        if num == 0:
            num = 1
            heightest = j
        else:
            heightest = too_height_rate.loc[cos_simi & between & too_height].argmax()
            obstacle_num.append(num)
    worst_obstacle_height.append(data['abs_height'][heightest])
    worst_obstacle_height_rate.append(too_height_rate[heightest])
    worst_obstacle_delta_d.append(np.sqrt(dx_1 ** 2 + dy_1 ** 2)[heightest])
    worst_obstacle_type.append(data['Clutter Index'][heightest])
    worst_obstacle_x.append(x_1[heightest])
    worst_obstacle_y.append(y_1[heightest])
    data['obstacle_num'] = np.array(obstacle_num)
    data['worst_obstacle_height'] = np.array(height)
    data['worst_obstacle_height_rate'] = np.array(worst_obstacle_height_rate)
    data['worst_obstacle_delta_d'] = np.array(worst_obstacle_delta_d)
    data['worst_obstacle_type'] = np.array(worst_obstacle_type)
    data['worst_obstacle_x'] = np.array(worst_obstacle_x)
    data['worst_obstacle_y'] = np.array(worst_obstacle_y)

    for j in range(1, 21):
```

```
        if j in [1, 3, 4, 9, 19, 20]:
            continue
        data['Clutter Index_{}'.format(j)] = (data['Clutter Index'] == j).astype(int)
        del data['Clutter Index']

    for j in range(1, 21):
        if j in [1, 3, 4, 9, 19, 20]:
            continue
        data['index_{}_sum'.format(j)] = data['Clutter Index_{}'.format(j)].sum()

    data = data.fillna(0)
    data[data==float('inf')] = 0

    if save_h5py:
        h5 = pd.HDFStore('/home1/xie/shuxue/' + file_name.split('/')[-1][:-3] + 'h5', 'w')
        h5['data'] = data
        h5.close()

    if train_flag:
        target = data['RSRP']
        data = data.drop(['RSRP'], axis=1)
        return data, target
    else:
        return data
```

特征评价关键代码：

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# CV ranking
cv_feature_impotance = [(column, np.std(data[column])/abs(np.mean(data[column]))) for column in features]
cv_feature_impotance.sort(key=lambda x: x[1], reverse=True)
cv_sorted_attrs = [item[0] for item in cv_feature_impotance]
cv_ranking = [cv_sorted_attrs.index(x) + 1 for x in features]
print("# CV result")
print(cv_ranking)
print(cv_feature_impotance)

# Pearson ranking
from scipy.stats import pearsonr
```

```
y_values = data[target]
pearsonr_feature_impotance = [(column, pearsonr(data[column], y_values)[0]) for column in features]
pearsonr_feature_impotance.sort(key=lambda x: abs(x[1]), reverse=True)
pearson_sorted_attrs = [item[0] for item in pearsonr_feature_impotance]
pearson_ranking = [pearson_sorted_attrs.index(x) + 1 for x in features]
print("# Pearson result")
print(pearson_ranking)
print(pearsonr_feature_impotance)

# RF (Random Forest) feature ranking
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=500, n_jobs=-1)
rf.fit(data.ix[:, features], y_values)
rf_scores = rf.feature_importances_
print(rf_scores)
rf_result = sorted(list(zip(features, rf_scores)), key=operator.itemgetter(1), reverse=True)
rf_sorted_attrs = [x[0] for x in rf_result]
rf_ranking = [rf_sorted_attrs.index(x) + 1 for x in features]
print("# Random forest result")
print(rf_sorted_attrs)
print(rf_ranking)

# MIC (Maximal Information Coefficient) feature ranking
from minepy import MINE

mine = MINE()
mic_scores = []
data_s = data.sample(frac=0.001)
print(data_s.shape)
for attr in features:
print(attr)
mine.compute_score(data_s.loc[:, attr], y_values)
mic_scores.append(mine.mic())
mic_result = sorted(list(zip(features, mic_scores)), key=operator.itemgetter(1), reverse=True)
mic_sorted_attrs = [x[0] for x in mic_result]
mic_ranking = [mic_sorted_attrs.index(x) + 1 for x in features]
print("# MIC result")
print(mic_sorted_attrs)
print(mic_ranking)

# Catboost feature ranking
import catboost as cat
crf = cat.CatBoostRegressor(
```

```
 iterations =10000,
depth=10,
 learning_rate =0.02,
verbose=100,
 loss_function ='RMSE',
eval_metric='RMSE',
 early_stopping_rounds=100,
random_seed=2019,
 task_type='GPU',
 bootstrap_type='Poisson',
devices='3'
)
crf.fit(train_data, eval_set=eval_data)
cat_ranking = [(y, x) for x, y in zip(data.columns, crf.feature_importances_)]
cat_ranking.sort()
print(cat_ranking)
```

模型训练关键代码：

```
import tensorflow as tf
import pandas as pd
import numpy as np

import util
import tf

def calRootMse(pre, label):
    pre = pre.reshape(-1)
    label = label.reshape(-1)
    return np.sqrt(np.mean((pre - label) ** 2))

def calPcrr(y_pred, y_true):
    y_pred = y_pred.reshape(-1)
    y_true = y_true.reshape(-1)
    t = -103
    tp = len(y_true[(y_true < t)&(y_pred < t)])
    fp = len(y_true[(y_true >= t)&(y_pred < t)])
    fn = len(y_true[(y_true < t) & (y_pred >= t)])
    try:
        precision =tp/(tp+fp)
        recall = tp/(tp+fn)
        pcrr = 2 * (precision * recall)/(precision + recall)
    except:
```

```
        return 0
    return pcrr

def linear_layer(inputs, output_dims):
    x = tf.layers.dense(inputs, output_dims, activation=tf.nn.relu,
            kernel_initializer=tf.contrib.layers.xavier_initializer())
    return x

def network(x):
    y1 = linear_layer(x, 2048)
    y1 = linear_layer(y1, 1024)
    y1 = linear_layer(y1, 256)

    y2 = tf.layers.dense(x, 256)
    y = tf.concat((y1, y2), axis=0)
    y = tf.layers.dense(y, 1)
    return y

train_data_np = np.array(train_data)
train_target_np = np.array(train_target).reshape(-1, 1)
eval_data_np = np.array(eval_data)
eval_target_np = np.array(eval_target).reshape(-1, 1)

train_size = len(train_target_np)
eval_size = len(eval_target_np)
input_dims = train_data_np.shape[-1]

batch_size = 1280
epoch_num = 10000

train_size = train_size // batch_size * batch_size
eval_size = eval_size // batch_size * batch_size
train_data_np = train_data_np[:train_size]
train_target_np = train_target_np[:train_size]
eval_data_np = eval_data_np[:eval_size]
eval_target_np = eval_target_np[:eval_size]
print('train_size', train_size, 'eval_size', eval_size)
input_pl = tf.placeholder(tf.float32, (None, input_dims))
label_pl = tf.placeholder(tf.float32, (None, 1))
outputs = network(input_pl)

global_step = tf.Variable(0, trainable=False)
starter_learning_rate = 2e-5
learning_rate = tf.train.exponential_decay(starter_learning_rate, global_step,
```

```
10, 0.8, staircase =True)
loss = tf.losses.mean_squared_error(labels=label_pl, predictions=outputs)
optimizer = tf.train.MomentumOptimizer(learning_rate=learning_rate, momentum=0.8)
train_op = optimizer.minimize(loss)

init = tf. global_variables_initializer ()
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
sess = tf.Session(config=config)

sess.run(init)
best_epoch_idx, best_loss = 0, 1e10
train_losses, eval_losses = [], []

for epoch_index in range(epoch_num):
    print('Runing')
    print('lf', sess.run(learning_rate, {global_step: epoch_index}))
    total_loss = 0
    for i in range(0, train_size, batch_size):
        datas, labels = train_data_np[i:i+batch_size, ...], train_target_np[i:i+batch_size, ...]
        los, _, out = sess.run([loss, train_op, outputs], feed_dict={input_pl: datas, label_pl: labels})
        total_loss += los / (train_size / batch_size)
    print('train epoch {}: {}'.format(epoch_index, total_loss))
    train_losses.append(total_loss)
    total_loss = 0
    result = []
    for i in range(0, eval_size, batch_size):
        datas, labels = eval_data_np[i:i+batch_size, ...], eval_target_np[i:i+batch_size, ...]
        los, out = sess.run([loss, outputs], feed_dict={input_pl: datas, label_pl: labels})
        total_loss += los / (eval_size / batch_size)
        result.append(np.array(out))
    result = np.concatenate(result, axis=0)
    print('eval epoch {}: {}, rmse: {}, pcrr: {}'.format(epoch_index, total_loss,
    calRootMse(result, eval_target_np), calPcrr(result, eval_target_np)))
    eval_losses.append(total_loss)

    if total_loss <= best_loss:
        best_loss, best_epoch_idx = total_loss, epoch_index
        tf.saved_model.simple_save(sess, model_save_path+'{}'.format(best_loss),
        inputs={"myInput": input_pl }, outputs={"myOutput": outputs})
    if epoch_index - best_epoch_idx >= 10:
        break
```