机器学习导论期末大作业报告

丁豪 李惟康 王宸旭
181220010 181220031 181220056
ETADingLiWang3EasyPieces
南京大学人工智能学院

2020 年 6 月 19 日

摘　　要

本文使用现成的lightGBM方法，基于海运中船只的 GPS-AIS 等原始数据清洗、加工后产生的特征，对船只预计到港时间 ETA(estimated time of arrival) 进行了预测。这种方法是 XGBoost 的优化，通过优化桶排序、结点分裂以及添加直方图算法等途径使得训练复杂度 $\mathcal{O}(\#\text{data} * \#\text{feature})$ 降至 $\mathcal{O}(k * \#\text{feature})$，其中 k 是一个常数。

我们使用本次比赛主办方提供的提交评测机制进行测试。最终结果为使用一千万加工过的训练集，将包含 20000 条订单共计 45457 条测试数据的 MSE 误差降至 13523(小时^2)，平均每条订单的误差在 100 小时左右。优于直接调用XGBoost及lightGBM包堆砌数据集取得的结果。

关键词： Port ETA，lightGBM，数据清洗

1　介绍 (Introduction)

在企业全球化的业务体系种，海运物流是其最重要的一项支撑。通常，船运公司会和数据供应公司进行合作，对运输用的船通过 GPS 进行定位以监控船的位置。在运输管理的过程中，货物到达目的港的时间是非常重要的数据，知悉较为准确的货物到达时间关系到港口当局、港口经营者、燃料供应商、船舶经营者、物流公司、货物买卖方等多方的利益。所以基于船运的历史数据构建模型，对目的港到达时间进行预测，是一个热门的研究方向。预测时间简称为 ETA(estimated time of arrival)。

基于历史数据预测 ETA 是一个回归任务，可以用传统的统计学方法以及现有的诸多支持回归任务的人工智能模型解决。而根据不同的航行数据，又有不同的适合的方法。针对 GPS 数据，有许多方法已经被使用，如Tay 等人[1] 使用的马尔可夫模型尝试预测船只轨迹和 ETA;Chun-Hsin等人 [2] 使用支持向量回归对 ETA 进行预测; Jiwon[3] 使用的 KNN 模型对 ETA 进行预测。然而，上述方法只能在特定的场景和数据集上表现出较好的性能，场景更换后准确率明显下降。并且，Oleh 的研究工作中[4] 提到，对 ETA 的预测虽然可以处理为分类问题，但是效果远没有处理为回归问题精确，上述模型中的部分并不是典型的适用于回归任务的模型。

对于上述问题，Bodunov 和 Oleh 等人[4] 使用了集成学习的方法，尝试使用若干种树分类器和前向神经网络作为基分类器，显著提升了预测的准确率。但是，将原理差异较大的基分类器

应用在集成学习当中，使得整体的可解释性降低，例如如何给基分类器分配权重、基分类器发挥的作用是优势互补还是相互抵消均无法说清。

综上，我们仍采用研究中证实性能较好的集成学习方法，但是我们的基分类器使用相同的模型，即梯度提升树 (GBDT)。出于训练时间考虑，我们采用轻量级的高效梯度提升树 (lightGBM)，以期更好地解释基分类器对集成学习整体的影响。通过多次数据清洗、交叉验证以及调整参数实验，结果表明使用相同的基分类器模型可以较好地解释各个参数对整体性能的影响 (参见数值分析部分)。

此外，Mestl 等人[5] 利用目标港口和 GPS 数据两个信息采用传统的机器学习方法取得了比较高的准确率。但是他们的模型比较依赖于船舶的类型，对于班轮 (liner), 即航线固定的船舶，模型预测精度很高；而对于流浪型轮船 (tramper)，模型的表现就很差。这表明他们过度依赖了目标港口的数据。而目标港口数据天然存在缺失或者录入等问题。所以为了借鉴 Mestl 的思路我们使用地理相关 API 库对港口名称、地址进行了清洗，统一化、标准化港口数据，加入训练特征当中，使误差从 76000 降至 32000。

本报告结构安排如下:

第 2—8 部分是模型、方法、算法的分析以及介绍，第 9—10 部分是结果分析与结论展示，随后的部分是致谢、参考文献、附录以及小组分工说明。

2 模型 (Model)

建立模型需要考虑特征的选择和表示形式，进而包含了数据清洗步骤。所以以下内容先介绍我们进行在数据清洗方面进行的尝试，然后再得出特征选择的结论，最后构建模型。

2.1 原始数据

在本次 ETA 比赛中，训练数据由 train、event、port 三个数据集构成，分别对应了历史 GPS 数据、历史港口事件以及港口信息，而测试数据由 test 数据集构成，以下分别针对每一个数据集进行分析。

2.1.1 历史运单数据

历史运单数据的基本单元是一条 “记录”，他含有下述 13 列信息。这些记录按照各自 timestamp 的时间顺序由前往后排列，因而各个航程的记录往往会穿插在一起。同时 speed 字段之后的 6 列为人工录入，经过检验存在大量的缺失情况。

- loadingOrder：具有唯一性的运单号，为 14 位字符串类型
- carrrierName：承运商名称，不定长字符串类型
- timestamp：yyyy-MM-dd’T’HH:mm:ss.SSSZ 格式的记录时间戳，分布在 2019 年全年到 2020 年初
- longitude：所在地经度，范围为 $(-180, 180]$，6 位浮点数
- latitude：所在地经度，范围为 $[-90, 90]$，6 位浮点数

- vesselMMSI：具有唯一性的船只识别码，11 位字符串
- speed：瞬时速度，单位为 km/h，整数
- direction：量化的船只朝向，范围 [0,36000)，正北方向为 0 度，整数类型
- vesselNextport：船舶将要到达的下一港口，字符串类型
- vesselNextportETA：船运公司给出的到“下一个港口”预计到达时间，格式为：yyyy-MM-dd’T’HH:mm:ss.SSSZ
- vesselStatus：当前船舶航行状态，主要有 moored、under way using engine、not under command、at anchor、under way sailing、constrained by her draught，字符串类型
- vesselDatasource：船舶数据来源，字符串类型
- TRANSPORT_TRACE：船的路由，由"-" 连接组成，连接数个港口，字符串类型

2.1.2 港口事件数据

港口事件数据每一行有以下 4 列数据，为上述运单数据信息的补充。经主办方讲解赛题时提示，本数据集为纯人工录入，因而存在一些不准确的现象，但是如果想要提升性能确实可以使用他作为辅助。

- loadingOrder：与上文 loadingOrder 等价
- EVENT_CODE：时间编码，主要有"TRANSIT PORT ATD 实际离开中转港"、"SHIPMENT ONBOARD DATE 实际离开起运港"、"TRANSIT PORT ATA" 实际到达中转港”、“ARRIVAL AT PORT 实际到达目的港”这么 4 中情况，数据类型为字符串
- EVENT_LOCATION_ID：港口名称，与历史运单数据中 TRANSPORT_TRACE 字段每一个港口等价
- EVENT_CONVOLUTION_DATE：yyyy/MM/dd HH:mm:ss 格式的时间戳信息

2.1.3 港口数据

- TRANS_NODE_NAME：与上文 EVENT_LOCATION_ID 等价，为港口名称字符串
- LONGITUDE：与上文 longitude 相同
- LATITUDE：与上文 latitude 相同
- COUNTRY：国家，字符串
- STATE：州/省，字符串
- CITY：城市：字符串
- REGION：县 | 区，字符串
- ADDRESS：详细地址，字符串
- PORT_CODE：港口编码，字符串 TIM 截图 20200619114558

2.1.4 测试数据

- loadingOrder、timestamp、longitude、latitude、speed、direction、carrierName、vesselMMSI、TRANSPORT_TRACE：与历史运单数据中的定义完全相同

- onboardDate：离开起运港时间，格式为：yyyy/MM/dd HH:mm:ss

2.2 数据选用与特征工程

2.2.1 基础数据特征选取

对比 train 与 test 两个数据集的各列，我们可以发现 loadingOrder 是两表的" 键"。timestamp、longitude、latitude、speed、direction 是公共的数值属性，比较容易直接作为特征学习。而 carrierName、vesselMMSI、TRANSPORT_TRACE 虽然也是公共属性，但是格式为字符串，如果要加以学习需要进行哑编码等操作，较为繁琐。因此第一批特征提取时我们选取的对象就是 timestamp、longitude、latitude、speed、direction 这 5 个数值属性。

测试数据集的记录源自整条航行的随机采样，对于时间先后没有明显的偏好性，因此我们在测试集中使用每一条航程的最末 timestamp 减去最初 timestamp 作为 ETA 的增加量，作为训练集的训练目标值，即：

$$\mathrm{ETA}_{\mathrm{increase}} = \mathrm{timestamp}_n - \mathrm{timestamp}_0$$

对于其余四个属性，分别提取其最大、最小、平均、中位数 4 个属性，最终扩展为 16 个特征值，即：

$$\mathrm{Features} = \{\mathrm{longitude}_{\mathrm{max}}, \mathrm{longitude}_{\mathrm{min}}, \mathrm{longitude}_{\mathrm{mean}}, \mathrm{longitude}_{\mathrm{median}}, \ldots, \mathrm{direction}_{\mathrm{median}}\}$$

这样一来我们对实际训练目标进行了一次转换，不是直接训练 ETA，而是根据已有数据训练 ETA 的增量，这样对测试集只需要给 onboardDate 增加此回归方法预测的增加量就可以的到最终 ETA，即：

$$\mathrm{ETA} = \mathrm{onboardDate} + \mathrm{ETA}_{\mathrm{increase}}$$

2.2.2 港口信息加入

为了进一步提升性能，决定尝试引入更多有意义的数值类型特征值。通过检查数据集我们发现，虽然 train 中 TRANSPORT_TRACE 经常缺失，但在 test 中却始终是完整的，而且通过查询港口信息，TRANSPORT_TRACE 又可以衍生出起点、终点的坐标信息。于是我们决定认为补全训练集的起点、终点信息，依次来和训练集相对应，在这里采用的方法十分简单，直接将每个运单第一条记录的坐标作为起点坐标，最后一条记录的坐标作为终点坐标，即：

$$\mathrm{Cordinate}(x)_{\mathrm{begin}} = \begin{cases} \mathrm{latitude}_0, \mathrm{longitude}_0 & x \in \mathrm{train} \\ \mathrm{latitude}(\mathrm{port}_0), \mathrm{longitude}(\mathrm{port}_0) & x \in \mathrm{test} \end{cases} \tag{1}$$

$$\mathrm{Cordinate}(x)_{\mathrm{end}} = \begin{cases} \mathrm{latitude}_n, \mathrm{longitude}_n & x \in \mathrm{train} \\ \mathrm{latitude}(\mathrm{port}_n), \mathrm{longitude}(\mathrm{port}_n) & x \in \mathrm{test} \end{cases} \tag{2}$$

2.2.3 事件信息加入

经过观察发现, 一开始并未使用的事件信息数据集中, 含有部分订单再 GPS 运单信息已经结束后的其余记录, 因此将他用上作为补充。由于事件信息中含港口名称字段, 通过查询港口信息表, 可以获得对应的经纬度数据。对于缺失值的情况, 仍旧使用 3.2.2 中的信息作为起点和终点坐标, 即:

$$\text{Cordinate}(x) = \begin{cases} \text{latitude}_0, \text{longitude}_0 & x \in \text{train}, x \notin \text{port} \\ \text{latitude}(\text{port}_x), \text{longitude}(\text{port}_x) & x \in \text{train}, x \in \text{port} \\ \text{latitude}(\text{port}_0), \text{longitude}(\text{port}_0) & x \in \text{test} \end{cases} \tag{3}$$

2.3 额外特征值的加入

最后我们分析了船航行中可能存在的”抛锚“停止情况, 可能对整个航程的整体时间产生较大影响, 因而将人工判断的”抛锚“属性额外加入特征值列表中, 体实现参考代码部分, 于是最终的特征值:

$$\text{Features}^* = \text{Features} \cup \{\text{latitude}_{\text{begin}}, \text{longitude}_{\text{begin}}, \text{latitude}_{\text{end}}, \text{longitude}_{\text{end}}, \text{anchor}\}$$

我的实验即通过 LGB 模型来从训练集的数值特征 Features^* 中学习其到 $\text{ETA}_{\text{increase}}$ 的映射关系, 并将它应用在训练集的预测问题上。

3 基本理论和方法 (Basic Theory and Method)

我们所使用的lightGBM是一种集成学习方法。这个方法可以被抽象概况为:

$$\sum_{i=1}^{M} \gamma_i T_i(\alpha\beta)$$

其中 $T_i(\alpha\beta)$ 是第 i 个基分类器在特征 α 和数据 β 上的输出值, γ_i 是第 i 个基分类器对整体预测结果的影响权重。

此外，在将数据投入训练以及测试之前，我们进行了诸多有关不良数据清洗、特征选择的尝试，具体已在模型部分中描述。

4 方法 (Method)

由于能力有限，本次作业中我们使用的是已有的方法。我们先使用梯度提升树 (GBDT)[6][7] 及其改进极端梯度提升树 (XGBoost)，但是XGBoost的训练开销太大，最终我们选择的是轻量级的极端梯度提升树 (lightGBM) 进行预测。下面按照我们的学习、尝试过程进行分析介绍。

4.1 梯度提升树 (GBDT)

下面先给出梯度提升树的在回归任务中的伪代码:

第一步是输入数据集以及给定一个**可导**的损失函数。数据集中的 x_i 表示我们特征工程中选取的特征，例如船舶速度 (Speed)、航行方向 (Direction) 二者数值化以后的最小、最大、均值，船

Algorithm 1 GBDT Algorithm

1: **Input:** Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$;
2: **Step 1:** Initialize model with a constant value: $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(\gamma_i, y)$
3: **for** $m = 1$ to M **do**
4: **(A)** Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for i = 1 to n
5: **(B)** Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for j = 1 to J_m
6: **(C)** For j = 1 to J_m, compute $\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{ij}} L(\gamma, F_{m-1}(x_i) + \gamma)$
7: **(D)** Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}(x \in R_{jm})$
8: **end for**

舶 GPS 数据，格式化以后的时间戳 (TimeStamp) 以及预计的目标港口 (VesselNextPort)。y_i 是船舶的实际到港时间 (ATA)。事实上，训练集的诸多订单数据中，船舶的到港时间存在缺失或者误差，所以我们在特征工程中进行了许多尝试来保证这一特征的有效性，在这里我们假设训练集中的每一个 x_i 都有其对应的 y_i。我们使用 MSE 损失函数:

$$\frac{1}{2}(\text{Observed ATA} - \text{Predicted ETA})^2$$

该损失函数的好处在于，其使用链式法则对预测值 ETA 求导后，得到的导数是 $-$(Observed ATA$-$ Predicted ETA)。这个式子表明负梯度 $-\nabla$ 就是残差 r，有了这个关联，后续步骤中我们就可以不直接对 ETA 进行回归预测，而是对残差 r 进行回归分析，这样我们的回归过程就和梯度提升联系了起来。

算法的第二行是 GBDT 模型的初始化，该步从形式上看是寻找一个使得损失函数最小化的 ETAγ。使用 MSE 损失函数的好处在这里进一步体现：对 $F_0(x)$ 使用链式法则进行求导及移项处理以后，我们给模型的初始值实际上就是所有到港时间 y_i 的平均值，即 $\frac{1}{n}\sum_{i=1}^{n} y_i$.

随后算法进入了一个循环，其中索引 m 表示当前树的序号，M 表示一共需要使用的树的数量。通过查阅 GBDT 文档我们得知，实际工程中 M 通常大于等于 100，我们训练使用的 M 值即是 100。

步骤 (A) 实际上只是在计算损失函数对预测值的偏导，这点我们在上面分析过，实际上就是残差。然后把上一轮循环中输出的预测结果 $F_{m-1}(x)$ 代入 $F(x)$ 中。最后 i 从 1 到 n 的遍历即遍历每一个训练数据，都计算残差。还有一点值得说明的是，这一步里我们所求的偏导 (与梯度同理) 就是梯度提升树 GBDT 里"GBDT" 的来源。

步骤 (B) 所做的事情很简单：利用现有数据构造一个回归树，然后使用该树来预测每一个样本的残差 r_{im}，而非 ETA。然后给这棵刚创建的树标记所谓的"terminal regions R_{jm}"，这实际上就是回归树的叶子结点，第 m 棵树的第 j 个结点被标记为 R_{jm}。

步骤 (C) 同算法第二行类似，是在对每个叶子 R_{jm} 求使得损失函数最下化的 ETAγ_{jm}，不过这一步的 MSE 中添加了前一轮循环的预测结果 $F_{m-1}(x_i)$。由于选取 MSE 损失函数，利用链式法则进行简单的数学推导以后，我们惊奇地发现：每一步的 γ_{jm} 实际上就是所有分到 R_{jm} 里面的残差的平均值！这是选取 MSE 作为损失函数带来的又一巧合 (也许不巧，但是挺妙的)。

步骤 (D) 是在更新模型的输出 ETA 值，由公式可以得知，第 m 轮的输出 ETA 值 $F_m(x)$ 包含两个部分，一个是上一轮循环中的输出 ETA 值 $F_{m-1}(x)$，另一部分是本轮循环中我们对残差

的预测更新。系数 ν 即代表学习率。有关学习率 ν 的分析将在第 5 部分——方法分析中提到。

4.2 极端梯度提升树 (XGBoost)

由于方法存在诸多相似之处以及时间因素，在XGBoost以及lightGBM的分析当中，我们抛去 4.1 节中已经提到的重复内容，尽量挑选不同点及创新点来分析。

在XGBoost当中，每一棵树不再是一个简单的回归树，而是一种引入了相似度增益 (Gain)、正则项以及后剪枝等诸多考量的回归树。相对于直接预测 ETA，在XGBoost中我们仍在预测 4.1 节所提到的残差。基于样本之间的残差，引入一种相似度特征 (Similarity)，其计算公式如下:

$$\text{Similarity Score} = \frac{(\sum_{i=1}^{n} r_{im})^2}{n + \lambda}$$

即所有残差的和的平方，除以残差的个数与正则项 λ 的和。注意到这里是残差的和的平方，而非残差的平方和，这一设定使得偏离较远的残差可以相互抵消使得数值降低，偏离不远的残差相互累加使得数值升高，从而刻画出相似性。

类似基础的决策树划分准则，在定义了相似度以后，我们即可基于它再定义出子分支结点相对其父结点的相似度增益 (Gain)，进而使用 Gain 作为划分准则和剪枝原则。

关于最优划分点的选择还可以细分为贪心算法和近似算法，正则项的设置标准也有一定的判定准则，而论文和文档当中还提到诸多了最优化技巧，最终得到一个完善的XGBoost模型。受限于时间因素，我们对XGBoost模型的了解止步与此，未深入推导每一步数学公式，就此说明。

通过调包使用XGBoost替换梯度提升树 GBDT 使我们在华为云平台上的提交误差从26 万降至了 17 万。是我们的第一个阶段性胜利。但是由于训练时间过长 (粗略估算训练时间随数据集大小增长不是线性关系)，我们开始参考 baseline，使用lightGBM模型。

5 方法分析 (Analysis of Method)

5.1 ν 与 Bias——Variance 的关系

在调参的过程中，我们使用固定的训练集 (前 100 万训练集数据，已做过特征工程，包含人为添加的港口等新特征)，测试学习率对最终提交的 MSE 误差的影响。这个过程中我们发现，在 (0.3-0.01) 这个区间内，学习率 ν 越小，模型的泛化性能越好，即在测试集上的误差越小。这与在 XGBoost 和 GBDT 上的调参情况相同。这一点只作为经验规律在 lightGBM 的文档中被提到，并没有给出解释。我想根据自己在前面数学推导中的理解提出一点猜想，可以与偏差-方差的关系起来理解：使用较小的学习率增大了单次迭代 ETA 与 ATA 之间的 Bias，即模型不能迅速的拟合训练数据，但是这样也避免了过拟合，从而使得最终模型在测试集上具有能具有较低的 Variance，即较好的性能。

6 算法 (Algorithm)

首先给出对于这一具体问题的解决方案流程图, 如下图1所示:

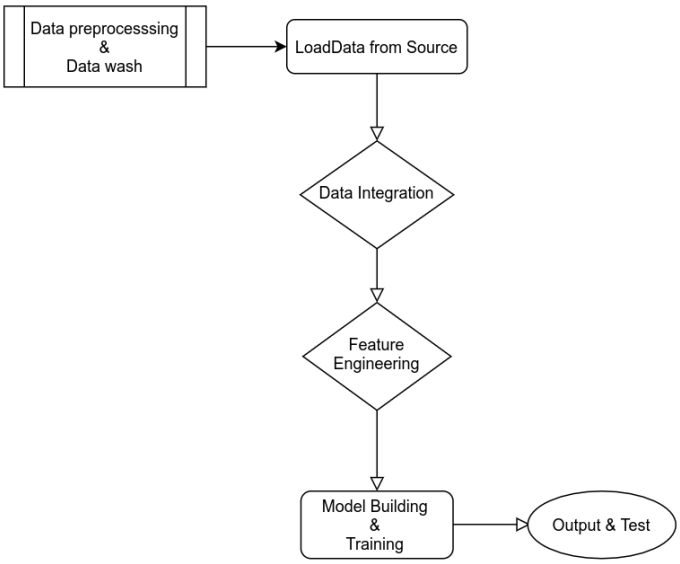

图 1: 流程图

下面仅对于较关键的数据预处理部分给出相应的算法伪代码:

Algorithm 2 Data preprocessing (test_begin_end)

Require: testData, port_info

Ensure: testData*

loadData from source
Function: trace2cordinate(trace):
    begin, end = trace.split
    Get GPS coordinates of begin, end.
EndFunction
Add coordinates of begin and end for each loadingorder.

Algorithm 3 Data preprocessing (event_feature)

Require: Order, port_info

Ensure: Order*

loadData from source
Function: port2cordinate(port):
    Get corresponding port's coordinates
    if len=1
        return coordinates
    else:
        return (200,200)
EndFunction
Remove rows with null values.
Add coordinates of EVENT_LOCATION for each loadingorder.

7 算法分析 (Analysis of Algorithm)

该部分主要从时间复杂度角度对现有算法进行分析, 经过数据预处理后, 不难发现主要的训练过程当中特征工程所占据的运行时间比例最高, 而特征工程当中经过逐行的分析之后发现是 `pandas.DataFrame.apply` 操作所占据的时间最多。这是由于通常 `apply` 操作不会利用矢量化。另外,`apply` 返回一个新的 Series 或 DataFrame 对象, 因此对于一个非常大的 DataFrame, 存在相当大的 IO 开销。从整体上来看, 这是由于 `pandas` 默认是在单个 CPU 核上，采用单进程执行函数，这在小数据集上运行得很好，因为可能觉察不到太多性能上的差异。但对于类似的大规模数据集时就成为了提高运行效率的瓶颈。

这里我们引入dask[8] 框架对原有的计算进行加速: Dask 是一个开源项目，提供 NumPy Arrays, Pandas Dataframes 和常规列表的抽象, 允许使用多核处理并行运行它们。这里需要注意的是, 显然并不是特征工程当中的所有操作均适合于转换为 `dask` 下的对应操作: 例如 `sort_values()`, 排序算法具有天生的不可并行性; 以及 `groupby()`, `pandas` 本身实现的 `groupby()` 效率已经很高; 由此看来可以将 `apply` 转化为 `dask` 框架下并行以提高效率, 大致修改如下:

```
df = dd.from_pandas(df, npartitions=NCORES)
df = df.apply(lambda x: ......)
df = df.compute()
```

注意 `npartitions` 数值最好与硬件环境的 CPU 核心数保持一致, 根据 StackOverflow 的说法，" 建议将 Dataframe 划分为与计算机所拥有的核心数量相同的分区，或者是该数量的几倍，因为每个分区将在不同的线程上运行，如果有的话，它们之间的通信将变得过于昂贵许多。"

修改之后经过本地测试 (500w 条), 在其他操作时间差距不大的情况下, 引入 `dask` 框架使得对于单个 `apply` 操作运行时间从 620s 降低至 120s, 提升约 5 倍左右的效率, 这一效应在云平台多核心的处理器以及大规模数据集上尤其明显。

8 数值结果 (Numerical Result)

本次实验的仿真环境: 针对 baseline 基础上的小规模数据集训练 (以训练选取训练集的行数为指标,1000w 行以下为小规模; 以上为大规模) 使用选手个人计算机进行, 列举本队中某选手的实验环境如下:

- CPU: Intel Core i7-9750H @ 12x 4.5GHz
- RAM: 16G
- KERNEL: x86_64 Linux 5.4.44-1
- OS: Manjaro 20.0.3 Lysia

对于大规模数据集的训练, 这里采用腾讯云智能钛机器学习平台进行训练, 配置信息如下:TI.8XLARGE64.32core64g。

首先, 注意到本次大赛所给的数据集存在着较大的缺陷, 数据缺失较多且含有一些重复以及不合理的值, 考虑首先对数据集进行预处理以及清洗:

1. 对于港口信息数据集 (port.csv), 注意到有较多的缺失值, 并且缺失值当中一大部分都存在

以不同的表达方式 (大小写不同) 在之前的数据集中出现的情况 (同一港口), 对于此类情况直接予以去除, 对于确实缺失的港口经纬度信息进行补全;

2. 对于订单数据集 (loadingOrderEvent.csv), 去除其中含有空值的订单, 并且根据 EVENT_LOCATION_ID 在港口信息当中进行查找并补充当前的经纬度信息, 对于查不到的, 经纬度均设为特殊值 (这里为 (200,200));

3. 对于测试数据集 (A_testData0531.csv), 根据 trace 信息在港口信息当中补充相应的起终点的经纬度信息作为特征;

4. 对于训练数据集 (train0523.csv), 注意到数据集中存在可能的 GPS 坐标漂移/速度异常的情况, 第一次清洗考虑根据同一订单不同时刻记录的经纬度差异及时间差估算平均速度, 去除速度过大 (大于 60) 的记录. 第二次清洗则取出同一订单的最后时刻 GPS 坐标, 删除距离最近的港口超过一定范围的记录;

接下来对于本小组成绩的提升过程进行简要描述, 首先基于讨论区 @ 姜大德放出的 baseline 代码并适当扩充训练数据量取得了 7w 分的基础分, 然而在经过初步的数据清洗 (清洗训练集的漂移数据) 之后仅仅将原始的 baseline 从 17w 分提升至 15w 分, 并且扩大训练数据量反而造成了误差的增大. 随后对数据清洗进行了持续的研究并没有收获明显的提升 (第二次对于训练集的清洗反倒使得分数降低), 于是转而对特征工程方向进行研究. 我们团队的做法是将历史订单数据和训练集结合起来, 以添加终点的经纬度信息作为新的特征. 经过对于特征工程的修改, 将 baseline 提升至了 10w 分.

在特征工程再次遇到瓶颈的情况下, 我们怀疑数据清洗 (尤其是第二次基于港口距离的清洗) 当中清洗掉了很多有用的数据, 于是尝试采用原始数据集的基础上修改特征工程, 成功提升至 3.5w 分左右. 由于此时又采用了原始数据集, 考虑到原始数据集当中存在异常数据, 我们在训练过程中引入适当的补偿性修正, 即在原有的训练基础上进行偏移式的修正. 经过几次尝试之后明确了修正的方向以及大小, 通过这几次尝试的修正值以及相应的误差数值使用二次函数进行拟合, 根据拟合的函数表达式确认了在修正数值为 107.95 时取得最小误差, 此时误差为 22336.79.

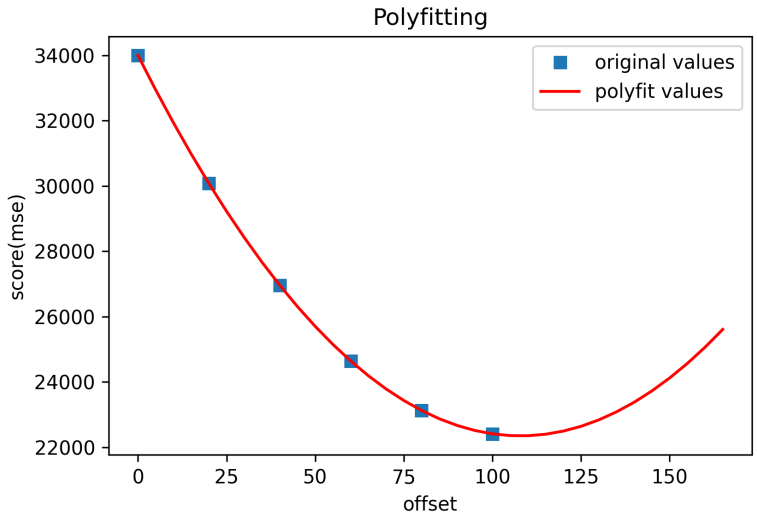

图 2: 拟合曲线

最后, 我们根据之前扩大数据量的经验再次对选取的训练数据集大小进行扩充, 最终取得了

13523.3888 的成绩.

103 ↓ 1	giaogiaogiao	12190.3142	2020/06/12
104 ↓ 1	铅笔猫	13207.3609	2020/06/16
105	ETADingLiWang3EasyPieces	13523.3888	2020/06/19
106	QAH&SC	14373.8813	2020/06/14

图 3: 最终排名 (截止 06/19 12:00)

9 讨论 (Discussion)

在最终结果呈现之前，我们经历了一段通过添加偏移量来提高测试集性能的过程，并且在几次尝试后通过二次函数拟合获得了近似最优解。之所以测试集的 ETA 相比训练集显得更小，一个十分重要的原因在于训练集全部都是整条航线无偏好随机采样得到，而测试数据人为控制在了一条航线的前百分之 10 到 20 左右，因而两者在特征值选取模式完全相同的条件下，测试集将由于自身经纬度变化幅度过小而被“误判”为整体移动速度偏慢，因而会导致系统性的预测 ETA 偏大情况，这里的减去一个特定偏移量正是在修正这种由于数据采样方式不同而导致的系统误差。

第二轮数据清洗之所以会失败，主要原因在于历史运单数据集中的许多订单其记录并非从起点就开始，也并非在终点很近距离内结束，而是存在一些采样缺失或稀疏情况。直接使用首尾位置匹配，一来删去了过多有价值的数据（清洗率越 85%），而来将一些航线错误匹配到了不属于自己的起点和终点，进而极大干扰了特征值的结果，最终导致学习性能下降。

10 结论 (Conclusion)

本次海运时间预测（ETA）比赛中，中我们对训练集和测试集中的数值、非数值属性进行了一定成程度的特征工程并且基于特征进行了数据清洗，最后使用lightGBM作为学习框架，取得了如下图（可以加序号）所示的成绩。我们认为下一步研究的重点应当放在特征工程这一块，如何充分利用训练运单数据，而不单单是对各个转换为数值的属性求其统计量，是提升最终预测性能上限的重要一环。在最初的尝试阶段我们准备试验使用深度神经网络帮助进行特征提取，但是在具体操作的过程中由于时间关系这部分被省略了，这也是未来特征工程可能的改进方向之一。此外，现阶段项目中只有 `dataframe.apply` 方法使用了并行库进行加速，未来也许可以在算法层面加以优化，充分发挥 CPU 和 GPU 的性能。

11 致谢 (Ackowledgement)

- 感谢 ID 为姜大德的用户在讨论区中开源 baseline，让我们了解从数据到提交的整体流程
- 感谢刘润泽 (181220036)、施顶威 (181220047)、赵瑞卿 (181220072) 的三人小组向我们提供调参经验、与我们交流
- 感谢陈翔 (181220005) 同学的小组与我们交流
- 感谢宋磊 (181220049) 同学与我们交流
- 感谢XGBoost[9] 以及lightGBM[10] 的开发者以及文档提供者，使我们能够在较短的时间内应用进工程当中并取得结果

参考文献

[1] TAY C, MEKHNACHA K, LAUGIER C. Probabilistic Vehicle Motion Modeling and Risk Estimation[M/OL]// ESKANDARIAN A. Handbook of Intelligent Vehicles: volume 2. Springer, 2012: 1479-1516. https://hal.inria.fr/hal-00779183.

[2] Chun-Hsin Wu, Chia-Chen Wei, Da-Chun Su, et al. Travel time prediction with support vector regression[C]//Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems: volume 2. [S.l.: s.n.], 2003: 1438-1442 vol.2.

[3] MYUNG J, KIM D K, KHO S Y, et al. Travel time prediction using k nearest neighbor method with combined data from vehicle detector system and automatic toll collection system[J/OL]. Transportation Research Record, 2011, 2256 (1):51-59. https://doi.org/10.3141/2256-07.

[4] BODUNOV O, SCHMIDT F, MARTIN A, et al. Real-time destination and eta prediction for maritime traffic[C/OL]// DEBS ' 18: Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems. New York, NY, USA: Association for Computing Machinery, 2018: 198–201. https://doi.org/10.1145/3210284.3220502.

[5] MESTL T, GL D, NORWAY H, et al. Port eta prediction based on ais data[C]//[S.l.: s.n.], 2016.

[6] Cory Maklin. Gradient boosting decision tree algorithm explained[EB/OL]. 2019. https://towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4.

[7] Wikipedia contributors. Gradient boosting — Wikipedia, the free encyclopedia[EB/OL]. 2020. https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=960490186.

[8] https://github.com/dask/dask. Dask documentation[EB/OL]. 2020. https://docs.dask.org/en/latest/.

[9] https://github.com/dmlc. Xgboost documentation[EB/OL]. 2020. https://xgboost.readthedocs.io/en/latest/.

[10] https://github.com/Microsoft/. Lightgbm documentation[EB/OL]. 2020. https://lightgbm.readthedocs.io/en/latest/.

12 附录 (Appendix)

工程文件共有一个用来存放数据(未上传)的 data 文件夹,两个数据预处理文件 port_fix.ipynb、data_wash.py，一个主要学习文件 LGB_baseline.ipynb，一个结果拟合文件 Polyfit.ipynb，一个报告文件 report.pdf 构成，其中各个文件以及部分关键函数的作用详见下图。

```
submit─┬─data─┬─origin─┬─train0523.csv                # 历史运单数据
       |      |        ├─loadingOrderEvent.csv       # 港口事件数据
       |      |        ├─port.csv                    # 港口信息数据
       |      |        └─A_testData0531.csv          # 测试数据
       |      ├─port_fixed.csv              # 补全、去重后的港口数据
       |      ├─wash3_test.csv              # 增添了起点终点经纬度
       |      └─wash1_event.csv             # 增加了经纬度
       ├─data_wash.py─┬─set_data_columns    # 重命名训练数据的列名
       |              ├─cor2speed           # 通过经纬度差值计算平均速度km/h
       |              ├─port2cordinate      # 将港口名称通过查表转换为坐标
       |              ├─trace2cordinate     # 将航行记录转换为起点、终点两个坐标
       |              ├─NN                  # 寻找最近的港口，返回港口坐标以及距离港口距离
       |              ├─speed_wash          # 使用地理库计算平均距离，并清洗掉速度过大的
       |              ├─test_begin_end      # 根据测试集最后一列Trace,添加起点与终点的坐标
       |              ├─distance_wash       # 训练集匹配起点终点最近邻港口，并清洗掉过远的
       |              └─event_feature       # 事件数据集 根据港口信息添加经纬度数据
       ├─port_fix.ipynb                     # 手动补全port.csv港口信息使用的一些辅助
       ├─LGB_baseline.ipynb─┬─get_data      # 对数据集列名规范化处理
       |                    ├─get_feature   # 特征工程
       |                    └─build_model   # 建模与训练
       ├─Polyfit.ipynb                      # 通过二次函数拟合修正测试集与训练集系统性误差
       └─report.pdf                         # 实验报告
```

13 其他说明

小组成员及分工如下:

- 丁豪 (181220010) 负责特征工程 (数据清洗), 学习算法的调试，分析最终数据偏移与误差关系
- 李惟康 (18122031) 负责多线程并行库 (dask), 地理数据库 (geopandas) 的学习与使用, 港口信息补全与清理, 云平台的部署使用，
- 王宸旭 (181220056) 负责 GDBT、XGBoost 和 lgbm 模型的理论学习以及使用调参，支持增量学习的 lgbm 模型，港口清洗，文献综述部分论文阅读

报告写到这里就快结束了，回顾 5 月 29 号匆匆组队选题的起点，这二十天的种种场景仿佛黄粱一梦，十分感慨。借此机会，想陈述一点自己 (组员王宸旭) 的心绪。二十天走来，颇为不顺，前中期做了大量的研讨和特征过程都未取得很好的效果，以至于到截止前三天 (6 月 15 号) 我们的最好成绩还停留在 6 月 7 号提交的七万误差。看着自己的排名不断被别的队伍超过，其他队伍里的大佬同学把队伍带飞，每一个夜晚入睡前我们的心里都充满不甘和疑惑。但我们队伍的每一个人都没有放弃，都在克服自己的惰性尝试自己不熟悉的事物，每次开完组会就开始对着陌生的英文文档和论文学习，说服自己一定要学会然后投入应用，以期获得分数的提高。

说起来这是大学以来第一次正式参加自己专业相关的比赛，尽管成绩不尽人意，但是相比于成绩，这二十天更大的收获是完整的参赛流程体验以及面对实际问题时自己如何去解决、怎样解决好的过程。团队分工完的那一刻，每个队员就是一个方向独当一面的一颗棋子，面对什么

样的困难都要尝试解决，不能辜负队友的努力和信任。我相信这也是培养计划里各种实验、ICS PA、问题求解、OS Lab 还有本课程最终想教会我们的能力。

最后再次感谢队伍里每一个人的付出，感谢任课老师和助教本学期提供的教诲和帮助，感谢其他给我们提供帮助的队伍和同学，感谢课程组给我们这样一个体验参赛的机会。