NNI学生项目2020

Task 1.3.1

ID: 14 Name: 中科大飙车队

任务描述

跑通 NNI Feature Engineering Sample

代码分析

我们只对作者给出的第一个简单的例子分析，针对其他数据集的情况可以作简单的类比即可。

数据预处理

原文的数据集是一个二分类问题，数据量也不算大。有1999个样本，每个样本的数据维度是40维。

原文中的数据有一些特点：一个是有部分的缺失值，一个是有不少特征是非数值的特征。对于这两个情况，作者对于缺失值的情况采取了Pandas中的fillna来解决问题，对于非数值特征的情况，采用了sklearn.preprocessing的LabelEncoder进行处理，LabelEncoder是一种对于非数值特征简单编码为数值特征的方法。主要的实现在model.py当中。

机器学习算法

原文中采用的机器算法是LightGBM，是一种基于决策树的集成学习算法，特点是训练速度很快，效率很高，即使采用NNI来自动调参也会比较节省时间。主要的实现在model.py当中。

特征选取

原生的NNI并不支持高阶的特征组合和选取，只有两种比较简单的工具GradientFeatureSelector和GBDTSelector。而该项目的作者没有选用原生NNI的特征选取工具，而是做了自己的实现。

我们首先介绍作者组合特征的方式，在const.py中作者使用了多种组合的方式，包括count,crosscount,aggregate_{min,max,mean,median,var},nunique,histstat,target,embedding等组合方式。最终的特征维数为128维。

> count: 计算某一列中的每个特征出现的次数并作为样本新的特征
>
> crosscount:将某两列或多列的组合特征当成单个特征$x_i' = (x_j, x_k)$，计算出现的次数并作为样本新的特征
>
> nunique: 考虑某列的元素是否为唯一的
>
> aggregate: 把两列中的元素合并到一起，每个样本仅由一行来代表，计算min max等对应的统计量
>
> embedding:在多类别的特征上做，当作自然语言做编码
>
> histstat在直方图上得到聚类的结果

作者在search_space.json里面也只使用了count,aggregate,crosscount这三种方法。

更新特征

这里更新特征的方式有点类似于遗传算法中的轮盘赌算法，根据上一次算法运行的结果得出特征的重要性排序，得到排序之后更新每个特征的分数，得到分数之后将分数换算为概率，按照对应的概率进行随机采样，通过采样得出下一轮所使用的特征。

环境配置

在Windows下开展实验

1. 配置NNI基础环境。原始仓库中要求NNI为0.9.1版本，实际上配置1.5最新版本也可以。
2. 通过git clone的方式下载代码包
3. 配置需要的其他安装包。

```
pip install lightgbm
```

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple gensim
```

4. 使用NNI命令运行程序

```
nnictl create --config config.yml
```

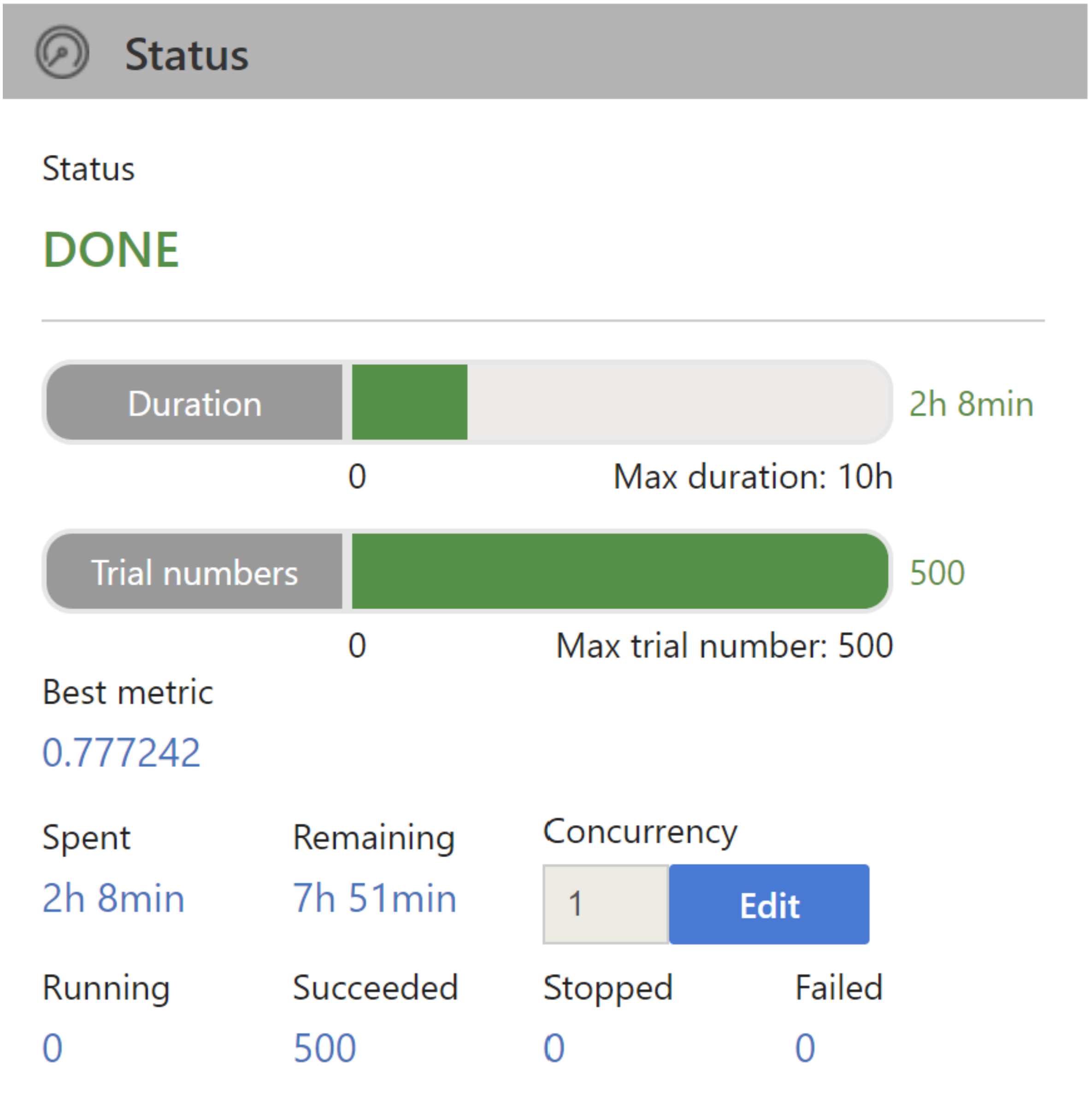
一些不算坑的小坑

- 在服务器上配置，队友们都会发现各种各样奇怪的问题
- 程序运行开始会报错，有可能是config.yml的这里出现问题

```
trial:
  command: python3 main.py #如果出现bug可以试着改为python
  codeDir: .
  gpuNum: 0
```

实现效果

我们运行了500次迭代，总计花费的时间为128分钟，具体的数据如下：



最后筛选出的最好的10个迭代的结果为:

Trial No.	ID	Duration	Status	Default metric
131	HxFqn	5s	SUCCEEDED	0.777242
177	Fklmn	4s	SUCCEEDED	0.776505
179	eG2ho	4s	SUCCEEDED	0.775434
227	eBabb	5s	SUCCEEDED	0.773357
238	DqH6f	4s	SUCCEEDED	0.773157
158	iVB4P	3s	SUCCEEDED	0.773157
219	ERGBB	4s	SUCCEEDED	0.772989
148	G4U44	4s	SUCCEEDED	0.772822
277	LysO4	3s	SUCCEEDED	0.77242
305	bHwlH	5s	SUCCEEDED	0.772051

结果分析

- 由于特征的选择和更新是通过随机采样得到的，因此并不能保证每一次的更新都能提升效果，实际上在本例中，最后的200轮基本没有得到更好的结果。
- 最后选出的最佳参数组合如下：

```
{
    "sample_feature": [
        "crosscount_C11_C6",
        "aggregate_min_I9_C17",
        "crosscount_C19_C24",
        "crosscount_C1_C17",
        "crosscount_C17_C26",
        "aggregate_var_I11_C23",
        "aggregate_max_I9_C14",
        "aggregate_median_I9_C7",
        "aggregate_median_I9_C3",
        "crosscount_C17_C19",
        "crosscount_C1_C11",
        "crosscount_C11_C20",
        "crosscount_C13_C8",
        "aggregate_mean_I11_C17",
        "aggregate_mean_I11_C24",
        "crosscount_C2_C21",
        "aggregate_median_I9_C9",
        "aggregate_mean_I9_C11",
        "crosscount_C2_C24",
        "crosscount_C13_C5",
        "crosscount_C18_C5",
        "crosscount_C6_C7",
        "crosscount_C17_C4",
        "aggregate_min_I11_C1",
        "crosscount_C21_C9",
        "crosscount_C16_C26",
        "aggregate_median_I11_C3",
        "crosscount_C13_C15",
        "aggregate_mean_I11_C8",
        "crosscount_C23_C6",
        "crosscount_C20_C23",
        "aggregate_median_I9_C21",
        "crosscount_C16_C2",
        "aggregate_median_I11_C26",
        "crosscount_C23_C5",
        "aggregate_min_I10_C17",
        "crosscount_C16_C26",
        "crosscount_C12_C4",
        "crosscount_C15_C8",
        "crosscount_C12_C24",
        "crosscount_C18_C3",
        "crosscount_C14_C18",
        "crosscount_C14_C17",
        "crosscount_C13_C6",
        "crosscount_C2_C20",
        "aggregate_median_I12_C21",
        "aggregate_max_I11_C7",
        "aggregate_var_I12_C16",
```

```
            "aggregate_mean_I12_C16",
            "crosscount_C17_C5",
            "aggregate_min_I11_C24",
            "crosscount_C15_C24",
            "crosscount_C15_C4",
            "crosscount_C16_C23",
            "crosscount_C10_C16",
            "crosscount_C2_C3",
            "aggregate_min_I12_C4",
            "crosscount_C22_C23",
            "crosscount_C15_C7",
            "crosscount_C17_C24",
            "aggregate_max_I10_C17",
            "aggregate_median_I10_C24",
            "aggregate_var_I9_C23",
            "crosscount_C17_C4",
            "crosscount_C12_C22",
            "crosscount_C15_C3",
            "aggregate_min_I12_C17",
            "crosscount_C21_C23",
            "crosscount_C12_C17",
            "aggregate_max_I11_C14",
            "crosscount_C16_C7",
            "aggregate_var_I11_C3",
            "crosscount_C20_C3",
            "crosscount_C17_C23",
            "aggregate_median_I11_C15",
            "crosscount_C3_C7",
            "crosscount_C2_C3",
            "aggregate_median_I12_C12",
            "aggregate_min_I11_C6",
            "aggregate_max_I9_C9",
            "aggregate_var_I11_C24",
            "crosscount_C17_C7",
            "crosscount_C21_C9",
            "crosscount_C19_C23",
            "aggregate_var_I11_C4",
            "crosscount_C10_C23",
            "crosscount_C11_C24",
            "aggregate_mean_I9_C17",
            "crosscount_C12_C5",
            "crosscount_C6_C9",
            "aggregate_var_I10_C4",
            "crosscount_C11_C23",
            "aggregate_median_I9_C19",
            "aggregate_min_I11_C5",
            "crosscount_C13_C15",
            "aggregate_median_I10_C23",
            "aggregate_max_I12_C4",
            "aggregate_mean_I11_C7",
            "aggregate_max_I9_C13",
            "crosscount_C17_C8",
            "aggregate_mean_I10_C13",
            "crosscount_C11_C21",
            "crosscount_C14_C21",
            "crosscount_C11_C25",
            "crosscount_C19_C20",
            "crosscount_C1_C17",
```

```
        "crosscount_C1_C2",
        "crosscount_C11_C14",
        "aggregate_min_I10_C21",
        "crosscount_C15_C17",
        "aggregate_mean_I11_C23",
        "aggregate_median_I11_C5",
        "aggregate_min_I9_C9",
        "crosscount_C1_C12",
        "aggregate_var_I10_C7",
        "aggregate_min_I11_C10",
        "aggregate_max_I9_C11",
        "aggregate_var_I11_C1",
        "aggregate_median_I12_C24",
        "aggregate_max_I9_C15",
        "aggregate_mean_I9_C2",
        "aggregate_min_I9_C4",
        "crosscount_C16_C23",
        "crosscount_C12_C8",
        "aggregate_var_I9_C13",
        "crosscount_C12_C17",
        "crosscount_C2_C5",
        "aggregate_max_I11_C4"
    ]
}
```

可以看出选出的特征以crosscount为最多，而aggregate次之，选出count的没有，可见count的表现力不够强，还是crosscount更能完整的呈现出样本。

下一阶段的考虑

- 优化特征选取更新的算法，能更充分地进行搜索，尤其是尽可能让搜索的结果随着次数的增加变得更好。另外能更好地做出exploitation和exploration的trade off，在时间允许的情况下设计出更加新颖的算法。目前不成熟的想法是借鉴MAB问题。
- 换一个数据集进行尝试，并跑出baseline。
- 对原作者的源码进行更加深入的理解，争取进行更加有效的更新和深入。