机器学习编码方法总结!

尤而小屋 2022-07-08 00:09 发表于北京

编者荐语:

特征工程必备知识点!

以下文章来源于机器学习实验室 ，作者louwill

机器学习实验室

专注于机器学习和深度学习技术与实践。

在做结构化数据训练时，类别特征是一个非常常见的变量类型。机器学习中有多种类别变量编码方式，各种编码方法都有各自的适用场景和特点。本文就对机器学习中常见的类别编码方式做一个简单的总结。

1、硬编码：Label Encoding

所谓硬编码，即直接对类别特征进行数值映射，有多少类别取值就映射多少数值。这种硬编码方式简单粗暴，方便快捷。

但其仅在类别特征内部取值是有序的情况才好使用，即类别特征取值存在明显的顺序性，比如说：

- 学历特征取值为高中、本科、硕士和博士，各学历之间存在明显的顺序关系；
- 成绩特征取值为不及格、良好、优秀，这种取值之间也是存在明显的顺序关系

Sklearn提供了Label Encoding的实现方式，示例代码如下：

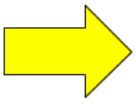
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(['undergraduate', 'master', 'PhD', 'Postdoc'])
le.transform(['undergraduate', 'master', 'PhD', 'Postdoc'])

# 结果
array([3, 2, 0, 1], dtype=int64)
```

尤而小屋

2、独热编码：One-hot Encoding

One-hot编码应该是应用最广泛的类别特征编码方式了。假设一个类别特征有m个类别取值，通过One-hot编码我们可以将其转换为m个二元特征，每个特征对应该取值类别。

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

对于类别特征内部取值不存在明显的内在顺序时，即直接的硬编码不适用时，One-hot编码的作用就凸显出来了。

但当类别特征取值过多时，One-hot编码很容易造成维度灾难，特别是对于文本类的特征，如果使用One-hot编码对其进行编码，基本上都是茫茫零海。所以，在类别特征取值无序，且特征取值数量少于5个时，可使用One-hot方法进行类别编码。

有朋友可能会问，一定得是5个吗，6个行不行，当然也可以，这里并没有固定标准，但差不多就是这个数据左右。数量再多就不建议使用One-hot了。

Pandas和Sklearn都提供了One-hot编码的实现方式，示例代码如下。

```python
import pandas as pd
df = pd.DataFrame({'f1':['A','B','C'],
'f2':['Male','Female','Male']})
df = pd.get_dummies(df, columns=['f1', 'f2'])
df
```


	f1_A	f1_B	f1_C	f2_Female	f2_Male
0	1	0	0	0	1
1	0	1	0	1	0
2	0	0	1	0	1

```python
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')
X = [['Male', 1], ['Female', 3], ['Female', 2]]
enc.fit(X)
enc.transform([['Female', 1], ['Male', 4]]).toarray()

# 结果
array([[1., 0., 1., 0., 0.],
[0., 1., 0., 0., 0.]])
```


3、目标变量编码：Target Encoding

Target Encoding就是用目标变量的类别均值来给类别特征做编码。CatBoost中就大量使用目标变量统计的方法来对类别特征编码。

- 但在实际操作时，直接用类别均值替换类别特征的话，会造成一定程度的标签信息泄露的情况，主流方法是使用两层的交叉验证来计算目标均值。

- Target Encoding一般适用于类别特征无序且类别取值数量大于5个的情形。

参考代码如下：

```
### 该代码来自知乎专栏：
### https://zhuanlan.zhihu.com/p/40231966

from sklearn.model_selection import KFold
n_folds = 20
n_inner_folds = 10
likelihood_encoded = pd.Series()
likelihood_coding_map = {}

# train -> train dataframe
# test -> test dataframe

oof_default_mean = train[target].mean()      # global prior mean
kf = KFold(n_splits=n_folds, shuffle=True)
oof_mean_cv = pd.DataFrame()
split = 0

for infold, oof in kf.split(train[feature]):
    print ('===level 1 encoding..., fold %s ===' % split)
    inner_kf = KFold(n_splits=n_inner_folds, shuffle=True)
    inner_oof_default_mean = train.iloc[infold][target].mean()
    inner_split = 0
    inner_oof_mean_cv = pd.DataFrame()

    likelihood_encoded_cv = pd.Series()
    for inner_infold, inner_oof in inner_kf.split(train.iloc[infold]):
        print ('===level 2 encoding..., inner fold %s ===' % inner_split)
        # inner out of fold mean
        oof_mean = train.iloc[inner_infold].groupby(by=feature)[target].mean()
        # assign oof_mean to the infold
        likelihood_encoded_cv = likelihood_encoded_cv.append(train.iloc[infold].apply(
            lambda x : oof_mean[x[feature]]
            if x[feature] in oof_mean.index
            else inner_oof_default_mean, axis = 1))
        inner_oof_mean_cv = inner_oof_mean_cv.join(pd.DataFrame(oof_mean), rsuffix=inner_spli
        inner_oof_mean_cv.fillna(inner_oof_default_mean, inplace=True)
        inner_split += 1

    oof_mean_cv = oof_mean_cv.join(pd.DataFrame(inner_oof_mean_cv), rsuffix=split, how='outer
    oof_mean_cv.fillna(value=oof_default_mean, inplace=True)
    split += 1
    print ('============final mapping...===========')
    likelihood_encoded = likelihood_encoded.append(train.iloc[oof].apply(
        lambda x: np.mean(inner_oof_mean_cv.loc[x[feature]].values)
        if x[feature] in inner_oof_mean_cv.index
        else oof_default_mean, axis=1))

# map into test dataframe
train[feature] = likelihood_encoded
likelihood_coding_mapping = oof_mean_cv.mean(axis = 1)
default_coding = oof_default_mean

likelihood_coding_map[feature] = (likelihood_coding_mapping, default_coding)
mapping, default_mean = likelihood_coding_map[feature]
test[feature] = test.apply(lambda x : mapping[x[feature]]
                                    if x[feature] in mapping
                                    else default_mean,axis = 1)
```

4、模型自动编码

在LightGBM和CatBoost等算法中，模型可以直接对类别特征进行编码，实际使用时直接将类别特征标记后传入对应的api即可。一个示例代码如下：

```
lgb_train = lgb.Dataset(train2[features], train2['total_cost'],
                        categorical_feature=['sex'])
```

5、总结

根据本文的梳理，可总结机器学习中类别特征的编码方式如下：

- Label Encoding
  - 类别特征内部有序
- One-hot Encoding
  - 类别特征内部无序
  - 类别数值<5
- Target Encoding
  - 类别特征内部无序
  - 类别数值>5
- 模型自动编码
  - LightGBM
  - CatBoost

推荐阅读

MySQL必须掌握4种语言!

三大树模型实战乳腺癌预测分类

Plotly+Pandas+Sklearn：实现用户聚类分群!

用户群组分析，Python实现!

Kaggle可视化：黑色星期五画像分析

kaggle实战：可视化深度探索苹果AppStores

kaggle实战：6大回归模型预测航班票价

> 尤而小屋，一个温馨的小屋。小屋主人，一手代码谋求生存，一手掌勺享受生活，欢迎你的光临

尤而小屋

尤而小屋，一个温馨且有爱的小屋🏠 小屋主人，一手代码谋求生存，一手掌勺享受生...

261篇原创内容

公众号