人工智能程序设计实验报告

聚类-关联-异常

白晋斌
171860607

目录

完成以下三个任务，提交源代码和实验报告。请在实验报告中记录你的实验过程、实验结果和思考。

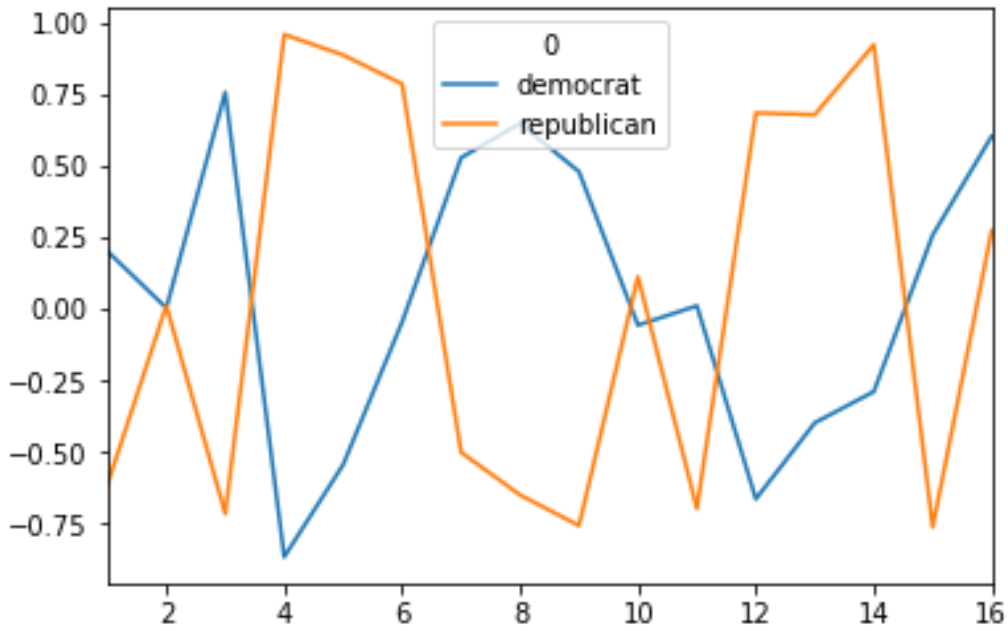
任务 1 聚类（对应文件 house-votes-84.csv）

任务描述：数据中给定议员对于 16 项不同议题的投票情况（属性表示为支持 Y、反对 N、弃权？），数据的第一列为议员所属的党派类别，请使用聚类方法对上述数据进行聚类分析：

1. 尝试定量的评价的聚类结果；

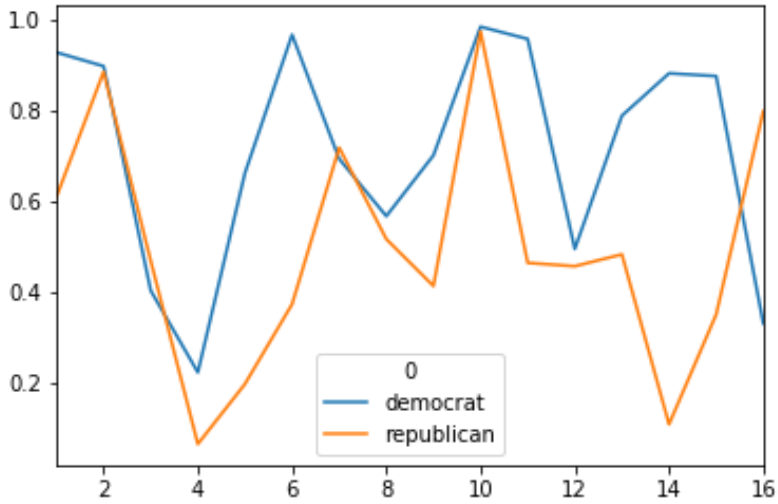
首先，我们分析原表中的数据。对赞同、反对、弃权做数字化处理后，通过将两个党派的数据分别统计，得到他们的对每项议题的平均支持率，并做了可视化。

0	democrat	republican
1	0.202247	-0.613095
2	0.003745	0.011905
3	0.756554	-0.714286
4	-0.865169	0.958333
5	-0.543071	0.886905
6	-0.044944	0.785714
7	0.528090	-0.500000
8	0.647940	-0.648810
9	0.479401	-0.755952
10	-0.056180	0.113095
11	0.011236	-0.696429
12	-0.662921	0.684524
13	-0.397004	0.678571
14	-0.288390	0.922619
15	0.258427	-0.761905
16	0.602996	0.273810



从上图可以看出，两党之前的赞同反对区别还是比较明显的。
再分析一下方差。

0	democrat	republican
1	0.928867	0.609887
2	0.898482	0.886085
3	0.402917	0.468777
4	0.222354	0.064122
5	0.662612	0.196714
6	0.967897	0.372968
7	0.693757	0.718563
8	0.567317	0.516645
9	0.701642	0.413138
10	0.985554	0.975157
11	0.958520	0.464179
12	0.494973	0.456765
13	0.789164	0.482891
14	0.882684	0.107749
15	0.876573	0.350157
16	0.330517	0.798831

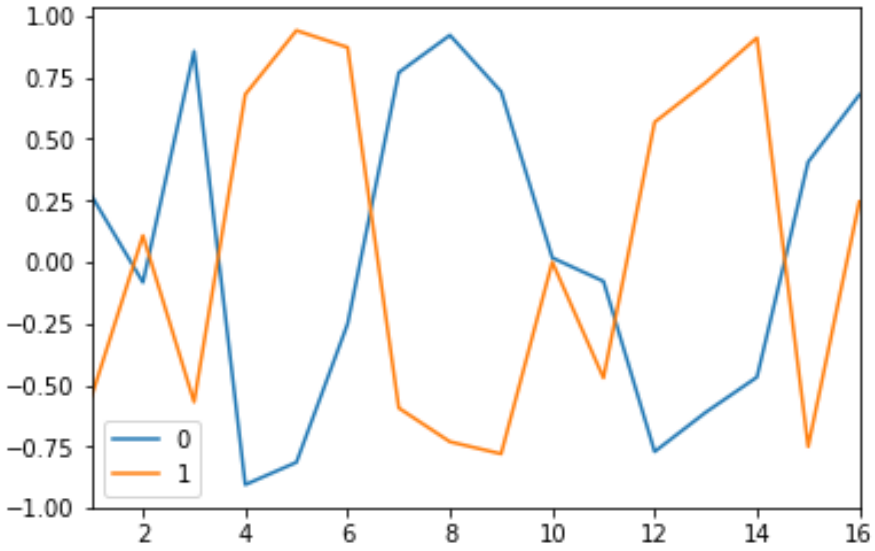


可以看出，二者在分歧度方面比较相近。因此，我们之后在聚类时主要采用平均值作为评判标准。

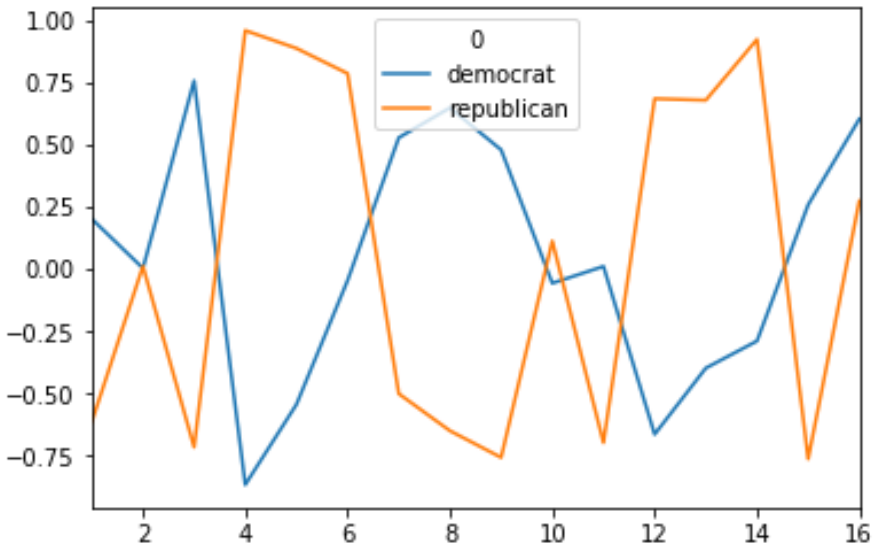
这里我们采用 kmeans 作为聚类方法，将无标签数据分为两类，初始化质心设置为 10，初始值选择方式为 kmeans++。

结果如下图。

	0	1
1	0.268398	-5.441176e-01
2	-0.082251	1.078431e-01
3	0.857143	-5.686275e-01
4	-0.904762	6.813725e-01
5	-0.813853	9.411765e-01
6	-0.251082	8.725490e-01
7	0.770563	-5.931373e-01
8	0.922078	-7.303922e-01
9	0.692641	-7.794118e-01
10	0.017316	1.630640e-16
11	-0.077922	-4.705882e-01
12	-0.770563	5.686275e-01
13	-0.610390	7.303922e-01
14	-0.467532	9.117647e-01
15	0.406926	-7.500000e-01
16	0.679654	2.450980e-01



这里直接看不直观，我们把带标签的统计结果放在下面。大家可以上图与下图进行对比。显然，聚类结果表现不错。对比各项平均值，聚类结果与处世结果也很接近。



接下来是量化分析。通过对先后生成的平均值作差，得下表。

	0	1
1	-0.066151	-0.068978
2	0.085996	-0.095938
3	-0.100589	-0.145658
4	0.039593	0.276961
5	0.270782	-0.054272
6	0.206138	-0.086835
7	-0.242473	0.093137
8	-0.274138	0.081583
9	-0.213240	0.023459
10	-0.073496	0.113095
11	0.089158	-0.225840
12	0.107641	0.115896
13	0.213386	-0.051821
14	0.179143	0.010854
15	-0.148499	-0.011905
16	-0.076657	0.028711

可以看出，在大多数情况下，约 50%的聚类误差都低于 10%，证明我们的聚类效果良好。

1) **n_clusters**: 即我们的 k 值，一般需要多试一些值以获得较好的聚类效果。k 值好坏的评估标准在下面会讲。

2）**max_iter**： 最大的迭代次数，一般如果是凸数据集的话可以不管这个值，如果数据集不是凸的，可能很难收敛，此时可以指定最大的迭代次数让算法可以及时退出循环。

3）**n_init :** 用不同的初始化质心运行算法的次数。由于 K-Means 是结果受初始值影响的局部最优的迭代算法，因此需要多跑几次以选择一个较好的聚类效果，默认是 10，一般不需要改。如果你的 k 值较大，则可以适当增大这个值。

4）**init :** 即初始值选择的方式，可以为完全随机选择'random',优化过的'k-means++'或者自己指定初始化的 k 个质心。一般建议使用默认的'k-means++'。

5）**algorithm**：有"auto", "full" or "elkan"三种选择。"full"就是我们传统的 K-Means 算法， "elkan"是我们原理篇讲的 elkan K-Means 算法。默认的"auto"则会根据数据值是否是稀疏的，来决定如何选择"full"和"elkan"。一般数据是稠密的，那么就是 "elkan"，否则就是"full"。一般来说建议直接用默认的"auto"

2. 尝试比较不同聚类方法或者不同初始情况对聚类结果的影响;

那么能不能更进一步优化呢?

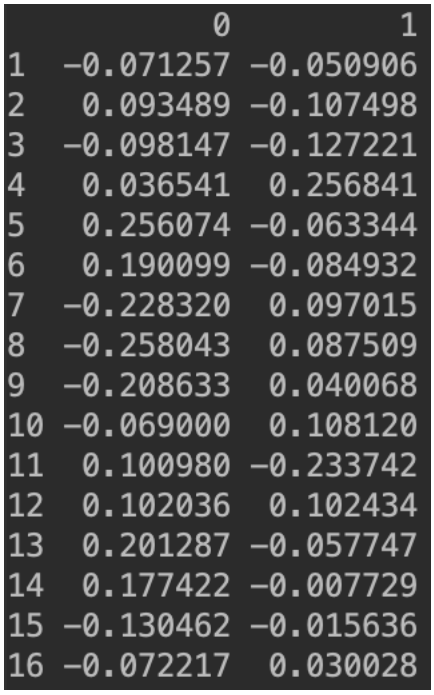
100 个初始化质心:

	0	1
1	-0.069305	-0.061371
2	0.089952	-0.101396
3	-0.101204	-0.137931
4	0.040004	0.268678
5	0.262963	-0.053982
6	0.200746	-0.086207
7	-0.234841	0.091133
8	-0.265853	0.080255
9	-0.205944	0.022373
10	-0.069111	0.108169
11	0.093133	-0.228448
12	0.108630	0.108169
13	0.215065	-0.060345
14	0.181438	0.001437
15	-0.142435	-0.013136
16	-0.078038	0.032430

5 个初始化质心:

	0	1
1	-0.071257	-0.050906
2	0.093489	-0.107498
3	-0.098147	-0.127221
4	0.036541	0.256841
5	0.256074	-0.063344
6	0.190099	-0.084932
7	-0.228320	0.097015
8	-0.258043	0.087509
9	-0.208633	0.040068
10	-0.069000	0.108120
11	0.100980	-0.233742
12	0.102036	0.102434
13	0.201287	-0.057747
14	0.177422	-0.007729
15	-0.130462	-0.015636
16	-0.072217	0.030028

2 个初始化质心:



	0	1
1	-0.071257	-0.050906
2	0.093489	-0.107498
3	-0.098147	-0.127221
4	0.036541	0.256841
5	0.256074	-0.063344
6	0.190099	-0.084932
7	-0.228320	0.097015
8	-0.258043	0.087509
9	-0.208633	0.040068
10	-0.069000	0.108120
11	0.100980	-0.233742
12	0.102036	0.102434
13	0.201287	-0.057747
14	0.177422	-0.007729
15	-0.130462	-0.015636
16	-0.072217	0.030028

通过上述统计结果，我们发现，对于 2 聚类任务，质心初始数目越多，结果精确性越高，但不是很明显。对于聚类数 k 很大的任务时，初始质心越多，将大大提高聚类精确性。

任务 2 关联分析（对应文件 car.data.csv）

任务描述：给定二手车评估情况的数据，第一行为各类属性名称，最后一列为二手车的评估情况分 类，以比例分为如下 4 类：

等级类别	样例数	占比（%）
unacc	1210	70.023
Acc	384	22.222
Good	69	3.993
V-good	65	3.762

各属性的可能取值如下（并不全是数值）：
buying： v - high, high, med, low
maint： v - high, high, med, low
doors： 2, 3, 4, 5
more

persons： 2, 4, more
lug_boot： small, med, big
safety： low, med, high

1. 请对数据进行关联分析，计算哪些属性的哪些值对于每个分类结果有较强的关联（请自定置信度 和支持的阈值）。

通过对数据一些简单的预处理（详见代码，主要是把文本数字化），通过 apriori 算法，设置最小支持度为 0.2，得出频繁项集后选择长度 >=2 的来求关联规则，这里设置置信度为 0.7.得到如下统计结果。

```
      support                        itemsets
 29  0.333333      (safety_low, class_unacc)
 28  0.333333       (persons_2, class_unacc)
 26  0.260417  (class_unacc, lug_boot_small)
 25  0.226852    (class_unacc, lug_boot_med)
 24  0.212963    (class_unacc, lug_boot_big)
 23  0.208333    (buying_vhigh, class_unacc)
 27  0.208333     (class_unacc, maint_vhigh)
 30  0.206597      (safety_med, class_unacc)
        antecedents    consequents  antecedent support  consequent support  \
0      (safety_low)  (class_unacc)            0.333333            0.700231
1       (persons_2)  (class_unacc)            0.333333            0.700231
3    (buying_vhigh)  (class_unacc)            0.250000            0.700231
4     (maint_vhigh)  (class_unacc)            0.250000            0.700231
2  (lug_boot_small)  (class_unacc)            0.333333            0.700231

    support  confidence      lift  leverage  conviction
0  0.333333    1.000000  1.428099  0.099923         inf
1  0.333333    1.000000  1.428099  0.099923         inf
3  0.208333    0.833333  1.190083  0.033275    1.798611
4  0.208333    0.833333  1.190083  0.033275    1.798611
2  0.260417    0.781250  1.115702  0.027006    1.370370
```

可以发现，安全性低的，载客量小的，价格贵的，维修度高的，行李箱小的，往往被定义为 unacc（这里我的理解是差车），这也与我们的直觉相符合。统计结果较为不错。其中，安全性对车的评级最为重要。

2. 尝试用前述聚类方法结果对该数据进行聚类，尝试分析聚类结果与关联分析的联系。

通过对数据一些简单的预处理（详见代码，主要是把文本数字化）

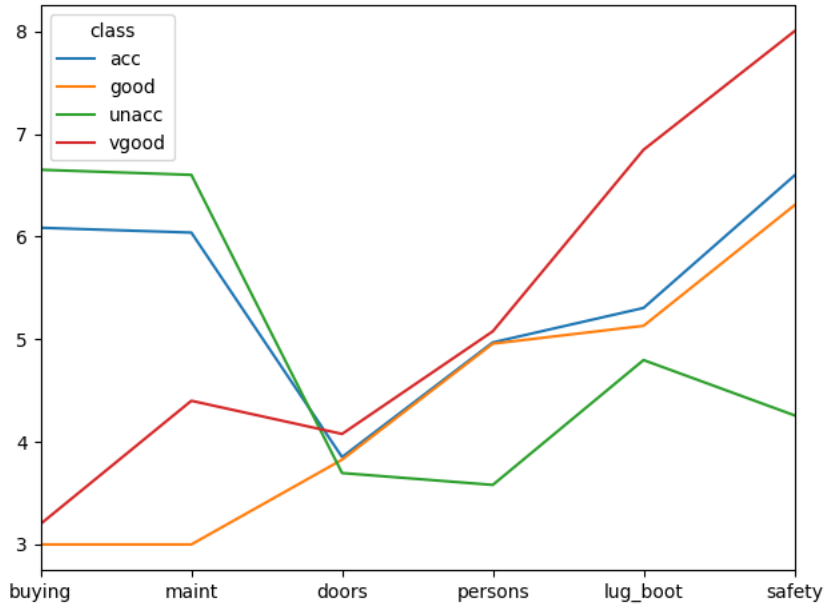
详细数字化算法如下：

```
def myfunc(k):
    if k=='vhigh':
        return 10
    if k=='high':
        return 8
    if k=='med':
        return 5
    if k=='low':
```

```
    return 2
if k=='5more':
    return 6
if k=='more':
    return 6
if k=='small':
    return 2
if k=='big':
    return 8
return int(k)
```

再按原始数据的标签进行分类，统计，得下图。

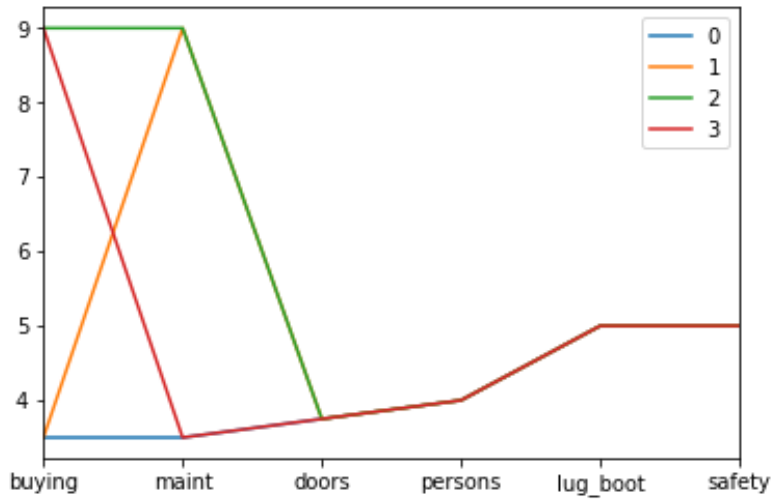
class	acc	good	unacc	vgood
buying	6.085938	3.000000	6.651240	3.200000
maint	6.039062	3.000000	6.601653	4.400000
doors	3.851562	3.826087	3.695868	4.076923
persons	4.968750	4.956522	3.580165	5.076923
lug_boot	5.304688	5.130435	4.796694	6.846154
safety	6.593750	6.304348	4.258678	8.000000



可以发现，主要是安全性和价格区分出最好的车和最坏的车，再通过维修程度区分出好车和一般般的车。

我们再通过 kmeans 算法，对结果聚类。

	buying	maint	doors	persons	lug_boot	safety	类别数目
0	3.5	3.5	3.75	4.0	5.0	5.0	432
1	3.5	9.0	3.75	4.0	5.0	5.0	432
2	9.0	9.0	3.75	4.0	5.0	5.0	432
3	9.0	3.5	3.75	4.0	5.0	5.0	432



kmeans 的聚类效果不是很好，但依旧完成 40%的区分度，那就是价格和维修程度是决定车的低级高级的主要原因。这也符合我们的直觉。

同时，对比第一问，价格贵的，维修度高的，载客量小的，往往被定义为 unacc（这里我的理解是差车），在这幅图中可以很好的体现出这三个方面，也就相互验证了第一问关联分析的结果。

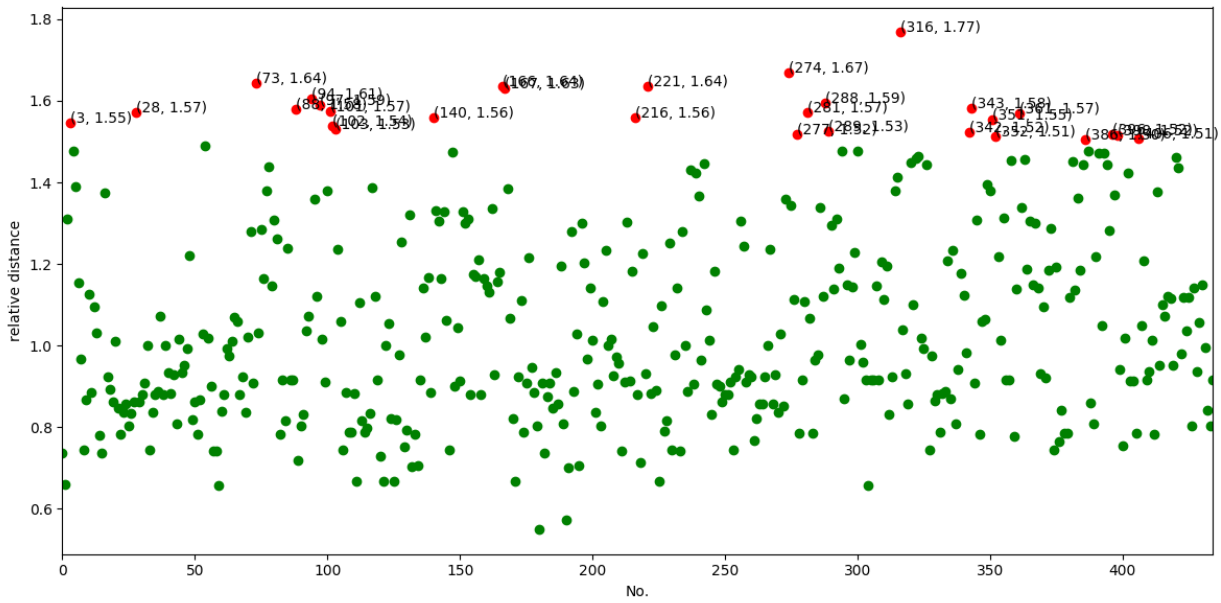
任务 3 异常检测

任务描述：针对上述两个任务中的某一个任务的数据，尝试使用某种异常检测方法分析数据的情 况，判断数据集合中是否存在异常数据点。

这里我们对任务一中的数据进行异常检测。检测方式采用通过 kmeans 算法聚类之后计算点集到中心的闵可夫斯基距离，我们再设置一个阈值，从而发现离群点。

我们以 1.5 作为离散点阈值。得到的离散点如下图。

```
discrete_points:
index
94      1.605116
101     1.574539  73      1.643841
102     1.537419  88      1.580234
140     1.558668  97      1.588489
167     1.631788  103     1.530758
216     1.559576  166     1.636089
221     1.635265  274     1.668727
288     1.594211  277     1.517723
342     1.521964  281     1.571684
352     1.512512  289     1.526849
386     1.504598  316     1.768405
396     1.518752  343     1.582014
398     1.515234  351     1.552653
3       1.546083  361     1.569323
28      1.571627  406     1.508436
```



但是从另一个方面也能看出，政界复杂，图中并没有很离散的点。若把阈值设为 2 则不存在离散点。因此我们可以认为任务一的数据中不存在异常数据点。

文件说明（readme）

1.py 文件分别对应第一题、第二题、第三题的全部代码

2.代码执行流程与修改过程详细记录于 jupyter 文件中，为方便读取，将该文件另存为 html 格式。