Cluster

2024 年 6 月 4 日

实验要求

1. 任务：对包含许多样本的数据集，将相似的样本进行合理聚类，将样本集化成多个簇，每个簇内的样本相似度较高.
   1.1. 实现聚类算法，实现上述的聚类.
   1.2. 找到合适的评估标准，对聚类结果进行评估.
2. 编程语言：Python 3.
3. 使用库：pandas, numpy.
4. 数据集：KDD Cup 1999 Data (KDD Cup 1999 Data (uci.edu)).

数据预处理

1. 对特征矩阵 X 的非数值列进行编码.
2. 将特征矩阵 X 归一化处理.
3. 取出 60% 的数据作为训练集，剩下 40% 作为验证集.
4. 由于数据分布不均衡，对训练集过采样.

normal.	count
DoS	3883370
Normal	972780
Probe	41102
R2L	1126
U2R	52

数据预处理

```python
def OverSample(X: pd.DataFrame, y: pd.Series, random_state=42):
    np.random.seed(random_state)
    unique, counts = np.unique(y, return_counts=True)
    MAX = counts.max()
    for i in range(unique.shape[0]):
        if counts[i] < MAX:
            idx = np.where(y == unique[i])[0]
            idx = np.random.choice(idx, (MAX - counts[i]) // 10)
            X = pd.concat([X, X.iloc[idx, :]], axis=0)
            y = pd.concat([y, y.iloc[idx]], axis=0)
    X = X.reset_index(drop=True)
    y = y.reset_index(drop=True)
    return X, y
```

数据预处理

normal.	count
DoS	2192451
Normal	712266
Probe	33187
R2L	1125
U2R	29

normal.	count
DoS	2192451
Normal	860284
Probe	249113
R2L	220257
U2R	219271

建立模型

K-means 参数

```
1 :param n_clusters:          聚类簇数
2 :param max_iter:            最大迭代次数
3 :param eps:                 精度
4 :param distance:            距离计算方式
5 :param random_state:        随机种子
6 :param Mu:                  聚类中心
7 :param label:               聚类标签
```

模型的输出与保存

Model 获得数据后训练出 n 个聚类中心以及其包含的数据.
每个中心的标签是它包含数据的众数.
以 DataFrame 的形式保存这些中心和标签到 csv 文件中.

模型评价

评价指标

Fowlkes-Mallows 指数：该指数通过比较真实标签和聚类算法的结果来评估聚类结果的准确性。具体而言，它考察了两两样本之间的匹配程度，即同时属于同一个簇的情况与同时不属于同一个簇的情况的比例。
Rand 指数：Rand 指数通过比较聚类结果中所有样本对的匹配情况来评估聚类算法的性能。它将样本对分为四类：同簇内的正样本、同簇内的负样本、不同簇的正样本和不同簇的负样本，然后计算一致性指标。
Jaccard 指数：Jaccard 指数通常用于评估分类算法的性能。它通过比较拟合预测结果和真实标签之间的重叠程度来评估分类结果的准确性。

带有标记的 K-means

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
必连约束集合 $\mathcal{M}$;
勿连约束集合 $\mathcal{C}$;
聚类簇数 k.

过程:

从 D 中随机选取 k 个样本作为初始均值向量$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$;
repeat
  $C_j = \varnothing \ (1 \leqslant j \leqslant k)$;
  for $i = 1, 2, \ldots, m$ **do**
    计算样本 $\boldsymbol{x}_i$ 与各均值向量 $\boldsymbol{\mu}_j \ (1 \leqslant j \leqslant k)$ 的距离: $d_{ij} = ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||_2$;
    $\mathcal{K} = \{1, 2, \ldots, k\}$;
    `is_merged=false`;
    while ¬ `is_merged` **do**
      基于 $\mathcal{K}$ 找出与样本 $\boldsymbol{x}_i$ 距离最近的簇: $r = \arg\min_{j \in \mathcal{K}} d_{ij}$;
      检测将 $\boldsymbol{x}_i$ 划入聚类簇 C_r 是否会违背 $\mathcal{M}$ 与 $\mathcal{C}$ 中的约束;
      if ¬ `is_voilated` **then**
        $C_r = C_r \bigcup \{\boldsymbol{x}_i\}$;
        `is_merged=true`
      else
        $\mathcal{K} = \mathcal{K} \setminus \{r\}$;
        if $\mathcal{K} = \varnothing$ **then**
          break并返回错误提示
        end if
      end if
    end while
  end for
  for $j = 1, 2, \ldots, k$ **do**
    $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\boldsymbol{x} \in C_j} \boldsymbol{x}$;
  end for
until 均值向量均未更新

输出: 簇划分 $\{C_1, C_2, \ldots, C_k\}$

训练代码

```
50          for n_iter in range(self.max_iter):
51              Mu_copy = Mu.copy()
52              C = [[] for _ in range(self.n_clusters)]
53              label = label_.copy()
54
55              for i in range(X.shape[0]):
56                  dist = np.array([self.distance(X[i], mu) for mu in Mu])
57                  for _ in range(self.n_clusters):
58                      r = np.argmin(dist)
59                      if not self.violated(mp[y[i]], r, label):
60                          C[r].append(i)
61                          label[r][mp[y[i]]] += 1
62                          break
63                      else:
64                          dist[r] = np.inf
65              for j in range(self.n_clusters):
66                  Mu[j] = np.mean(X[C[j]], axis=0)
67              if np.all(np.abs(Mu_copy - Mu) < self.eps):
68                  break
```

迭代过程

```
iter: 1
[[      0       0 1705940       0       0]
 [   4637       0       0       0       0]
 [      0       0  484848       0       0]
 [ 792977       0       0       0  237484]
 [      0       0       0       0   11629]
 [  62670  219271    1663  220257       0]]
iter: 2
[[      0       0 1780891       0       0]
 [  63016       0       0       0       0]
 [    110       0  411560       0       0]
 [ 684385       0       0       0  249113]
 [  41702       0       0       0       0]
 [  71071  219271       0  220257       0]]
```

```
iter: 14
[[      0       0 1781126       0       0]
 [  97604       0       0       0       0]
 [     67       0  411325       0       0]
 [ 655742       0       0       0       0]
 [  39174       0       0       0  249113]
 [  67697  219271       0  220257       0]]
iter: 15
[[      0       0 1781126       0       0]
 [  97604       0       0       0       0]
 [     67       0  411325       0       0]
 [ 655742       0       0       0       0]
 [  39174       0       0       0  249113]
 [  67697  219271       0  220257       0]]
```

各项指标

	FM	Rand	Jaccard
Normal	0.838398	0.702912	0.829821
DoS	0.88663	0.786113	0.878334
Probe	0.702892	0.494057	0.505496
R2L	0.0	1.0	0.0
U2R	0.955533	0.913043	0.0
All	0.883164	0.832228	0.867862

超参数的选取

本模型的主要超参数如下:

n, distance, random_state

由于测试时模型在 15 次左右的迭代后均可收敛, 所以此处不讨论 iter.

经过实验测试后发现, n=6, random_state=24 时具有较好的表现, distance 在选取曼哈顿距离与欧几里得距离时的差距并不大.

对于 n 选取 6 而不是 5 的一个可能理由如下:

DoS 块的分布呈现类似哑铃的形状, 而 K-means 算法出现的聚类簇是球形分布的, 两个球正好可以很好的拟合哑铃的形状.

存在的问题

1. K-means 可能发生大球吃小球的现象 (可以考虑混合高斯模型)
2. 对于文本数据的度量算法有缺陷 (可以考虑 VDM 距离)
3. 样本分布不平衡的问题仍为解决 (可以引入对干扰数据的考察)