原 标准化互信息NMI计算步骤及其Python实现

2017年10月28日 21:37:19 梦家 阅读数：3578 标签： 标准化互信息 NMI python 更多

> Excellence is a continuous process and not an accident.
> 卓越是一个持续的过程而不是一个偶然事件。

标准化互信息NMI计算步骤及其Python实现

> 标准化互信息NMI具体定义可以参考另一篇博客：
> https://smj2284672469.github.io/2017/10/27/community-detection-measures/#more
> 本文介绍其计算步骤和代码实现

假设对于17个样本点$(v_1, v_2, \ldots, v_{17})$进行聚类：

某一种算法得到聚类结果为：

A=[1 2 1 1 1 1 1 2 2 2 2 3 1 1 3 3 3]

标准的聚类结果为：

B=[1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3]

问题：需要度量算法结果与标准结果之间的相似度，如果结果越相似NMI值应接近1；如果算法结果很差则NMI值接近0。

根据公式计算MI的值其中X=unique(A)=[1 2 3] , Y=unique(B)=[1 2 3]：

$$MI(X,Y) = \sum_{i=1}^{|X|}\sum_{j=1}^{|Y|} P(i,j)log(\frac{P(i,j)}{P(i)P'(j)})$$

首先计算上式分子中联合概率分布$P(i,j) = \frac{|X_i \cap Y_j|}{N}$

$P(1,1) = 5/17, P(1,2) = 1/17, P(1,3) = 2/17$

$P(2,1) = 1/17, P(2,2) = 4/17, P(2,3) = 0$

$P(3,1) = 0, P(3,2) = 1/17, P(3,3) = 3/17$

再计算分母中概率函数$P(i) = X_i/N$，$P(i)$为i的概率分布函数，$P^{'}(j)$为j的概率分布函数：

对于$P(i)$：

$P(1) = 8/17, P(2) = 5/17, p(3) = 4/17$

对于$P(j)$：

$P^{'}(1) = 6/17, P^{'}(2) = 6/17, P^{'}(3) = 5/17$

根据以上计算可以计算出MI的值。

至于标准化互信息使用第二个公式计算：

$$NMI(X,Y) = \frac{2MI(X,Y)}{H(X)+H(Y)}$$

上式分母中$H(X), H(Y)$分别为X, Y的熵：

$$H(X) = -\sum_{i=1}^{|X|} P(i)log(P(i)); H(Y) = -\sum_{j=1}^{|Y|} P^{'}(j)log(P^{'}(j))$$

$$H(Y) = P^{'}(1)log_2(P^{'}(1)) + P^{'}(2)log_2(P^{'}(2)) + P^{'}(3)log_2(P^{'}(3))$$

综上则可以计算出NMI的值。

代码实现以上计算过程：

- 可以直接调用scikit-learn包中集成的度量函数
- 自己编写函数实现计算过程

Python代码实现如下(包含上述两种方式)：

```
# -*- coding:utf-8 -*-
'''
Created on 2017年10月28日

@summary: 利用Python实现NMI计算

@author: dreamhome
'''
import math
import numpy as np
from sklearn import metrics
def NMI(A,B):
    #样本点数
    total = len(A)
    A_ids = set(A)
    B_ids = set(B)
    #互信息计算
    MI = 0
    eps = 1.4e-45
    for idA in A_ids:
        for idB in B_ids:
            idAOccur = np.where(A==idA)
            idBOccur = np.where(B==idB)
            idABOccur = np.intersect1d(idAOccur,idBOccur)
            px = 1.0*len(idAOccur[0])/total
            py = 1.0*len(idBOccur[0])/total
            pxy = 1.0*len(idABOccur)/total
            MI = MI + pxy*math.log(pxy/(px*py)+eps,2)
    # 标准化互信息
    Hx = 0
    for idA in A_ids:
        idAOccurCount = 1.0*len(np.where(A==idA)[0])
        Hx = Hx - (idAOccurCount/total)*math.log(idAOccurCount/total+eps,2)
    Hy = 0
    for idB in B_ids:
        idBOccurCount = 1.0*len(np.where(B==idB)[0])
        Hy = Hy - (idBOccurCount/total)*math.log(idBOccurCount/total+eps,2)
    MIhat = 2.0*MI/(Hx+Hy)
    return MIhat

if __name__ == '__main__':
    A = np.array([1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3])
    B = np.array([1,2,1,1,1,1,1,2,2,2,2,3,1,1,3,3,3])
    print NMI(A,B)
    print metrics.normalized_mutual_info_score(A,B)
```

KMeans中
clf = KMeans(n_clusters=89)
s=clf.fit(weight)

'''
print(clf.labels_) #[, , ,]