实验内容

1、 数据来源 http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

2、使用 Apriori算法，支持度设为 30%，置信度为 90%，挖掘高置信度的规则

分析及设计

转换数据集

首先将数据集转换为课堂常见的形式，每一条记录中如果为赞同，那么记录该列的正索引，否则记录负索引，忽略?。其中第1列，将'republican'标记为正索引

获取频繁项集

1. 实现计算支持度的函数caluSupp，该函数通过遍历整个数据库和候选项集列表，累加每个候选集的出现次数。然后计算其支持度与支持度阈值比较，返回频繁项集和所有候选项集的支持度计数
2. [-17,-1],[1,17]构成了候选1-项集，利用上述函数计算频繁1-项集。
3. 将频繁1-项集中元素进行两两合并，得到候选2-项集，再利用1中函数计算得到频繁2-项集。同时扩展所有候选项集的支持度计数
4. 将频繁k-项集中元素前k-2个元素相同的项集进行两两合并得到候选k+1-项集，以支持度阈值过滤得到频繁k+1项集
5. 重复步骤4直到频繁k-项集为空

推导关联规则

1. 从频繁2-项集开始挖掘关联规则。对于频繁2-项集，只需要判断两两之间是否有关联关系，即对于集合{a,b}只需要计算S(a,b)/S(a)以及S(a,b)/S(b)即可
2. 对于其他频繁k-项集(k>2)，则需要递归判断。对于规则$A \rightarrow B$，其中A,B为集合，首先判断B大小为1的情况计算其置信度，并通过置信度阈值过滤。然后逐步增大B的大小(利用计算频繁项集中的合并函数)，再次计算置信度，并用阈值过滤。重复上述步骤直到B的大小=频繁k-项集大小-1时停止。返回得到的规则

详细实现

转换数据集

```
import pandas as pd
import joblib
dataset = pd.read_table("./house-votes-84.data",sep=',',header = None).values
a = []
for i in range(len(dataset)):
    tmp = []
    for j in range(len(dataset[i])):
        if dataset[i,j] == 'y' or dataset[i,j] == 'republican':
            tmp.append(j)
        elif dataset[i,j] == 'n' or dataset[i,j] == 'democrat':
            tmp.append(-(j+1))
    a.append(tmp)
joblib.dump(a,"MyDataset")
```

获取频繁项集

计算支持度

```
def caluSupp(dataset, Ck, minSupport):
    total = 435
    Ck = list(Ck)
    tmp = {}# tmp用来暂存支持度计数
    for tid in dataset:
        for can in Ck:
            if can.issubset(tid): # can是一个集合类型
                if can not in tmp.keys():# 第一次加入初始化为1
                    tmp[can] = 1
                else:   # 之后每次+1
                    tmp[can] += 1
    frequency = []
    supportData = {}
    for key in tmp:
        # 计算支持度
        support = tmp[key] / total
        if support >= minSupport:
            # 将满足阈值的放入频繁项集中
            frequency.append(key)
        # 记录所有的支持度
        supportData[key] = support
    return frequency, supportData
```

Apriori算法生成频繁项集

```
def apriori(dataSet, minSupport):
    C1 = map(frozenset,[[i] for i in range(17)])
    # 对每一行进行 set 转换，然后存放到集合中
    dataSet = list(map(set, dataSet))
    # print 'dataSet=', dataSet
    # 计算候选数据集 C1 在数据集 dataSet 中的支持度，并返回支持度大于 minSupport 的数据
    L1, supportData = caluSupp(dataSet, C1, minSupport)

    frequency = [L1] # frequency为所有频繁项集
    k = 2
    while (len(frequency[k - 2]) > 0):
```

```
        # 从候选2-项集开始计算，直到
        Ck = aprioriGen(frequency[k - 2],k) # 合并得到候选k项集

        Lk, supK = caluSupp(dataSet, Ck, minSupport)  # 计算频繁k-项集
        # 将新产生的项集的支持度计数加入到集合中
        supportData.update(supK)
        if len(Lk) == 0:
            break
        frequency.append(Lk)
        k += 1
    return frequency, supportData
```

上面用到的合并得到k项集函数

```
def aprioriGen(Lk, k):
    retList = []
    lenLk = len(Lk)

    for i in range(lenLk):
        for j in range(i + 1, lenLk):
            L1 = list(Lk[i])[: k - 2]
            L2 = list(Lk[j])[: k - 2]
            L1.sort()
            L2.sort()
            # 对于前k-2项相同的两个集合进行合并，计入到k项集中
            if L1 == L2:
                # print(Lk[i] | Lk[j], Lk[i], Lk[j])
                retList.append(Lk[i] | Lk[j])
    return retList
```

推导关联规则

计算置信度

```
def calcConf(freqSet, subSet, supportData, brl, minConf):
    # subSet 是集合的列表，包含本次要判断的规则的右部集合
    retList = []
    for conseq in subSet:
        conf = supportData[freqSet] / supportData[freqSet - conseq]
        # 支持度定义: a -> b = support(a | b) / support(a).
        if conf >= minConf:
            print(freqSet - conseq, '->', conseq, 'confidence:', conf)
            brl.append((freqSet - conseq, conseq, conf))
            retList.append(conseq)
    return retList
```

生成关联规则

```
def generateRules(L, supportData, minConf):
    ruleList = []
    for i in range(1, len(L)):# L[i]为所有i-1项集构成的列表
        for freqSet in L[i]:
            H1 = [frozenset([item]) for item in freqSet] # 频繁项集中所有元素形成的
列表
            if i > 1:# 2以上的项集，还需要递归的扩大关联规则的右部
                rulesFromConseq(freqSet, H1, supportData, ruleList, minConf)
            else:# 频繁2-项集，只需要计算两个置信度，所以单拿出来
                calcConf(freqSet, H1, supportData, ruleList, minConf)
    return ruleList
```

递归的合并规则

```
def rulesFromConseq(freqSet, H, supportData, brl, minConf):
    Hmp1 = calcConf(freqSet, H, supportData, brl, minConf)  # 利用阈值进行过滤，返回
可信度大于阈值的集合
    if (len(Hmp1) >= 1):
        m = len(H[0])
        if m < len(freqSet)-1: # 当关联规则的右部小于频繁项集的长度时
            Hmp1 = aprioriGen(H, m+1) # 利用合并函数将规则右部扩展一位
            # 如果仍有满足阈值的规则，那么递归的合并规则，并进行过滤，直到没有满足阈值要求的规
则
            rulesFromConseq(freqSet, Hmp1, supportData, brl, minConf)
```

实验结果

- 具体输出内容见out.txt
- 在支持度阈值0.3，置信度阈值0.9的情况下，共计挖掘到2990条规则。
- 结果文件的集合中数字为每个属性的索引，索引从1开始

心得体会

- 本次实验实现了Apriori算法生成频繁项集，以及关联规则的产生，并且利用置信度进行了剪枝。通过本次实验，我更加了解算法的流程，对于一些边界处的处理也有了实际体会。
- 起初忘记用set了，还在考虑将下标转字符串作为键。后来参考了《机器学习实战》慢慢理清了思路，以后还要多多看书，才能少走弯路。

17069130005

崔恩博