西安電子科技大學
XIDIAN UNIVERSITY

数据挖掘第二次上机实验报告

基于 Apriori 算法的国会投票关联规则挖掘

姓名：刘臻劼

学号：19030700016

班级：1903071

导师：郭杏莉

目录

1 实验内容

1.1 实验目标

本次实验目标为：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。

1.2 数据集介绍

Congressional Voting Records 数据集包含 435 条议员对于 16 项提案的投票记录（其中含 267 条民主党投票与 168 条共和党投票）。

每条记录包含 17 项，其中第一项为党派：republican 或 democrat；其余 16 项是对于各个议题的投票情况，取值为 y、n、?，分别表示支持、反对与弃权。

以下为各提案弃权票情况。

表 1: 各项提案的弃权票

提案	弃权票数
1.handicapped-infants	0
2.water-project-cost-sharing	12
3.adoption-of-the-budget-resolution	48
4.physician-fee-freeze	11
5.el-salvador-aid	11
6.religious-groups-in-schools	15
7.anti-satellite-test-ban	11
8.aid-to-nicaraguan-contras	14
9.mx-missile	15
10.immigration	22
11.synfuels-corporation-cutback	7
12.education-spending	21
13.superfund-right-to-sue	31
14.crime	25
15.duty-free-exports	17
16.export-administration-act-south-africa	28

2 实验分析

2.1 题目分析与数据预处理

（1）数据预处理

我们可将党派与各提案投票情况看做一个项，各项列举如下：

表 2: 各项提案的弃权票

项	项
I_1.Republican	I_2.Democrat
I_3.handicapped-infants=Y	I_4.handicapped-infants=N
I_5.water-project-cost-sharing=Y	I_6.water-project-cost-sharing=N
I_7.adoption-of-the-budget-resolution=Y	I_8.adoption-of-the-budget-resolution=N
I_9.physician-fee-freeze=Y	I_{10}.physician-fee-freeze=N
I_{11}.el-salvador-aid=Y	I_{12}.el-salvador-aid=N
I_{13}.religious-groups-in-schools=Y	I_{14}.religious-groups-in-schools=N
I_{15}.anti-satellite-test-ban=Y	I_{16}.anti-satellite-test-ban=N
I_{17}.aid-to-nicaraguan-contras=Y	I_{18}.aid-to-nicaraguan-contras=N
I_{19}.mx-missile=Y	I_{20}.mx-missile=N
I_{21}.immigration=Y	I_{22}.immigration=N
I_{23}.synfuels-corporation-cutback=Y	I_{24}.synfuels-corporation-cutback=N
I_{25}.education-spending=Y	I_{26}.education-spending=N
I_{27}.superfund-right-to-sue=Y	I_{28}.superfund-right-to-sue=N
I_{29}.crime=Y	I_30.crime=N
I_{31}.duty-free-exports=Y	I_{32}.duty-free-exports=N
I_{33}.export-administration-act-south-africa=Y	I_{34}.export-administration-act-south-africa=N

则有全项集 $I = \{I_1, \cdots, I_{34}\}$。则可将每项记录转换为事务，有事务集 $T = \{T_1, \cdots, T_{435}\}$。其中：$I(T_i) \subseteq I,\ T_i \in T$。

（2）挖掘目标鉴于强关联规则数量过多，此处考虑挖掘可解释性较好的两类关联规则：

1. 根据投票情况挖掘党派信息：形如 $X \Rightarrow Y$ 的关联规则，其中 $X \subseteq I - \{I_1, I_2\}$，$Y = I_1$ 或 $Y = I_2$；

2. 某党派内部的投票规则：形如 $X \Rightarrow Y$ 的关联规则，其中 $X, Y \subseteq I - \{I_1, I_2\}$，$X$ 包含且仅包含 I_1、I_2 中的一个；

2.2 先验原理与 Apriori 算法概述

对于频繁项集与其子集，我们有如下的先验原理：

一个项集是频繁的，则其所有子集是频繁的；一个项集是非频繁的，则其所有超集是非频繁的。

该原理启示我们可以在项集格上采用基于支持度的剪枝策略以降低候选项集的数量。由此我们有如下算法。记 C_k 为 k-候选集，F_k 为 k-频繁项集。

其中通过 (k-1)-频繁项集产生 k-候选项集的算法 $C_k\_Generator$ 采用 $F_{k-1} \times F_{k-1}$ 策略。

Algorithm 1 基于支持度剪枝的频繁项集产生算法（Apriori）

Output: 所有频繁项集的集合 F。

$F_1 = \{i | i \in I \wedge \sigma(i)/|I| \geqslant SUP_{min}\}, k = 2$
while $F_k \neq \Phi$ **do**
  $C_k = C_k\_Generator(F_{k-1})$
  for $t \in T$ **do**
    for $c \in C_k$ **do**
      if $issubset(c, t)$ **then**
        $\sigma(c) = \sigma(c) + 1$
      end if
    end for
  end for
  $F_k = \{c | c \in C_k \wedge \sigma(c)/|I| \geqslant SUP_{min}\}$
  $k = k + 1$
end while
$F = \bigcup F_k$
return F;

Algorithm 2 $C_k\_Generator$ 算法

Output: (k-1)-频繁项集 F_{k-1}。

$F_k = \{\}$
for $c_1 \in F_{k-1}$ **do**
  for $c_2 \in F_{k-1}$ **do**
    if $c_1^{(i)} = c_2^{(i)},\ i = 1, 2, \cdots, k-2 \wedge c_1^{k-1} \neq c_2^{k-2}$ **then**
      $F_k = F_k + \{c_1^{(i)}, c_2^{k-2}\},\ i = 1, 2, \cdots, k-1$
    end if
  end for
end forreturn F_k

3 具体实现

3.1 数据预处理

首先读取并加载数据：

```
SUP_min = 0.3 # 最小支持度
CONF_min = 0.9 # 最小置信度
# 加载数据, 构建事务集
def loadData():
    f = open('data/house-votes-84.data')
    data = []
    for txt_line in f.read().split('\n'):
        data_processed = []
        if txt_line == '':
            pass
        else:
            tmp = txt_line.split(',')
```

```
            data_processed.append(1) if tmp[0] == 'republican' else
                data_processed.append(2)
            for i in range(len(tmp)):
                if tmp[i] == 'y':
                    data_processed.append(2 * i + 1)
                elif tmp[i] == 'n':
                    data_processed.append(2 * i + 2)
                else:
                    pass
            data.append(data_processed)
    f.close()
    return data
raw_data = loadData()
```

根据任务分割数据集：

```
def split_data(raw_data, party):
    if party == '':
        return raw_data
    elif party == 'republic':
        data_republic = []
        for data in raw_data:
            if len(data) == 1:
                continue
            if data[0] == 1:
                data_republic.append(data[1:])
        return data_republic
    elif party == 'democrat':
        data_democrat = []
        for data in raw_data:
            if len(data) == 1:
                continue
            if data[0] == 2:
                data_democrat.append(data[1:])
        return data_democrat
#data_republic = split_data(raw_data, party='republic')
```

```
#data_democrat = split_data(raw_data, party='democrat')
data = split_data(raw_data, party='')
```

3.2 挖掘 1-频繁集与 2-频繁集

```
def compare(item1, item2):
    #判断两个(k-1)候选项集前(k-2)项是否相等
    tmp = []
    equal = True
    if len(item1) == 1:
        return equal, (item1 + item2)
    for i in range(len(item1) - 1):
        tmp.append(item1[i])
        if item1[i] != item2[i]:
            equal = False
            break
    if equal:
        minitem = min(item1[-1], item1[-1])
        maxitem = max(item2[-1], item1[-1])
        tmp.append(minitem)
        tmp.append(maxitem)
    return equal, tuple(tmp)

def issubset(item1, item2):
    # 判断 item1 是否为 item2 的子集
    return set(item1).issubset(set(item2))

#挖掘1-频繁集
def F1_Mining(data):
    C1, F1 = {}, {}
    count = len(data)
    for data_line in data:
        # 对于数据集中的每一行投票数据
        for i in range(len(data_line)):
            # 对于每一行数据中的下标（对应某个议题）
```

```
            key = tuple([data_line[i]])
            if key in C1.keys():
                # 以键值对的形式进行存储和计数
                C1[key] += 1
            else:
                C1[key] = 1
    for item in C1:
        if C1[item] / count >= SUP_min:
            F1[item] = C1[item]
    return F1

# 挖掘 2-频繁集
def F2_Mining(data, F1):
    C2, F2 = {}, {}
    F1_keys = sorted(list(F1.keys()))
    for i in range(len(F1_keys)):
        for j in range(i + 1, len(F1_keys)):
            equal, tmp = compare(F1_keys[i], F1_keys[j])
            if equal:
                C2[tmp] = 0
            else:
                continue
    for data_line in data:
        for item in C2:
            if issubset(item, data_line):
                C2[item] += 1
    for item in C2:
        if C2[item] / len(data) >= SUP_min:
            F2[item] = C2[item]
    return F2

F1 = F1_Mining(data)
F2 = F2_Mining(data, F1)
```

3.3 挖掘所有频繁集

```
# 挖掘3-频繁项集乃至n-频繁项集，并合并所有频繁项集
def F_Mining(data, F1, F2):
    Fk_1 = F2
    F = [F1]  # 存储的是所有频繁项集
    while Fk_1 != {}:
        F.append(Fk_1)
        Ck, Fk = {}, {}
        Fk_1_keys = sorted(list(Fk_1.keys()))
        for i in range(len(Fk_1_keys)):
            for j in range(i + 1, len(Fk_1_keys)):
                equal, tmp = compare(Fk_1_keys[i], Fk_1_keys[j])
                if equal:
                    Ck[tmp] = 0
                else:
                    continue
        for data_line in data:
            for item in Ck:
                if issubset(item, data_line):
                    Ck[item] += 1
        for item in Ck:
            if Ck[item] / len(data) >= SUP_min:
                Fk[item] = Ck[item]
        Fk_1 = Fk
    return F

F = F_Mining(data, F1, F2)
```

3.4 生成第一类关联规则

```
def generate_1_rules(F):
    rules = {}
    for i in range(2, len(F)):
        for item in F[i]:
            if 1 in item:
```

```
                antecedent = tuple(sorted(set(item) - {1}))  # 关联规则
                    前件
                confidence = F[i][item] / F[i - 1][antecedent]
                if 1 > confidence >= CONF_min:
                    rules[(antecedent, 1)] = confidence
            elif 2 in item:
                antecedent = tuple(sorted(set(item) - {2}))  # 关联规则
                    前件
                confidence = F[i][item] / F[i - 1][antecedent]
                if 1 > confidence >= CONF_min:
                    rules[(antecedent, 2)] = confidence
            else:
                continue
    return rules

rules = generate_1_rules(F).items()
rules = sorted(rules, key=lambda x: x[1], reverse=True)
```

3.5 生成第二类关联规则

```
def get_subset(freq):
    # 获得某一元组的所有子集
    if len(freq) == 0:
        return [[]]
    all_subsets = get_subset(freq[1:]) + [[freq[0]] + r for r in
        get_subset(freq[1:])]
    all_subsets = sorted(all_subsets, key=lambda x: (len(x), x))
    return all_subsets

def subset_tuple(all_subsets, freq):
    # 整理子集
    all_subsets_tuple = []
    for subset in all_subsets:
        if len(subset) == 0 or len(subset) == len(freq):
            continue
```

```
        else:
            subset = set(subset)
            complement = set(list(freq)) - subset
            subset = tuple(sorted(subset, key=lambda x: x))
            complement = tuple(sorted(complement, key=lambda x: x))
            all_subsets_tuple.append(tuple([subset, complement]))
    return all_subsets_tuple

def generate_2_rules(F, party=party):
    # 生成第二类关联规则
    rules = {}
    for i in range(1, len(F)):
        for item in F[i]:
            for subset in subset_tuple(get_subset(item), item):
                rule_confidence = F[len(item) - 1][item] / F[len(subset
                    [0]) - 1][subset[0]]
                if rule_confidence >= CONF_min:
                    rules[subset] = rule_confidence
    return rules
rules = sorted(generate_2_rules(F).items(), key=lambda x: x[1], reverse
    =True)
```

4 实验结果

4.1 第一类关联规则

第一类关联规则即根据投票情况挖掘党派信息，共挖掘出 202 条关联规则。高置信度的规则如下（部分）：

```
physician-fee-freeze=N, aid-to-nicaraguan-contras=Y,  -> Democrat, 置信度:0.99526
physician-fee-freeze=N, education-spending=N,  -> Democrat, 置信度:0.99505
physician-fee-freeze=N, el-salvador-aid=N,  -> Democrat, 置信度:0.99487
physician-fee-freeze=N, el-salvador-aid=N, aid-to-nicaraguan-contras=Y,  -> Democrat, 置信度:0.99479
physician-fee-freeze=N, anti-satellite-test-ban=Y, aid-to-nicaraguan-contras=Y,  -> Democrat, 置信度:0.99459
physician-fee-freeze=N, aid-to-nicaraguan-contras=Y, education-spending=N,  -> Democrat, 置信度:0.99448
physician-fee-freeze=N, mx-missile=Y,  -> Democrat, 置信度:0.99444
physician-fee-freeze=N, el-salvador-aid=N, anti-satellite-test-ban=Y,  -> Democrat, 置信度:0.99438
physician-fee-freeze=N, el-salvador-aid=N, anti-satellite-test-ban=Y, aid-to-nicaraguan-contras=Y,  -> Democrat, 置信度:0.99429
physician-fee-freeze=N, aid-to-nicaraguan-contras=Y, mx-missile=Y,  -> Democrat, 置信度:0.99425
```

4.2 第二类关联规则

第一类关联规则即党派内部投票规律。共和党内部分高置信度的投票规律如下：

```
关联规则:water-project-cost-sharing=Y,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:water-project-cost-sharing=N,  -> crime=Y, 置信度:1.00000
关联规则:handicapped-infants=N, water-project-cost-sharing=Y,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:handicapped-infants=N, water-project-cost-sharing=N,  -> crime=Y, 置信度:1.00000
关联规则:handicapped-infants=N, anti-satellite-test-ban=N,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:handicapped-infants=N, immigration=Y,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:handicapped-infants=N, education-spending=Y,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:handicapped-infants=N, superfund-right-to-sue=Y,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:handicapped-infants=N, duty-free-exports=N,  -> physician-fee-freeze=Y, 置信度:1.00000
关联规则:water-project-cost-sharing=Y, adoption-of-the-budget-resolution=N,  -> physician-fee-freeze=Y, 置信度:1.00000
```

民主党内部分高置信度的投票规律如下：

```
关联规则:handicapped-infants=Y, synfuels-corporation-cutback=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:water-project-cost-sharing=N, el-salvador-aid=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:water-project-cost-sharing=N, anti-satellite-test-ban=Y,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:water-project-cost-sharing=N, aid-to-nicaraguan-contras=Y,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:water-project-cost-sharing=N, superfund-right-to-sue=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:water-project-cost-sharing=N, export-administration-act-south-africa=Y,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:el-salvador-aid=N, synfuels-corporation-cutback=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:religious-groups-in-schools=N, synfuels-corporation-cutback=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:anti-satellite-test-ban=Y, synfuels-corporation-cutback=N,  -> physician-fee-freeze=N, 置信度:1.00000
关联规则:aid-to-nicaraguan-contras=Y, synfuels-corporation-cutback=N,  -> physician-fee-freeze=N, 置信度:1.00000
```

注意到前 10 的关联规则置信度均为 1，这是由于数据集较小，党派内在某几项议题上的投票情况完全相同。

5 结语与心得

本实验中统计候选项集的支持度时还是暴力遍历方法，没有使用教材上的哈希树等方法，算是偷了点小懒，今后应当更加努力，尽可能优化算法。