COSI 126A Homework 0

Jie Tang

Abstract

This is Homework 0 of Intro to Data Mining class

01/21/2020

Jietang@brandeis.edu

Problem 1 (9 points)

Discuss whether or not each of the following activities is a data mining task.

(A) Dividing the customers of a company according to their gender.
No. A simple SQL query can implement this work.

(B) Dividing the customers of a company according to their profitability.
No. This is an accounting calculation. Thus it's not data mining.

(C) Computing the total sales of company.
No. This is an accounting calculation. This is just a computation and didn't mine anything.

(D) Sorting a student database based on student identification numbers.
No. A simple SQL query can implement this work as well. Thus it's not data mining.

(E) Predicting the outcomes of tossing a fair pair of dice.
No. Since the die is fair, this is a probability calculation, not data mining.

(F) Predicting the future stock price of a company using historical records.
Yes. We would attempt to create a model that can predict the continuous value of the stock price.

(G) Monitoring the heart rate of a patient for abnormalities.
Yes. It is detecting abnormalities, which is a part of data mining.

(H) Monitoring seismic waves for earthquake activities.
Yes. It needs to detect when the earthquake is coming and needs data and standard to do this kind of work.

(I) Extracting the frequencies of a sound wave.
No. This action is more about physics.

Problem 2 (10 points)

Suppose that you are employed as a data mining consultant for an Internet search engine company. Describe how data mining can help the company by giving specific examples of how techniques, such as clustering, classification, association rule mining, and anomaly detection can be applied.

Answer:

The Internet search engine company can benefit a lot from data mining. For example, the clustering can help them group different types of news or information. This can improve their effeciency. Moreover, classification and association rule mining can help them classify customers from which they can wisely recommend staffs for their customers. For example, if a person ususally search some snacks on the engine, then we can recommend some delicious snacks to them, and they will tend to buy those snacks.

Problem 3 (10 points)

For each of the following data sets, explain whether or not data privacy is an important issue.

(A) Census data collected from 1900-1950.
Answer: No. Because the data is out of date.

(B) IP addresses and visit times of Web users who visit your Website.
Answer: Yes. We can locate people by IP addresses.

(C) Images from Earth-orbiting satellites.
Answer: No. Because those images are public.

(D) Names and addresses of people from the telephone book.
Answer: No. Because the data is public.

(E) Names and email addresses collected from the Web.
Answer: No. Because the data is public.

Problem 4 (15 points)

Matrix A = $\begin{matrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{matrix}$ calculate A^{-1} , A^{+}, A^{100}

Answer:

$$A = \begin{matrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{matrix} \sim \begin{matrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

A^{-1} doesn't exist because A is a singular matirx.

$$A^{+} = \begin{matrix} -0.6389 & -0.1667 & 0.3056 \\ -0.0556 & 0 & 0.0556 \\ 0.5278 & 0.1667 & -0.1944 \end{matrix}$$

$$A^{100} = \begin{matrix} 1 * 14^{99} & 2 * 14^{99} & 3 * 14^{99} \\ 2 * 14^{99} & 4 * 14^{99} & 6 * 14^{99} \\ 3 * 14^{99} & 6 * 14^{99} & 9 * 14^{99} \end{matrix}$$

Problem 5 (14 points)

Assume there three students, X, Y , Z. Only one of them gets a score A+. X asks Teacher if he gets A+. Teacher refuses to tell X his score. Instead, Teacher says that Y does not get A+. Calculate P(Z gets A+)

Answer:

This is a question like 3-door question. Firstly, we random choose a door, and the P(the door you choose doesn't contain the gift) = 2/3, when the host tells you that one of the remaining doors doesn't contain the gift. And then P(last door contains gift) = 2/3*1 =2/3.
This question is the same as the question I discussed above. Before the professor tell X that Y doesn't get A, P(X don't get A+) is 2/3. After the professor tell X that Y doesn't get A. P(Z gets A+) = 2/3*1 = 2/3. As a result:
P(Z gets A+) = 2/3*1 = 2/3.

Problem 6 (14 points)

There are two kinds of products in a warehouse, A and B. The percentage of A is 70%, B is 30%. The probability of substandard products in A is P(A = sub) = 2.5%, for B, it's P(B = sub) = 5%. Warehouse tests 4 products and one of them is substandard. What is the probability that this product is from A, P(this sub from A)

Answer:

Usually, we can use Bayes and conditional probability to solve this problem. However, in this case, we only care about the substandard items, also taking products from the warehouse has no effect on P(A = sub), P(B = sub), the percentage of A, and the percentage of B. We surprisingly found that they are independent. As a result:
P(this sub from A) = (70%*2.5%)/(70%*2.5%+30*5%) = 0.5385.

Problem 7 (14 points)

Calculate the similarity matrix between 9 planets. The data of planets is in Table 1.
You can use s(p1, p2) = $\sqrt{a0(d1-d2)^2 + a1(r1-r2)^2 + a2(m1-m2)^2}$, a0 = 3.5 ∗ 10^−7, a1 = 1.6 ∗ 10^−5, a2 = 1.1 ∗ 10^−27.
Set a threshold to separate 9 planets into different groups. What is the relationship between threshold and groups.

Answer:
For this question, I did it in Python by Scipy package. And I set the mean of total distance as a threshold.

```
from scipy.spatial.distance import euclidean, pdist, squareform
import math
var =  pd.DataFrame([['Jupiter', 778000, 71492, 1.90e27],
                    ['Saturn',1429000, 60268, 5.69e26],
                    ['Uranus',2870990, 25559, 8.69e25],
                    ['Neptune', 4504300, 24764, 1.02e26],
                    ['Earth', 149600, 6378, 5.98e24],
                    ['Venus', 108200, 6052, 4.87e24],
                    ['Mars', 227940, 3398, 6.42e23],
                    ['Mercury', 57910, 2439, 3.30e23],
                    ['Pluto', 5913520, 1160, 1.32e22]],
                   columns=['p', 'd','r','m']).set_index('p')
var
```

	d	r	m
p			
Jupiter	778000	71492	1.900000e+27
Saturn	1429000	60268	5.690000e+26
Uranus	2870990	25559	8.690000e+25
Neptune	4504300	24764	1.020000e+26
Earth	149600	6378	5.980000e+24
Venus	108200	6052	4.870000e+24
Mars	227940	3398	6.420000e+23
Mercury	57910	2439	3.300000e+23
Pluto	5913520	1160	1.320000e+22

```
a0 = 3.5e-7
a1 = 1.6e-5
a2 = 1.1e-22
coef = np.array([a0,a1,a2])
dists  = pdist(var, lambda u, v: math.sqrt((((u-v)**2)*coef).sum()))
square_dists = pd.DataFrame(squareform(dists),columns=var.index, index = var.index)
square_dists
```

p	Jupiter	Saturn	Uranus	Neptune	Earth	Venus	Mars	Mercury	Pluto
p									
Jupiter	0.000000e+00	1.395965e+16	1.901595e+16	1.885758e+16	1.986465e+16	1.987629e+16	1.992063e+16	1.992391e+16	1.992723e+16
Saturn	1.395965e+16	0.000000e+00	5.056307e+15	4.897937e+15	5.905004e+15	5.916645e+15	5.960989e+15	5.964261e+15	5.967584e+15
Uranus	1.901595e+16	5.056307e+15	0.000000e+00	1.583701e+14	8.486961e+14	8.603379e+14	9.046815e+14	9.079538e+14	9.112764e+14
Neptune	1.885758e+16	4.897937e+15	1.583701e+14	0.000000e+00	1.007066e+15	1.018708e+15	1.063052e+15	1.066324e+15	1.069647e+15
Earth	1.986465e+16	5.905004e+15	8.486961e+14	1.007066e+15	0.000000e+00	1.164178e+13	5.598542e+13	5.925770e+13	6.258033e+13
Venus	1.987629e+16	5.916645e+15	8.603379e+14	1.018708e+15	1.164178e+13	0.000000e+00	4.434364e+13	4.761592e+13	5.093855e+13
Mars	1.992063e+16	5.960989e+15	9.046815e+14	1.063052e+15	5.598542e+13	4.434364e+13	0.000000e+00	3.272284e+12	6.594910e+12
Mercury	1.992391e+16	5.964261e+15	9.079538e+14	1.066324e+15	5.925770e+13	4.761592e+13	3.272284e+12	0.000000e+00	3.322626e+12
Pluto	1.992723e+16	5.967584e+15	9.112764e+14	1.069647e+15	6.258033e+13	5.093855e+13	6.594910e+12	3.322626e+12	0.000000e+00

```
square_dists > dists.mean()
```

p	Jupiter	Saturn	Uranus	Neptune	Earth	Venus	Mars	Mercury	Pluto
p									
Jupiter	False	True	True	True	True	True	True	True	True
Saturn	True	False	False	False	True	True	True	True	True
Uranus	True	False	False	False	False	False	False	False	False
Neptune	True	False	False	False	False	False	False	False	False
Earth	True	True	False	False	False	False	False	False	False
Venus	True	True	False	False	False	False	False	False	False
Mars	True	True	False	False	False	False	False	False	False
Mercury	True	True	False	False	False	False	False	False	False
Pluto	True	True	False	False	False	False	False	False	False

From the result above, I set the mean of total distance as a threshold. Because mean distance can be a center point of all of those planets (based on distance). If most of a planet's distances to other planets are larger than mean distance, then we divide those into a group, otherwise, another group. In this way they have some similarity. Thus, Group1: Jupiter, Saturn. Group2: Uranus, Neptune, Earth, Venus, Mar, Mercury, Pluto.

Problem 8 (14 points)

Given N documents. Write a Python program to find the most frequent
1. < word >
2. < word1, word2 >
3. < word1, word2, word3 >

e.g. D1 = {aa aa a aaa}, D2 = {aa aa aaa}, D3 = {aaa}, most frequent < word > is < aaa > whose frequency is 3, < word1, word2 > is < aa, aaa > whose frequency is 2, < word1, word2, word3 > is < a, aa, aaa > whose frequency is 1.

Answer:
Discussion: In this case, because there are so many documents and words, if we just count and do combinations of words, there are so many situations and our computers will definitely crash. So, I figure out an algorithm which create a word dictionary based on all the words occur in all documents. In the end, I got a word dictionary which contains more than 2000 words. And then, we use the created dictionary to create a vector for every document whose length is the same as the length of the dictionary. When a word in the dictionary appear in a document, we let it be 1, if not, we let it be 0. And we put all the 93 arrays into a data frame. The column names are the words, while the row names are the number of documents. For one word condition, we can count all the times in each columns of each word. For the 2-word condition, we can pair each word and count the times. For the 3-word condition, it's the same as the 2-word condition.

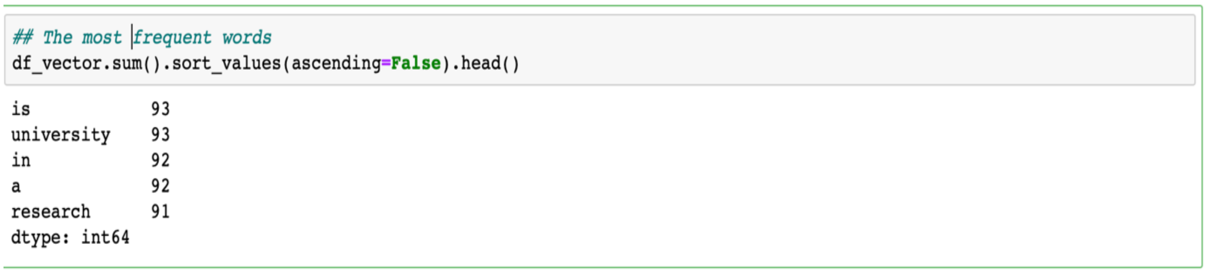
```python
## loading packages
import os
import numpy as np
import pandas as pd
import re
from string import digits
from nltk.tokenize import word_tokenize
from collections import Counter

## loading all the file names into a list
path = '/Users/tjmask/Desktop/Semester 2/Course/Data Mining/HW0/docs/'
fileList = os.listdir(path)
print("There are %d documents in total" %len(fileList))
print(path+fileList[1])

## reading all the files in the 'fileList[i]' fromat
for i in range(93):
    fileList[i] = open(path+fileList[i],"r")


## put all the files into a list
doc = []
for i in range(93):
    d = []
    for j in fileList[i]:
        d.append(j.lower())
    doc.append(d)


## deleting all specail characters
for i in range(93):
    doc[i] = re.sub('[\s+\.\!|/_,$%^&*(+\"\')]+|[+--()?[]“”! ? 。 、 ]+', ' ', doc[i][0])
    remove_digits = str.maketrans('','',digits)
    doc[i] = doc[i].translate(remove_digits).lower()
    doc[i] = doc[i].split()
```

```python
## getting the unique word in a set
word_set = set()
for i in range(93):
    word_set =  word_set.union(doc[i])

## getting the word dictionary in a set
word_dict ={}
for index, word in enumerate(list(word_set)):
    word_dict[word] = index


## getting every document and transform it into 0 & 1 based on word_dict
doc_vector =[]
for j in range(93):
    X = {}
    for i in word_dict.keys():
        if i in doc[j]:
            X[i] = 1
        else:
            X[i] = 0
    word_vector.append(X)
```

```python
## have a grasp of the data frame
df_vector = pd.DataFrame(word_vector)
df_vector.head()
```

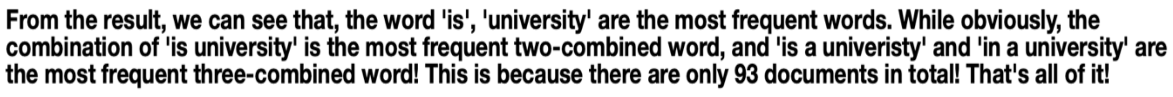
	#	-	-acre	;	a	aau	abbreviated	abel	able	abolitionist	...	year	year-round	years	yerkes	yeshiva	york	youngest	zanvyl	–	—of
0	0	0	0	0	1	0	0	0	0	0	...	1	0	0	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 2376 columns



```
## The most frequent words
df_vector.sum().sort_values(ascending=False).head()
```

```
is            93
university    93
in            92
a             92
research      91
dtype: int64
```



From the result, we can see that, the word 'is', 'university' are the most frequent words. While obviously, the combination of 'is university' is the most frequent two-combined word, and 'is a univeristy' and 'in a university' are the most frequent three-combined word! This is because there are only 93 documents in total! That's all of it!

From the result, we can see that, the word < is > and < university > are the most frequent words whose frequency is 93. While obviously, the combination of < is, university > is the most frequent two-combined word whose frequency is 93 as well. And < is a university > and < in, a, university > are the most frequent three-combined word whose frequency is 92! This is because there are only 93 documents in total! That's all of it!