VECTORS REPORT

School or Department: Wu Han University

Grade and Specialty: Excellent engineer 2016

Name: Guo Yang, Kong Chuishun, Tang Rui

Advisor: Fangling Pu

March 18 2018

【Abstract】: The purpose of the lab is to apply the vector to a small lab and learn how to use the vector to solve the problem through the python. The lab is as following: In this lab, we will represent a US senator's voting record as a vector over R and will use dot-products to compare voting records. For this lab, we will just use a list to represent a vector where each element of that vector represents how that senator voted on a given piece of legislation. By looking at the difference between the "voting vectors" of two senators, we can dispel the fog of politics and see just where our representatives stand. The data file is in the appendix.
【Key words】 Python; vectors; voting

Table of Contents

Chapter 1 Introduction of Tasks

We would like to determine just how like-minded two given senators are. We will use the dot-product of vectors u and v to judge how often two senators are in agreement. We would like to solve the following tasks.

Task 1: Write a procedure create voting dict(strlist) that, given a list of strings (voting records from the source file), returns a dictionary that maps the last name of a senator to a list of numbers representing that senator's voting record. You will need to use the built-in procedure int (·) to convert a string representation of an integer (e.g. '1') to the actual integer (e.g. 1).

Task 2: Write a procedure policy compare (sen a, sen b, voting dict) that, given two names of senators and a dictionary mapping senator names to lists representing voting records, returns the dot-product representing the degree of similarity between two senators' voting policies.

Task 3: Write a procedure most similar (sen, voting dict) that, given the name of a senator and a dictionary mapping senator names to lists representing voting records, returns the name of the senator whose political mindset is most like the input senator (excluding, of course, the input senator him/herself).

Task 4: Write a very similar procedure least similar (sen, voting dict) that returns the name of the senator whose voting record agrees the least with the senator whose name is sen.

Task 5: Use these procedures to figure out which senator is most like Rhode Island legend Lincoln Chafee. Then use these procedures to see who disagrees most with Pennsylvania's Rick Santorum. Give their names.

Task 6: How similar are the voting records of the two senators from your favorite state?

Task 7: Write a procedure find average similarity(sen, sen set, voting dict) that, given the name sen of a senator, compares that senator's voting record to the voting records of all senators whose names are in sen set, computing a dot-product for each, and then returns the average dot-product. Use your procedure to compute which senator has the greatest average similarity with the set of Democrats (you can extract this set from the input file).

Chapter 2 Solutions

Each line of the file represents the voting record of a different senator. In case we have forgotten how to read in the file, we do it like this:

```
>>> f = open('voting_record_dump109.txt') >>> mylist = list(f)
```

We split each line of the file into a list; the first element of the list will be the senator's name, the second will be his/her party affiliation (R or D), the third will be his/her home state, and the remaining elements of the list will be that senator's voting record on a collection of bills. A "1" represents a 'yea' vote, a "-1" a 'nay', and a "0" an abstention.

Suppose u and v are two vectors. Let's take the simple case (relevant to the current lab) in which the entries are all 1, 0, or -1. Recall that the dot-product of u and v is defined as

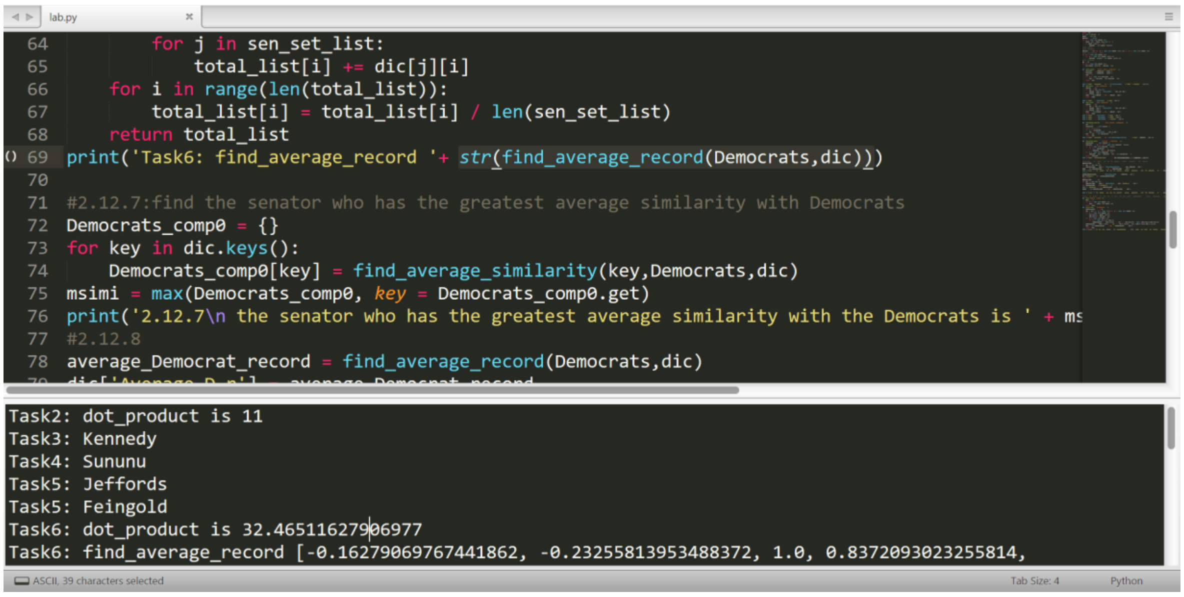
$$u * v = \sum_{k} u[k]v[k]$$

Consider the k^{th} entry. If both u[k] and v[k] are 1, the corresponding term in the sum is 1. If both u[k] and v[k] are -1, the corresponding term in the sum is also 1. Thus a term in the sum that is 1 indicates agreement. If, on the other hand, u[k] and v[k] have different signs, the corresponding term is -1. Thus a term in the sum that is -1 indicates disagreement. (If one or both of u[k] and v[k] are zero then the term is zero, reflecting the fact that those entries provide no evidence of either agreement or disagreement.) The dot-product of u and v therefore is a measure of how much u and v are in agreement.

In the last task, we compare each senator's record to the voting record of each Democrat senator. Next we see that there is a computational shortcut, based on an algebraic property of the dot-product: the distributive property:

$$(v1 + v2)\cdot x = v1 \cdot x + v2 \cdot x$$

Chapter 3 Results and Conclusion



```
        for j in sen_set_list:
            total_list[i] += dic[j][i]
    for i in range(len(total_list)):
        total_list[i] = total_list[i] / len(sen_set_list)
    return total_list
print('Task6: find_average_record '+ str(find_average_record(Democrats,dic)))

#2.12.7:find the senator who has the greatest average similarity with Democrats
Democrats_comp0 = {}
for key in dic.keys():
    Democrats_comp0[key] = find_average_similarity(key,Democrats,dic)
msimi = max(Democrats_comp0, key = Democrats_comp0.get)
print('2.12.7\n the senator who has the greatest average similarity with the Democrats is ' + ms
#2.12.8
average_Democrat_record = find_average_record(Democrats,dic)
```

```
Task2: dot_product is 11
Task3: Kennedy
Task4: Sununu
Task5: Jeffords
Task5: Feingold
Task6: dot_product is 32.46511627906977
Task6: find_average_record [-0.16279069767441862, -0.23255813953488372, 1.0, 0.8372093023255814,
```

Figure 1

The result of the program running is shown in the figure 1. Because the data of the total list is very large, the result of tasks1 is not shown in the figure. Other five answers are all given. The whole coding is in the appendix and the .python file is attached in the Compressed package.

Bibliography

[1]Coding the Matrix Linear Algebra through Applications to Computer Science, Edition 1, PHILIP N. KLEIN, Brown University

Appendix

```
import sys
f = open('vote.txt')
mylist = list(f)
Democrats = set()
for i in range(len(mylist)):
    mylist[i] = mylist[i].split(' ')
    if mylist[i][1] == 'D':
        Democrats.add(mylist[i][0])
#print(mylist)
#print(Democrats)
vote_list = [[0 for j in range(len(mylist[0])-3)] for i in range(len(mylist))]
#print(len(vote_list[0]))
for i in range(len(mylist)):
    for j in range(len(mylist[i])-3):
        vote_list[i][j] = int(mylist[i][j+3])
#print(vote_list)
dic = {}
for i in range(len(mylist)):
    dic[mylist[i][0]] = vote_list[i]

def policy_compare(sen_a,sen_b,voting_dict):
    sen_a_list = voting_dict[sen_a]
    sen_b_list = voting_dict[sen_b]
    simi = 0
    for i in range(len(sen_a_list)):
        simi += sen_a_list[i]*sen_b_list[i]
    return simi

print('Task2: dot_product is '+ str(policy_compare('Akaka','Alexander',dic)))
def most_similar(sen,voting_dict):
    comp_dic = {}
    for key in dic.keys():
        comp_dic[key] = policy_compare(key,sen,dic)
    del comp_dic[sen]
    msimi = max(comp_dic, key = comp_dic.get)
    return msimi

print('Task3: '+ most_similar('Akaka',dic))
def least_similar(sen,voting_dict):
    comp_dic = {}
    for key in dic.keys():
        comp_dic[key] = policy_compare(key,sen,dic)
    del comp_dic[sen]
    msimi = min(comp_dic, key = comp_dic.get)
    return msimi
print('Task4: '+ least_similar('Akaka',dic))
print('Task5: ' + most_similar('Chafee',dic))
print('Task5: ' + least_similar('Santorum',dic))

def find_average_similarity(sen,sen_set,voting_dict):
    sum_ = 0
    sen_set_list = list(sen_set)
    #print(sen_set_list)
    for i in sen_set_list:
        sum_ += policy_compare(sen,i,dic)
    aver = sum_ / len(sen_set_list)
    return aver
print('Task6:                              dot_product                              is
'+str(find_average_similarity('Akaka',Democrats,dic)))

def find_average_record(sen_set, voting_dict):
    sen_set_list = list(sen_set)
    total_list = [0 for i in range(len(vote_list[0]))]
    for i in range(len(vote_list[0])):
        for j in sen_set_list:
            total_list[i] += dic[j][i]
    for i in range(len(total_list)):
        total_list[i] = total_list[i] / len(sen_set_list)
    return total_list
print('Task6: find_average_record '+ str(find_average_record(Democrats,dic)))

#2.12.7:find the senator who has the greatest average similarity with Democrats
Democrats_comp0 = {}
for key in dic.keys():
    Democrats_comp0[key] = find_average_similarity(key,Democrats,dic)
msimi = max(Democrats_comp0, key = Democrats_comp0.get)
print('2.12.7\n the senator who has the greatest average similarity with the
Democrats is ' + msimi)
#2.12.8
average_Democrat_record = find_average_record(Democrats,dic)
dic['Average_D_r'] = average_Democrat_record
#print(dic)
```

```
Democrats_comp1 = {}
for key in dic.keys():
    Democrats_comp1[key] = policy_compare(key,'Average_D_r',dic)
del Democrats_comp1['Average_D_r' ]
print( 'Average_D_r' in Democrats_comp1)
msimi0 = max(Democrats_comp1, key = Democrats_comp1.get)

print('2.12.8\n the senator who has the greatest average similarity with the
Democrats is ' + msimi0)

#2.12.9 use obvious way (too lazy to write the fast matrix multiplication without
third-party modelus)
def dot_p(list_a,list_b):
    dot_p = 0
    for i in range(len(list_a)):
        dot_p += list_a[i]*list_b[i]
    return dot_p
def bitter_rivals(voting_dict):
    obvious_way_dic = {}
    tmp = [[0 for i in range(2)] for j in range(len(mylist))]
    for i in range(len(mylist)):
        tmp[i][0] = mylist[i][0]
        tmp[i][1] = vote_list[i]
    for i in range(len(tmp)-1):
        for j in range(i+1,len(tmp)):
            obvious_way_dic[tmp[i][0] + ' and ' + tmp[j][0]] =
dot_p(tmp[i][1],tmp[j][1])
    obvious_way_dic[tmp[-2][0] + ' and ' + tmp[-1][0]] =
dot_p(tmp[-2][1],tmp[-1][1])
    mdis = min(obvious_way_dic, key = obvious_way_dic.get)
```

```
        return mdis
    print('2.12.9\n Im not sure because the lexicographical order sucks but these two
senators disagree most according to my program: ',bitter_rivals(dic))
```