机器学习学习报告

周珊琳

上海电力大学
上海, 2019-07-04, 中国

第一节　算法概述

算法名称：PageRank

算法原理：PageRank又称网页排名、谷歌左侧排名，是一种由搜索引擎根据网页之间相互的超链接计算的技术，而作为网页排名的要素之一，以Google公司创办人拉里·佩奇（Larry Page）之姓来命名。 Google用它来体现网页的相关性和重要性，在搜索引擎优化操作中是经常被用来评估网页优化的成效因素之一。

两个假设：数量假设（一个页面越被其他页面链接，说明他越重要（ps：难怪好多技术博客的都互相链接）；质量假设（越是被高质量页面链接，说明该页面越重要（ps：最好能被大博主推荐一波，粉丝蹭蹭蹭往上涨）

两个问题：Dead Ends；Spider Traps。

Dead Ends：在几个网页的节点之间跳转，经过一段很长的时间之后，没有出路了，连几点n这个网页也不能访问。解决方法：判断网页节点矩阵M中是否有一列全部是0，如果有，则将这一列的值全部替换成1/n。

Spider Traps（蛛网陷阱）：在几个网页的节点之间跳转，经过一段很长的时间之后，只能在节点n来回跳转（也就是说不嫩访问到其他的网页，只能点击访问节点n这个网页）。解决方法：在访问节点n的几率接近于1的时候，让他随机的跳转到任意一个网页（唉网页可以不在这些几点网页中）。

PageRank根据假设的原始公式：

$$PR(A)i+1=\sum_{i=0}^{n}\frac{PR(Ti)}{L(Ti)}\text{[1]} \tag{1.1}$$

[1]i:循环次数
Ti：其他节点
PR(Ti)：其他节点的PR值
L(Ti)：其他节点的出链数

PageRank根据假设的原始公式（使用马尔可夫矩阵表达）：

$$PR = M * V^{2} \tag{1.2}$$

PageRank为解决Dead Ends，修正M后的公式：

$$PR = (M + a^{T} * \frac{ee^{T}}{n}) * V^{3} \tag{1.3}$$

PageRank为解决Spider Traps，修正M后的公式：

$$PR = [d * M + (1-d) * \frac{ee^{T}}{n}] * V^{4} \tag{1.4}$$

PageRank最终公式：

$$PR = [d * (M + a^{T} * \frac{ee^{T}}{n}) + (1-d) * \frac{ee^{T}}{n}] * V \tag{1.5}$$

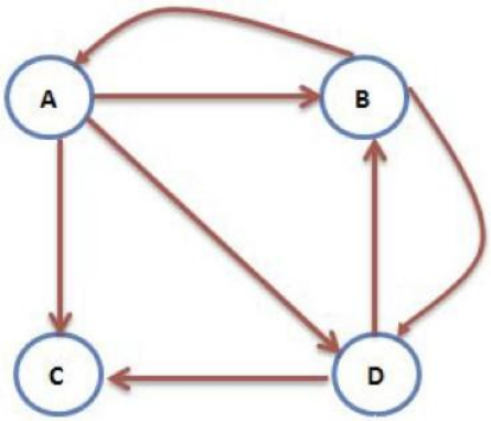
[2]M:马尔可夫矩阵/转移概率矩阵
V：PR值矩阵

[3]a=[a1,a2...ai...],若M的某列全为0则ai=1，其余均为0
e：全1的列矩阵
n：记录的总数

[4]d:跟随出链打开网页的概率
1-d：随机跳到其他网页的概率

第二节　算法设计

2.1　算法流程（以例子进行说明）



1表示有从列x到行y的链接

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

图 1: 数据展示

$$\begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

图 2: 转为马尔可夫矩阵

2.2　核心代码

源代码 1:

```
1
2 # -*- coding: utf-8 -*-
```

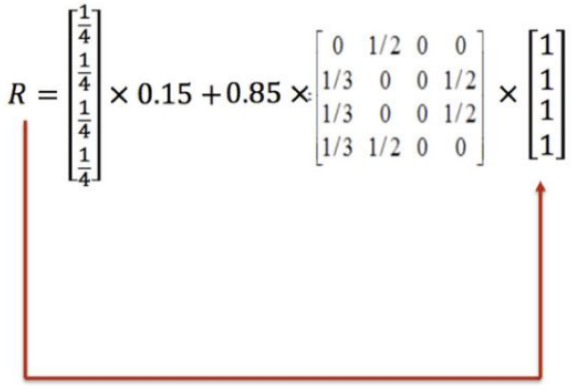


图 3: 进行迭代

```
"""
Created on Thu Jul 4 18:44:07 2019

@author: zsl
"""

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx


f = open('input.txt', 'r')
edges = [line.strip('\n').split(' ') for line in f]

G = nx.DiGraph()
for edge in edges:
    G.add_edge(edge[0], edge[1])
nx.draw(G, with_labels=True)
plt.show()

filename ='input.txt'
def load_data(f):
    data =open(f,'r')
    edges =[line.strip('\n').split(' ') for line in data]
    return edges

def get_V(edges):
    nodes =[]
    for edge in edges:
        for i in edge:
            if i not in nodes:
                nodes.append(i)
    print(nodes)
```

```
36
37      N=len(nodes)
38      i = 0
39      node_to_num ={}
40      for node in nodes:
41          node_to_num[node] =i
42          i += 1
43      for edge in edges:
44          edge[0] =node_to_num[edge[0]]
45          edge[1] =node_to_num[edge[1]]
46      print(edges)
47
48      M = np.zeros([N, N])
49      for edge in edges:
50          M[edge[1], edge[0]] =1
51      print(M)
52      for j in range(N):
53          sum_cnt =sum(M[:,j])
54          for k in range(N):
55              if sum_cnt !=0:
56                  M[k,j]/=sum_cnt
57              else:
58                  M[k,j]=1/N
59      print('=================',M)
60      V = np.ones(N)/N
61      return M,V,N
62
63  def PageRank(M,V,N,beta=0.85):
64      cnt =0
65      e=np.ones([N,N])/N
66      M_ = np.dot(M,beta)+np.dot(1-beta,e)
67      er =100000
68      pr1=V
69      re=[]
70      while er >0.00000001:
71          pr2 =np.dot(M_,pr1)
72          er =pr2-pr1
73          er = max(map(abs, er))
74          pr1 =pr2
75          cnt +=1
76          print('iteration %s'%str(cnt),pr1)
77          if cnt%10==0:
78              re.append(pr1.tolist())
79      return pr1,re
80
81  data=load_data(filename)
82  M,V,N =get_V(data)
83  pr1,re =PageRank(M,V,N)
84  print(re)
85  re = np.array(re)
86  ln1, =plt.plot(re[:,0], color ='red', linewidth =2.0, linestyle ='--')
87  ln2, =plt.plot(re[:,1], color ='blue', linewidth =2.0, linestyle ='--')
88  ln3, =plt.plot(re[:,2], color ='pink', linewidth =2.0, linestyle ='--')
89  ln4, =plt.plot(re[:,3], color ='green', linewidth =2.0, linestyle ='--')
```

```
ln5, =plt.plot(re[:,4], color ='yellow', linewidth =2.0, linestyle ='--')
plt.legend(handles=[ln1,ln2,ln3,ln4,ln5], labels=['A', 'B','C','D','F'],
    loc='lower right')
plt.plot(re)

plt.bar(['A', 'B','C','D','F'], pr1)
plt.show()
```

第三节　选用数据

自拟：

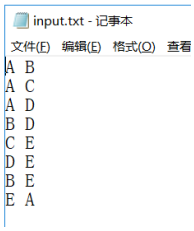


图 4: 每行表示前者指向后者

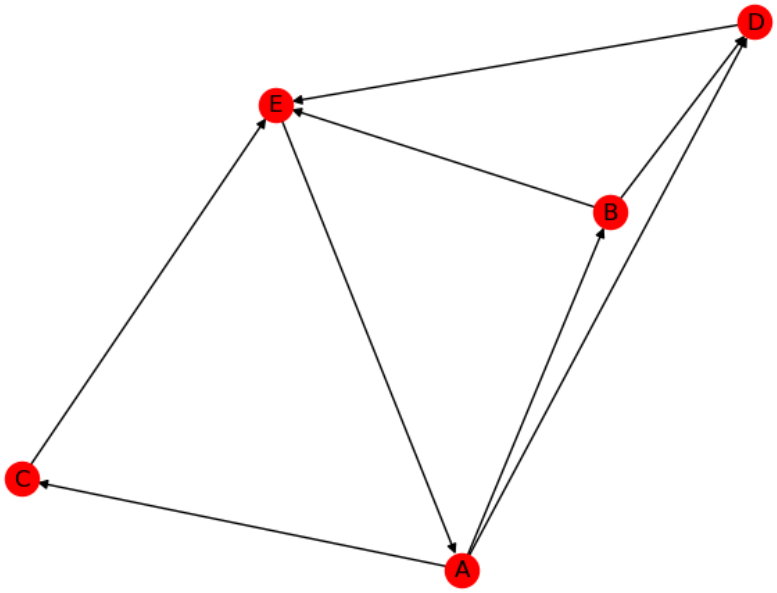


图 5: 数据图示

第四节　实验结果展示

```
pr1
array([0.29633859, 0.1139626 , 0.1139626 , 0.1623967 , 0.31333951])
```

图 6: 最终循环得到的PR值

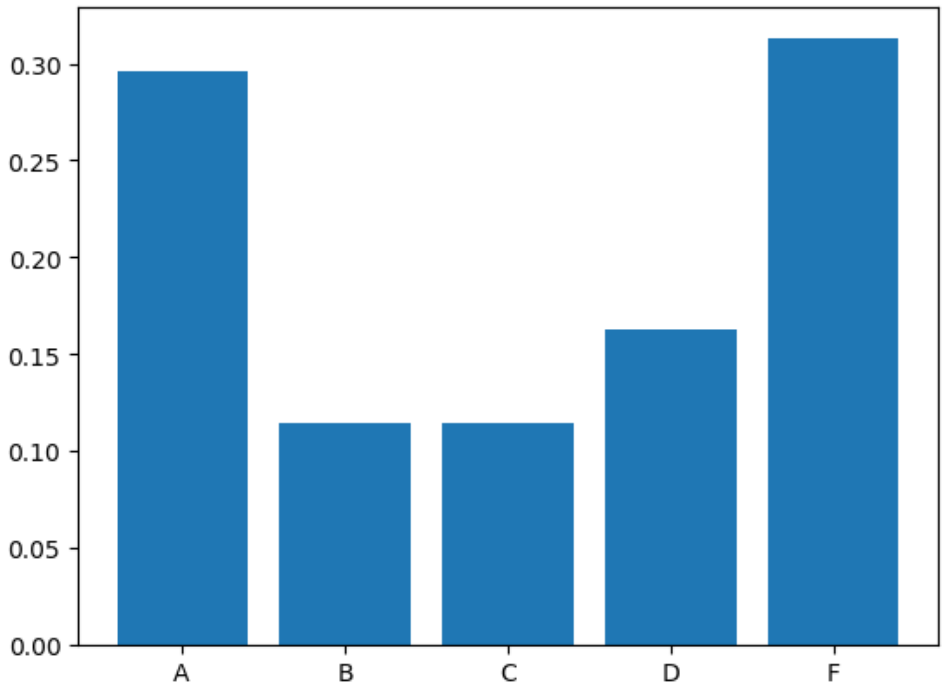


图 7: 最终循环得到的PR值

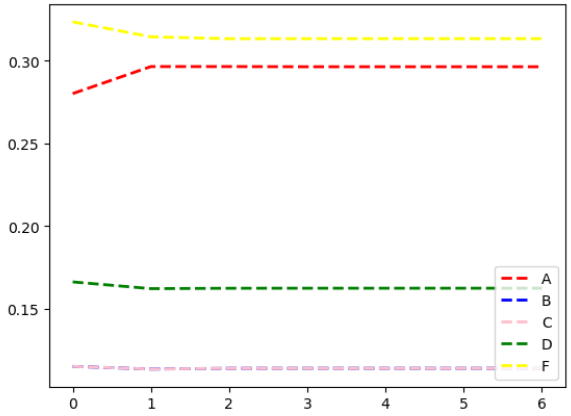


图 8: 每循环10次变化折线

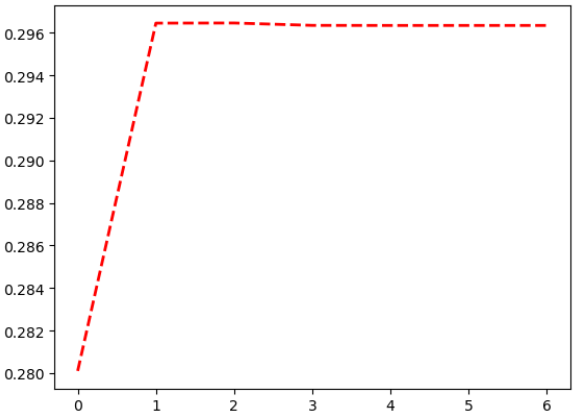


图 9: A的变化

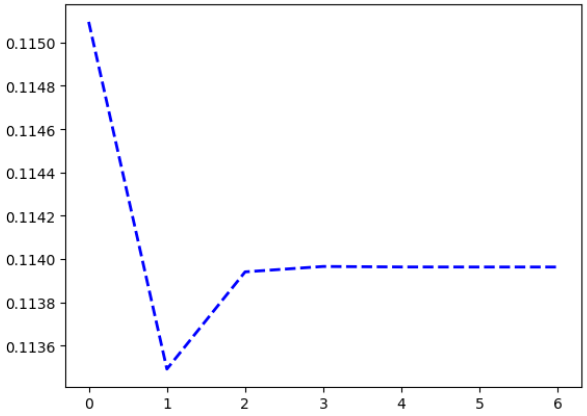


图 10: B的变化

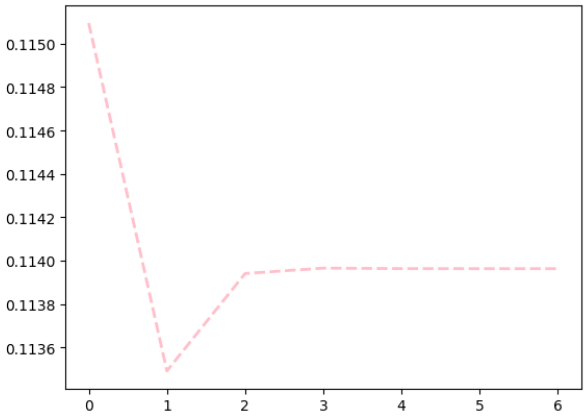


图 11: C的变化

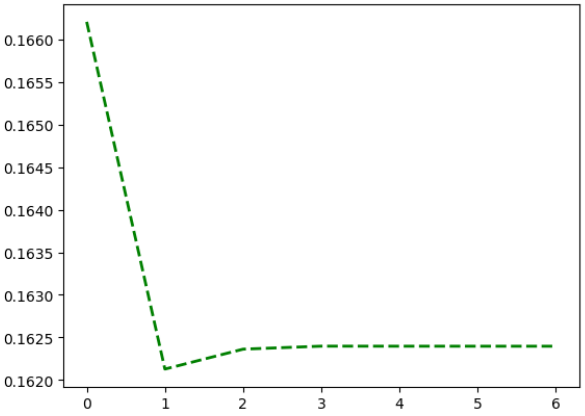


图 12: D的变化

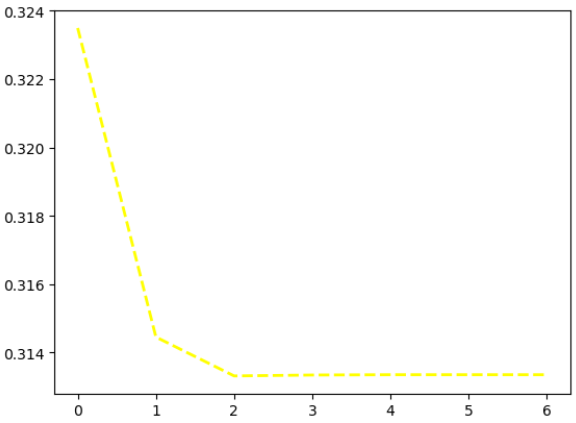


图 13: E的变化

第五节　实验分析和比较

基本在循环30次之后，PR值均趋于稳定。最后的排名是F,A,D,B,C.由于找的真实数据量太大，电脑内存不足，没办法运行，只能使用自拟数据

PageRank的优点：一定程度上消除了人为的影响；离线计算PR值，而非查找时计算，提高了查找效率。缺点：时间久的网站PR值越来越大，新生网站生长缓慢；可以通过“僵尸”网站或者链接人为的刷PR值