exercises

August 15, 2019

[11]:
```
import numpy as np

# https://glowingpython.blogspot.com/2011/05/
 ↪four-ways-to-compute-google-pagerank.html
def get_dimension(A):
    # Returns the dimensions of a square matrix
    return A.shape[1]

def get_eigenvector(A):
    n = get_dimension(A)
    _, v = np.linalg.eig(A)

    return np.abs(np.real(v[:n, 0]) / np.linalg.norm(v[:n, 0], 1))

def get_eigenvalues(A):
    # Returns the eigen values
    eigenvalues, _ = np.linalg.eig(A)
    return eigenvalues

def create_M(A, m):
    # Returns (1-m)A + mS
    # This is the weighted matrix where m is user defined
    # Google set m = 0.15
    n = get_dimension(A)
    S = np.ones((n,n)) / n
    return (1 - m) * A + m * S

def print_page_scores(vector):
    i = 1
    for page in vector:
        print("Page {} has score {:.3f}".format(i, page))
        i+= 1

def print_matrix(matrix):
    for row in matrix:
        for value in row:
            print("{:.3f}".format(value), end="\t")
```

```
        print("")
```

1 Exercises

1.1 Exercise 1

[2]:
```
A = np.array([
    [0,   0,   1/2, 1/2, 0],
    [1/3, 0,   0,   0,   0],
    [1/3, 1/2, 0,   1/2, 1],
    [1/3, 1/2, 0,   0,   0],
    [0,   0,   1/2, 0,   0]
])

vector = get_eigenvector(A)
print_page_scores(vector)
```

```
Page 1 has score 0.245
Page 2 has score 0.082
Page 3 has score 0.367
Page 4 has score 0.122
Page 5 has score 0.184
```

It would appear that page 3's score has increased above that of page 1. If a page 5 is added, the following two items will happen: 1. The "vote" of page 3 will now split between page 1 and page 5, therefore reducing page 1's score 2. The "vote" of page 5 will boost the score of page 3.

1.2 Exercise 2

[3]:
```
A = np.array([
    [0, 1, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1/2],
    [0, 0, 0, 0, 1, 0, 1/2],
    [0, 0, 0, 0, 0, 0, 0]
])

eigenvalues = get_eigenvalues(A)

print("The eigen values are:", eigenvalues)
print("There are {} eigenvalues with value 1".format(len(np.where(eigenvalues␣
 ↪== 1)[0])))
```

```
The eigen values are: [ 1. -1.  1. -1.  1. -1.  0.]
There are 3 eigenvalues with value 1
```

The example web has 3 separate subwebs, similar to figure 2.2. As you can see, there are three eigenvalues equal to 1. This demonstrates that the dimension of $V_1(A)$ equals (or exceeds) the number of components in the web.

1.3 Exercise 3

[4]:
```
A = np.array([
    [0, 1, 0, 0, 1/3],
    [1, 0, 0, 0, 0],
    [0, 0, 0, 1, 1/3],
    [0, 0, 1, 0, 1/3],
    [0, 0, 0, 0, 0],
])

eigenvalues = get_eigenvalues(A)

print("The eigen values are:", eigenvalues)
print("There are {} eigenvalues with value 1".format(len(np.where(eigenvalues␣
 ↪== 1)[0])))
```

```
The eigen values are: [ 1. -1.  1. -1.  0.]
There are 2 eigenvalues with value 1
```

1.4 Exercise 4

[5]:
```
A = np.array([
    [0   , 0   , 0, 1/2],
    [1/3, 0   , 0, 0  ],
    [1/3, 1/2, 0, 1/2],
    [1/3, 1/2, 0, 0  ]
])

eigenvalues, eigenvectors = np.linalg.eig(A)

print(eigenvalues)
```

```
[ 0.         +0.j          -0.28067662+0.26395346j -0.28067662-0.26395346j
  0.56135324+0.j        ]
```

Thus, the largest positive (Perron) enigenvalue is 0.56135324

[6]:
```
v = eigenvectors[:4, 3]
v_scaled = np.abs(v) / np.linalg.norm(v, 1)
```

The resulting ranking seems reasonable. Page 3, linked to by all other pages, is the most important. Page 4, linked to by page 1 and 2 scores the second, while page 1 and page 2 are only linked to by one page with 1/2 and 1/3 vote namely.

1.5 Exercise 5

Prove that in any web the importance score of a page with no backlinks is zero.

Let x_i denote the importance score of page i. If page i has no backlinks, then the i_{th} row of the adjacency matrix A is a zero-vector. Thus for

$$AX = \lambda X$$

where $\lambda_i = 1$

$$x_i = 0$$

1.6 Exercise 6

- 1.

To transpose page i and page j, we need to transpose both the outgoing links from page i, j and backlinks to page i, j,

Compared with the original link matrix A, the new link matrix $\overline{A}$ swaps row i with row j, and swaps column i with column j. Thus,

$$P\overline{A} = AP(\overline{A}P = PA) \tag{1}$$

Also, P^2 = I Therefore, $P\overline{A}P = A$

- 2.

Suppose X is the eigenvector of A, then

$$AX = \lambda X$$

$$PAX = \lambda PX$$

From 1,

$$\overline{A}P = PA$$

Thus,

$$\overline{A}PX = \lambda PX$$

Therefore, $y = PX$ is an eigenvector for $\overline{A}$ with eigenvalue λ

- 3.

Suppose X_i is the eigenvector of A where $\lambda_i = 1$

Thus,

$$AX_i = \lambda_i X_i$$

Since P is an identity matrix, then

$$PX_i = X_i$$

From 2, we have

$$\overline{A}PX = \lambda PX$$

Thus,

$$\overline{A}PX_i = \lambda_i PX_i$$

From (8) (10),

$$\overline{A}X_i = \lambda_i X_i$$

Therefore, X_i is also the eigenvector for $\overline{A}$ with eigenvalue $\lambda_i = 1$, so the importance scores are left unchanged after the single permutation.

For multiple permutations, we can draw the conclusion of the importance scores unchanged by iterating over the above steps.

1.7 Exercise 7

Since A is a $n \times n$ column-stochastic matrix,

Thus,

$$\forall i, a_{ij} \in [0, 1]$$

$$\sum_{j \in n} a_{ij} = 1$$

For matrix S,

$$\forall i, j, s_{ij} = \frac{1}{n}$$

$$\sum_{j \in n} s_{ij} = 1$$

Since

$$m \in [0, 1], 1 - m \in [0, 1]$$

then

$$0 \leq (1 - m)a_{ij} + ms_{ij} \leq (1 - m) + m = 1$$

Also for any column j

$$\sum_j (1 - m)A + mS = \sum_j (1 - m)a_{ij} + ms_{ij} = (1 - m)\sum_j a_{ij} + m\sum_j s_{ij} = (1 - m) + m = 1$$

Therefore, $M = (1 - m)A + mS$ is also a column-stochastic matrix

1.8 Exercise 8

Suppose A and B are both $n \times n$ column-stochastic matrices,

Thus,

$$\forall i, a_{ij} \in [0, 1], \sum_j a_{ij} = 1$$

$$\forall i, b_{ij} \in [0, 1], \sum_j b_{ij} = 1$$

For

$$M = A \cdot B$$

$$m_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + ... + a_{in} b_{nj}$$

Thus, for each column j:

$$\sum_{i=1}^{n} m_{ij} \tag{2}$$

$$= \sum_{j=1}^{n} \sum_{k=1}^{n} a_{ik} b_{kj} \tag{3}$$

$$= (a_{11}b_{1j} + a_{12}b_{2j} + ... + a_{1n}b_{nj}) + (a_{21}b_{1j} + a_{22}b_{2j} + ... + a_{2n}b_{nj}) + ... + (a_{n1}b_{1j} + a_{n2}b_{2j} + ... + a_{nn}b_{nj}) \tag{4}$$

$$= b_{1j}(a_{11} + a_{21} + ... + a_{n1}) + b_{2j}(a_{12} + a_{22} + ... + a_{n2}) + ... + b_{nj}(a_{1n} + a_{2n} + ... + a_{nn}) \tag{5}$$

$$= b_{1j} + b_{2j} + ... + b_{nj} = 1 \tag{6}$$

Therefore, $M = A \cdot B$ is also a column-stochastic matrix

1.9 Exercise 9

A page with no backlinks will have a score of $\frac{m}{n}$ because the array S would give it an score of $\frac{1}{n}$ (as the array must be column stochastic and equally weighted for all pages) and then by exercise 5, the page would have a score of 0 from the array A.

Therefore, the score of a particular page would be the score given by

$$(1-m)A + mS = (1-m) * 0 + m\frac{1}{n} = \frac{m}{n}$$

Note how A resolves to 0 because the page has no backlinks.

1.10 Exercise 10 - INCOMPLETE

- **1. Show that $(A^2)_{ij} > 0$ if and only if page i can be reached from page j in exactly two steps.**

Let A be a link matrix with non-negative values in position $A_{ij} \forall i, j$ if there exists a link between pages i and j.

Further, consider that when computing A^2, A_{ij} will be non-negative if there exists a link from page i to page k and from page k to page j (where $0 \leq k \leq n$ and $k \neq i$ and $k \neq j$).

We can prove this to be true by examining $A_{ij}^2 = \sum_{k=1}^{n} A_{ik}A_{kj}$. So if any A_{ik} and A_{kj} is nonzero, then the sum will similarly be non-zero.

- $A_{ik} > 0$
- $A_{kj} > 0$
- Therefore, $A_{ij}^2 \geq A_{ik} \cdot A_{kj} > 0$

Note that if a link from page i to j is not possible in exactly two steps, then the value of that cell will be 0 as (per the formula described), the sum of all the links for $A_{ik} * A_{kj}$ for all $k \in 0..n$ will be 0.

Therefore, A^2ij will be non-negative if and only if we can reach page i from page j in exactly 2 steps.

- **2. Show more generally that $(A^p)_{ij} > 0$ if and only if page i can be reached from page j in EXACTLY p steps.**

Consider a page i that can be reached in k steps from page j.
This means that for each step, there exists a page q_t and page q_{t+1} with a link along the chain.
Therefore, we can write $(A^p)_{ij} = \Pi_{t=1}^{n} A_{q_t q_{t+1}}$ where $q_1 = A_{iq_2}$ and $q_n = A_{q_{n-1}j}$
The formula described means that we have links from page i to page q1, q1 to q2, [...], and finally qn-1 to page j to complete the chain.
Expanded from the base case might be $(A^2)_{ij} = A_{ik}A_{kj}$

- **3. Argue that $(I + A + A^2 + ... + A^p)_{ij} > 0$ if and only if page i can be reached from page j in p or fewer steps.**

Consider that if we sum A raised to the kth power from $k = \{0..p\}$. Each power of A represents if a node is reachable in that number of steps.
By definition, if there exists a link, the value will be non-zero and if there exists no link the value will be zero.
Therefore, the value of $(I + A + A^2 + ... + A^p)_{ij} = (I_{ij} + A_{ij} + A_{ij}^2 + ... + A_{ij}^p)$ which is non-zero if some value is non-zero.

- **4. Explain why $I + A + A^2 + ... + A^{n-1}$ is a positive matrix if the web is strongly connected.**

In order for the matrix above to be a positive matrix, this would imply that **every** page can be reached by any other page in at most $n - 1$ steps.
If a web is not strongly connected, then $\exists (I + A + A^2 + ... + A^{n-1})_{ij} = 0$ for some i, j (where page i cannot reach page j in $n - 1$ steps) which is not a positive matrix.

- **5. Use the last part (and Exercise 8) to show that $B = \frac{1}{n}(I + A + A^2 + ... + A^{n-1})$ is positive and column-stochastic (and hence by Lemma 3.2 $dim(V_1(B)) = 1$).**

First note that each item in $(A^2 + ... + A^{n-1})$ is column-stochastic as proven by exercise 8 which states that the product of two column-stochastic matrices is also column-stochastic.
Next, consider that each component in $(I + A + A^2 + ... + A^{n-1})$ is nonnegative as proven in part 3.
Now, consider that the sum of n column-stochastic matrices would yield a matrix that has sum n for each column. Therefore, if we divide this matrix by n, we get another column-stochastic matrix.
Therefore, we have a matrix that is both positive and column-stochastic and therefore $dim(V_1(B)) = 1$.

- **6. Show that if $x \in V_1(A)$ then $x \in V_1(B)$.**

1.11 Exercise 11

[7]:
```
# Figure 2.1 with addition of page 5 that links to page 3 and page 3 also links␣
 ↪to page 5.
A = np.array([
    [0,   0,   1/2, 1/2, 0],
    [1/3, 0,   0,   0,   0],
    [1/3, 1/2, 0,   1/2, 1],
    [1/3, 1/2, 0,   0,   0],
    [0,   0,   1/2, 0,   0]
])

# Calculate the new ranking by finding the eigenvector of M
# Using an S, a matrix of 1/n in each cell
# use m = 0.15
M = create_M(A, m=0.15)

vector = get_eigenvector(M)
print_page_scores(vector)
```

```
Page 1 has score 0.237
Page 2 has score 0.097
Page 3 has score 0.349
Page 4 has score 0.138
Page 5 has score 0.178
```

1.12 Exercise 12

[8]:
```
A = np.array([
    [0,   0,   1/2, 1/2, 0, 1/5],
    [1/3, 0,   0,   0,   0, 1/5],
    [1/3, 1/2, 0,   1/2, 1, 1/5],
    [1/3, 1/2, 0,   0,   0, 1/5],
    [0,   0,   1/2, 0,   0, 1/5],
    [0,   0,   0,   0,   0, 0  ]
])

# Calculate the new ranking by finding the eigenvector of M
# Using an S, a matrix of 1/n in each cell
# use m = 0.15
M = create_M(A, m=0.15)

print("The matrix M (weighted average of A and S) is:")
print_matrix(M)
```

```
The matrix M (weighted average of A and S) is:
0.025   0.025   0.450   0.450   0.025   0.195
0.308   0.025   0.025   0.025   0.025   0.195
0.308   0.450   0.025   0.450   0.875   0.195
```

```
0.308   0.450   0.025   0.025   0.025   0.195
0.025   0.025   0.450   0.025   0.025   0.195
0.025   0.025   0.025   0.025   0.025   0.025
```

[9]:
```
vector = get_eigenvector(M)
print_page_scores(vector)
```

```
Page 1 has score 0.231
Page 2 has score 0.095
Page 3 has score 0.340
Page 4 has score 0.135
Page 5 has score 0.174
Page 6 has score 0.025
```

1.13 Exercise 13

[14]:
```
# example of 3 separate subwebs with seven pages
A = np.array([
    [0, 1, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1/2],
    [0, 0, 0, 0, 1, 0, 1/2],
    [0, 0, 0, 0, 0, 0, 0]
])

# Calculate the new ranking by finding the eigenvector of M
# Using an S, a matrix of 1/n in each cell
# use m = 0.15
M = create_M(A, m=0.15)

print("The matrix M (weighted average of A and S) is:")
print_matrix(M)

print("")

vector = get_eigenvector(M)
print_page_scores(vector)
```

```
The matrix M (weighted average of A and S) is:
0.021   0.871   0.021   0.021   0.021   0.021   0.021
0.871   0.021   0.021   0.021   0.021   0.021   0.021
0.021   0.021   0.021   0.871   0.021   0.021   0.021
0.021   0.021   0.871   0.021   0.021   0.021   0.021
0.021   0.021   0.021   0.021   0.021   0.871   0.446
0.021   0.021   0.021   0.021   0.871   0.021   0.446
```

```
0.021   0.021   0.021   0.021   0.021   0.021   0.021

Page 1 has score 0.143
Page 2 has score 0.143
Page 3 has score 0.143
Page 4 has score 0.143
Page 5 has score 0.204
Page 6 has score 0.204
Page 7 has score 0.021
```

1.14 Exercise 14 - CLEAN UP

First, we can represent the $M^k \cdot x_0$ with the following Python function:

[4]:
```python
def calc_q(M, x0, k):
    # this calculates (M^k)*x0
    m_to_power_k = np.linalg.matrix_power(M, k)
    return np.matmul(m_to_power_k, x0)
```

[21]:
```python
# Represents the web in Exercise 11
ex_11A = np.array([
    [0,   0,   1/2, 1/2, 0],
    [1/3, 0,   0,   0,   0],
    [1/3, 1/2, 0,   1/2, 1],
    [1/3, 1/2, 0,   0,   0],
    [0,   0,   1/2, 0,   0]
])

fig_2_2A = np.array([
    [0,   1,   0,   0,   0],
    [1,   0,   0,   0,   0],
    [0,   0,   0,   1,   1/2],
    [0,   0,   1,   0,   1/2],
    [0,   0,   0,   0,   0]
])

# Our initial guess for q, an evenly distributed set of values
x0 = np.array([
    [.20],
    [.20],
    [.20],
    [.20],
    [.20]
])

# Create our M array, using m=0.15
M = create_M(ex_11A, m=0.15)

# The resulting matrix
```

```
print("Printing M:")
print_matrix(M)

# Printing our initial guess
print("\nPrinting x0 (initial guess):")
print_matrix(x0)

def diff_q_iter_and_q_actual(M, k, x0):
    # first column of table under 4.2
    # This is the iterative calculation of q
    q_iter = calc_q(M, x0, k)
    q_actual = calc_q(M, x0, 1000000)

    # Frobenius norm (1st)
    return np.linalg.norm(q_iter-q_actual, 1)

def calc_convergence(M, k, x0):
    # second column of table under 4.2
    # This is the rate of convergence for a particular k
    x_k = diff_q_iter_and_q_actual(M, k, x0)
    x_k_minus_1 = diff_q_iter_and_q_actual(M, k - 1, x0)

    return x_k / x_k_minus_1

# Print a representation of the table under 4.2 for this exercise
print("k\t|x_k - q\t|Rate of Convergence (c)")
print("================================================")
for i in [1,5,10,50]:
    col1 = diff_q_iter_and_q_actual(M, i, x0)
    col2 = calc_convergence(M, i, x0)
    print("{}\t|{:.5f}\t|{:.5f}".format(i, col1, col2))

def get_second_highest_eigenval(M):
    vals = sorted(get_eigenvalues(M))
    return vals[-2].real

lam2 = get_second_highest_eigenval(M)

A_ = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
])

M = create_M(ex_11A, m=0.15)

def calc_c(M):
```