Numerical Analysis HW 13

Kevin Ho

April 2025

13.1. Consider Example 13.13, p.488. Compute vectors of cosines, for each subspace approximations, i.e., with A_k where $k = 1, 2, \ldots, 5$.

Cosine Similarities

For $k = 1$: $\begin{bmatrix}1. & 1. & 1. & 1. & 1.\end{bmatrix}$
For $k = 2$: $\begin{bmatrix}0.7857 & 0.8332 & 0.967 & 0.4873 & 0.1819\end{bmatrix}$
For $k = 3$: $\begin{bmatrix}0.1024 & 0.8501 & 0.8371 & 0.4218 & 0.4685\end{bmatrix}$
For $k = 4$: $\begin{bmatrix}0.054 & 0.8476 & 0.8291 & 0.4165 & 0.3918\end{bmatrix}$
For $k = 5$: $\begin{bmatrix}0. & 0.7223 & 0.8393 & 0.3612 & 0.3612\end{bmatrix}$

13.2. Verify equations in Derivation 13.26, p.497, particularly (13.28), (13.29), and (13.30).

We start with equation (13.28):

$$z = \alpha Qy + \beta \frac{e}{n}$$

Left-multiply by the vector of ones, e^T:

$$e^T z = e^T \left(\alpha Qy + \beta \frac{e}{n} \right)$$

Distribute e^T:

$$e^T z = \alpha (e^T Qy) + \beta \left(e^T \frac{e}{n} \right)$$

Evaluate $e^T e$. Since e is a vector of n ones, $e^T e = 1 \cdot 1 + \cdots + 1 \cdot 1 = n$:

$$e^T z = \alpha (e^T Qy) + \beta \left(\frac{n}{n} \right)$$
$$e^T z = \alpha e^T Qy + \beta$$

From premise (2), we know $e^T z = \|z\|_1 = 1$. Substitute this value:

$$1 = \alpha e^T Qy + \beta$$

Now, solve for β:

$$\beta = 1 - \alpha e^T Qy$$

This verifies the first equality in (13.30).

To verify the second equality, $\beta = 1 - \|\alpha Qy\|_1$, we need to relate $e^T Qy$ to the L1 norm.

- In PageRank, y represents probabilities/scores, so its entries are non-negative.
- The matrix Q represents transition probabilities (possibly adjusted for dangling nodes), so its entries are also non-negative.
- Therefore, the vector Qy has non-negative entries. Let $v = Qy$, then $v_i \geq 0$.
- The L1 norm of a vector v with non-negative entries is $\|v\|_1 = \sum_i |v_i| = \sum_i v_i$.
- The sum of the elements of a vector v is $e^T v$.
- Thus, for the non-negative vector Qy, we have $\|Qy\|_1 = e^T(Qy)$.

Substituting this into our result for β:

$$\beta = 1 - \alpha(e^T Qy)$$
$$\beta = 1 - \alpha(\|Qy\|_1)$$

Since $\alpha > 0$, we can move it inside the norm:

$$\beta = 1 - \|\alpha Qy\|_1$$

This verifies the second equality in (13.30).

Conclusion: We have successfully shown, starting from equation (13.28) and using property (13.27), that $\beta = 1 - \alpha e^T Qy$ and subsequently that $\beta = 1 - \|\alpha Qy\|_1$. This completes the verification of equation (13.30).

13.3. Consider the link matrix Q in (13.4) and its corresponding link graph in Figure 13.2. Find the pagerank vector r by solving the Google pagerank equation.

- You may initialize the power method with any vector $r^{(0)}$ satisfying $\|r^{(0)}\|_1 = 1$.
- Set $\alpha = 0.85$.
- Let the iteration stop, when residual $< 10^{-4}$.

PageRank Vector: $\begin{bmatrix}0.057922 & 0.057922 & 0.249026 & 0.116522 & 0.206839 & 0.311769\end{bmatrix}$

13.4. Now, consider a **modified** link matrix $\tilde{Q}$, by adding an outlink from page ④ to ⑤ in Figure 13.2. Find the pagerank vector $\tilde{r}$, by setting parameters and initialization the same way as for the previous problem.

- Compare r with $\tilde{r}$.
- Compare the number of iterations for convergence.

PageRank Vector: $\begin{bmatrix}0.034884 & 0.034884 & 0.241867 & 0.111357 & 0.269946 & 0.307063\end{bmatrix}$

Table 1: Comparison of PageRank Vectors (r vs e_r)

Page	Original r	Modified e_r	Difference $(e_r - r)$
1	0.057 922	0.034 884	−0.023 038
2	0.057 922	0.034 884	−0.023 038
3	0.249 026	0.241 867	−0.007 159
4	0.116 522	0.111 357	−0.005 165
5	0.206 839	0.269 946	0.063 107
6	0.311 769	0.307 063	−0.004 706

Overall the iterations for convergence for r took 11 iterations while the modified link matrix ended 16 for me.

Code for Homework 13

13.1

```
import numpy as np

# Define the term-document matrix A (10x5)
A = np.array([
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1],
    [0, 0, 0, 0, 1],
    [1, 0, 1, 0, 0],
    [1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0],
    [1, 0, 1, 1, 0],
    [0, 1, 1, 0, 0],
    [0, 0, 1, 1, 1],
    [0, 1, 1, 0, 0]
], dtype=float)

# Define the query vector q (10-dimensional)
q = np.array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1], dtype=float)

# Compute the full SVD of A
U, s, Vh = np.linalg.svd(A, full_matrices=False)


L = np.array([0,0,0,1,1,0,1,0,0,0])

print(L @ L.T)

def cosine_similarity(vec1, vec2):
    norm1 = np.linalg.norm(vec1)
    norm2 = np.linalg.norm(vec2)
    if norm1 == 0 or norm2 == 0:
        return 0.0
    return np.dot(vec1, vec2) / (norm1 * norm2)


# Compute cosine similarities for each rank-k approximation
results = {}
for k in range(1, 6):
    print("K = {}".format(k))
    Uk = U[:, :k]
    Sk = np.diag(s[:k])
    Vhk = Vh[:k, :]
    print(f"\nSTEP {np.round(k, 4)} ")
    print(f"Uk ={np.round(Uk,4)}\n")
    print(f"Sk ={np.round(Sk,4)}\n")
    print(f"Vhk ={np.round(Vhk,4)}")

    Dk = Sk @ Vhk
    qb_k = Uk.T @ q
```

```
    cosines = []
    for j in range(5):
        doc_vec = Dk[:, j]
        cos_sim = cosine_similarity(qb_k, doc_vec)
        cosines.append(cos_sim)
    results[k] = np.array(cosines)


for k in range(1, 6):
    print(f"k = {k} cosines: {np.round(results[k], 4)}")

orig_cosines = []
for j in range(5):
    doc_vec = A[:, j]
    cos_sim = cosine_similarity(q, doc_vec)
    orig_cosines.append(cos_sim)
print("\nOriginal cosine similarities (full space):", np.round(orig_cosines,
4))
```

13.3 & 13.4

```
import numpy as np


def calculate_pagerank(Q, alpha=0.85, tolerance=1e-4, max_iterations=100):

    n = Q.shape[0]
    if n == 0:
        return np.array([]), 0
    S = Q.copy()
    column_sums = np.sum(S, axis=0)
    dangling_nodes = np.where(column_sums == 0)[0]
    uniform_prob = 1.0 / n
    for node_idx in dangling_nodes:
        S[:, node_idx] = uniform_prob
    r = np.full(n, uniform_prob)
    p = np.full(n, uniform_prob)
    iterations = 0
    residual = 1.0

    while residual >= tolerance and iterations < max_iterations:
        r_prev = r.copy()
        r = alpha * (S @ r_prev) + (1 - alpha) * p
        r /= np.sum(r)
        residual = np.linalg.norm(r - r_prev, 1)
        iterations += 1

    if iterations == max_iterations:
        print(f"Warning: PageRank did not converge within {max_iterations}
iterations.")
        return None, iterations
```

```
    return r, iterations


# --- Problem 13.3 ---
print("--- Problem 13.3: Original Link Matrix Q ---")

# Define the original link matrix Q from (13.4)
Q = np.array([
    [0, 1 / 3, 0, 0, 0, 0],
    [1 / 3, 0, 0, 0, 0, 0],
    [0, 1 / 3, 0, 0, 1 / 3, 1 / 2],
    [1 / 3, 0, 0, 0, 1 / 3, 0],
    [1 / 3, 1 / 3, 0, 0, 0, 1 / 2],
    [0, 0, 1, 0, 1 / 3, 0]
])

# Set parameters
alpha = 0.85
tolerance = 1e-4

# Calculate PageRank for original Q
r, iterations_r = calculate_pagerank(Q, alpha, tolerance)

if r is not None:
    print(f"Converged in {iterations_r} iterations.")
    print("PageRank vector r:")
    for i, rank in enumerate(r):
        print(f"  Page {i + 1}: {rank:.6f}")

# --- Problem 13.4 ---
print("\n--- Problem 13.4: Modified Link Matrix Qe ---")
Qe = Q.copy()
Qe[:, 3] = 0.0
Qe[4, 3] = 1.0

# Calculate PageRank for modified Qe
er, iterations_er = calculate_pagerank(Qe, alpha, tolerance)

if er is not None:
    print(f"Converged in {iterations_er} iterations.")
    print("PageRank vector er (modified):")
    # Print nicely formatted
    for i, rank in enumerate(er):
        print(f"  Page {i + 1}: {rank:.6f}")


# --- Comparison ---
print("\n--- Comparison ---")
if r is not None and er is not None:
    print(f"Number of iterations (original Q): {iterations_r}")
    print(f"Number of iterations (modified Qe): {iterations_er}")

    print("\nComparison of PageRank Vectors (r vs er):")
    print("Page | Original r | Modified er | Difference (er - r)")
    print("-----------------------------------------------------")
    for i in range(len(r)):
```

```
        diff = er[i] - r[i]
        print(f"  {i + 1}  |  {r[i]:.6f}  |  {er[i]:.6f}  | {diff:+.6f}")
```