MEM704 - Machine Learning

1st Laboratory Exercise: The PageRank Algorithm and SVD Analysis

Submission Deadline: 07/03/2024, 18:00
Examination: 08/03/2024 in the Lab

This exercise focuses on two key numerical algorithms: the **PageRank algorithm** for ranking web pages and **Singular Value Decomposition (SVD)** for image compression. We will also explore the **power method** for computing dominant eigenvalues and eigenvectors, which is essential in both tasks.

1 Power Method

The **power method** approximates the dominant eigenvalue λ and its corresponding eigenvector x for a given matrix $A \in \mathbb{R}^{n \times n}$. The algorithm proceeds as follows:

Power Method Algorithm

1. Choose a random initial vector x_0 and normalize it:
$$x_0 = \frac{x_0}{\|x_0\|}$$
2. Set parameters:
   - Maximum iterations: $k_{\max}$
   - Error threshold: $\epsilon \approx 10^{-6}$
3. Initialize: $k = 0, d_k = 1$
4. **Repeat until convergence:**
   - Compute $x_k = Ax_{k-1}$
   - Normalize: $x_k = \frac{x_k}{\|x_k\|}$
   - Compute difference: $d_k = \|x_k - x_{k-1}\|$
   - Estimate eigenvalue: $\lambda = \frac{x_k^T A x_k}{x_k^T x_k}$

Task:

- Implement this algorithm in **Python** using NumPy ('linalg' and 'random' packages).
- Test it on a matrix A of size $n \geq 3$ with known eigenvalues.
- Compare the convergence rate d_{k+1}/d_k with the theoretical ratio $|\lambda_2/\lambda_1|$.
- Apply it to the tridiagonal matrix:

$$T = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

where the eigenvalues are:

$$\lambda_k = 2 - 2\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, ..., n.$$

2 PageRank Algorithm

The **PageRank algorithm**, originally developed by Google, estimates the importance of a web page based on link structures. It finds the dominant eigenvector of a **Markov matrix**:

$$M = d \cdot A + \frac{(1-d)}{N} B,$$

where:

- $d \approx 0.85$ (damping factor)
- A is the link matrix:

$$a_{ij} = \begin{cases} \frac{1}{L(j)}, & \text{if page } j \text{ links to page } i \\ 0, & \text{otherwise} \end{cases}$$

- $L(j)$ is the number of outbound links from page j.

Task:

1. Implement a Python program to construct matrix M from a file describing the web graph.
2. Use the power method to compute the dominant eigenvector (PageRank scores).
3. Sort and print the pages in descending order of importance.

Test Case:

`graph0.txt:`

```
1 2
1 3
1 4
2 3
2 4
3 1
4 1
4 3
```

Compare your results with 'networkx.pagerank(G, d)'.

3 SVD Analysis & Image Reconstruction

Singular Value Decomposition (SVD) decomposes a matrix A into:

$$A = U \Sigma V^T$$

We approximate A using only the top k singular values:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

where A_k is the best rank-k approximation of A.

Task:

1. Implement a Python program to:
   - Read an image as a matrix.
   - Compute its SVD.
   - Reconstruct the image using k singular values.
2. Compute error: $\epsilon_k = \|A - A_k\|_2$.

3. Plot:
    - The sequence (k, ϵ_k).
    - Singular values σ_i.
    - The original and reconstructed images.

Test Images:

- ‘uoc$_l$*ogo.png*“*python$_l$ogo.png*‘
  Use ‘matplotlib.image.imread‘ for image reading and ‘numpy.linalg.svd‘ for SVD computation.

4 Deliverables

Submit your Python implementations and a report including:

1. Code explanations.
2. Results (convergence analysis, rankings, reconstructed images).
3. Comparisons with theoretical expectations.

Good luck!