pagerankalgo

October 4, 2024

[1]:
```
import numpy as np

def pagerank(links, d=0.85, num_iterations=100, tolerance=1.0e-6):
    pages = list(links.keys())
    num_pages = len(pages)
    pageranks = {page: 1.0 / num_pages for page in pages}
    link_matrix = np.zeros((num_pages, num_pages))
    page_index = {page: idx for idx, page in enumerate(pages)}

    for page, outgoing_links in links.items():
        if len(outgoing_links) == 0:
            link_matrix[page_index[page], :] = 1.0 / num_pages
        else:
            for link in outgoing_links:
                if link in page_index:
                    link_matrix[page_index[page], page_index[link]] = 1.0 /␣
 ↪len(outgoing_links)

    teleport_matrix = np.ones((num_pages, num_pages)) / num_pages
    transition_matrix = d * link_matrix + (1 - d) * teleport_matrix

    for i in range(num_iterations):
        old_pageranks = pageranks.copy()
        for page in pages:
            rank_sum = 0
            for other_page in pages:
                rank_sum += old_pageranks[other_page] *␣
 ↪transition_matrix[page_index[other_page], page_index[page]]
            pageranks[page] = rank_sum

        if all(abs(pageranks[page] - old_pageranks[page]) < tolerance for page␣
 ↪in pages):
            print(f"Converged after {i+1} iterations.")
            break

    return pageranks
```

```python
if __name__ == "__main__":
    links = {
        "A": ["B", "C"],
        "B": ["C", "D"],
        "C": ["A"],
        "D": ["C"],
        "E": ["D"],
    }

    ranks = pagerank(links)

    print("\n--- PageRank Results ---")
    for page, rank in sorted(ranks.items(), key=lambda x: -x[1]):
        print(f"Page {page} has rank: {rank:.4f}")
```

```
Converged after 29 iterations.

--- PageRank Results ---
Page C has rank: 0.3483
Page A has rank: 0.3260
Page B has rank: 0.1686
Page D has rank: 0.1271
Page E has rank: 0.0300
```