06: Introduction to PageRank

Chapter Goal: To understand the core idea behind Google's PageRank algorithm as a prime real-world application of eigenvectors and eigenvalues.

---

1. The Core Idea: Ranking Webpages by Links

- **Goal:** To determine the "importance" or "rank" of every page on the internet.
- **Fundamental Assumption:** A webpage is considered important if many other **important** webpages link to it.
- **The Paradox:** This is a recursive or "chicken-and-egg" problem. The rank of page `A` depends on the rank of page `B`, which might depend back on the rank of page `A`.

---

2. A Simple Model: "Procrastinating Pat"

- **Analogy:** Imagine a web surfer ("Pat") who only browses by clicking on links randomly.
- **Hypothesis:** The most important webpages are the ones where Pat will spend the most time in the long run.
- **The Rank Vector (`r`):** The rank vector $\vec{r}$ will contain the probability of Pat being on each page at a given time. Its components will sum to 1.

---

3. Building the Link Matrix (`L`)

- **Goal:** To create a "machine" (a matrix) that can simulate one random click-step from Pat.
- **How to Build It:**
  1. For each page (e.g., Page `A`), create an "outgoing link vector".
  2. If `A` links to `B`, `C`, and `D` (3 total links), then its probability vector is `[0, 1/3, 1/3, 1/3]` (for pages A, B, C, D respectively). A has a 1/3 chance of going to B, 1/3 to C, and 1/3 to D.
  3. Collect all of these outgoing link vectors as the **COLUMNS** of a matrix `L`.
- **What does `L` mean?**
  - Matrix `L` is a **"probability transition machine"**.
  - $$\vec{r}_{\text{new}} = L \cdot \vec{r}_{\text{old}}$$
  - If $\vec{r}_{\text{old}}$ is the current probability distribution of Pat, multiplying it by `L` tells us the probability distribution of Pat after one random click.

4. "Aha!" Moment: The Connection to Eigenvectors

- **The Logic of a Stable Rank:** The "stable" ranking of the internet is achieved when the probability distribution of Pat **no longer changes** after he clicks a link.
- **The Steady-State Condition:**

$$\vec{r}_{\text{stable}} = L \cdot \vec{r}_{\text{stable}}$$

- **Reading This Equation:**
  - Look closely: $L\vec{r} = 1 \cdot \vec{r}$.
  - This is the **DEFINITION** of an eigenvector!
  - $\vec{r}_{\text{stable}}$ is an **eigenvector** of the link matrix `L`.
  - The corresponding **eigenvalue** is `λ = 1`.
- **Extraordinary Conclusion:**

> The problem of "finding the most important webpages" is identical to the problem of "finding the eigenvector of the link matrix with an eigenvalue of 1".

---

5. The Solution Method: Power Iteration

- **Problem:** We can't use diagonalization (CDC^{-1}) to find the eigenvector, because the eigenvector is what we are trying to find in the first place.
- **Iterative Method (Power Iteration):**
  1. Start with an initial guess for `r` (e.g., all pages are equally important: $\vec{r}_0 = [1/4, 1/4, 1/4, 1/4]$).
  2. Repeatedly apply the transformation:
     $\vec{r}_{\text{new}} = L \cdot \vec{r}_{\text{old}}$
     $\vec{r}_1, \vec{r}_2, \vec{r}_3, \ldots$
  3. Do this many times until the vector `r` stops changing significantly (it **converges**).
  4. The final vector `r` is the eigenvector we are looking for.
- **Why is this effective for PageRank?**
  - This method will naturally converge to the **dominant eigenvector** (the one with the largest eigenvalue).
  - Due to the way we construct matrix `L` (as a stochastic matrix), it can be proven that its largest eigenvalue is **always 1**, which is exactly what we need.
  - For the real internet, the matrix `L` is very **"sparse"** (almost all of its entries are zero), making the matrix-vector multiplication very efficient.
- **Additional Detail:**
  - **Damping Factor (`d`):** In the original PageRank algorithm, a small modification is added to model the chance of Pat getting "bored" and typing a new web address

randomly, instead of always clicking a link. This helps with the stability and speed of convergence.

Tags: #mml-specialization #linear-algebra #eigenvectors #pagerank #power-iteration