CSCI 145 Problem 4: How Many Iterations Does the Power Method Need?

July 20, 2026

Submission Instructions

Please upload your work to Gradescope by **11:59pm on DATE**.

- Turn in a single PDF of *handwritten* solutions. Typed (including LaTeX) solutions will not be accepted.
- Problems are grouped into three-week **cycles**. Once per cycle, you will meet with the grader and present the solution to *one* problem from that cycle, selected at random when you arrive — so bring your work and be ready to explain any of them.

Problem 4: How Many Iterations Does the Power Method Need?

In the class demo, the power method's error fell along a straight line on a log-scale plot, and we predicted the line's slope before running a single iteration. This problem proves that prediction and turns it into an iteration count: exactly how many multiplications by $\mathbf{A}$ does accuracy ϵ take? The tool is the one the lecture leaned on all day: expand in the eigenbasis, and watch each coefficient scale.

Throughout, let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be symmetric with eigendecomposition

$$\mathbf{A} = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top,$$

where the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ are orthonormal and the eigenvalues satisfy $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ (note the strict gap at the top). The power method starts from a unit vector $\mathbf{v}^{(0)} = \sum_{i=1}^{d} c_i \mathbf{v}_i$ with $c_1 = \langle \mathbf{v}_1, \mathbf{v}^{(0)} \rangle \neq 0$ and repeatedly multiplies by $\mathbf{A}$, normalizing as it goes. Since normalization changes only the length, the *direction* after k iterations is the direction of $\mathbf{A}^k \mathbf{v}^{(0)}$, so that is the vector we analyze.

Part A: Powers in the Eigenbasis

In lecture, we multiplied two outer-product sums and used orthonormality to collapse the cross terms. Use the same move on $\mathbf{A} \cdot \mathbf{A}$ to show

$$\mathbf{A}^2 = \sum_{i=1}^{d} \lambda_i^2 \, \mathbf{v}_i \mathbf{v}_i^\top,$$

and explain in one sentence why repeating the argument gives $\mathbf{A}^k = \sum_{i=1}^{d} \lambda_i^k \, \mathbf{v}_i \mathbf{v}_i^\top$ for every $k \geq 1$. Conclude that

$$\mathbf{A}^k \mathbf{v}^{(0)} = \sum_{i=1}^{d} \lambda_i^k c_i \, \mathbf{v}_i.$$

Powering a matrix is expensive and powering its eigenvalues is free, which is most of the reason decompositions are worth computing.

Part B: The Contraction

How far is $\mathbf{A}^k \mathbf{v}^{(0)}$ from pointing exactly along $\mathbf{v}_1$? As in lecture, let θ_k be the angle between them, and measure it by

$$\tan \theta_k = \frac{\left\| \text{component of } \mathbf{A}^k \mathbf{v}^{(0)} \text{ orthogonal to } \mathbf{v}_1 \right\|_2}{\left| \text{component of } \mathbf{A}^k \mathbf{v}^{(0)} \text{ along } \mathbf{v}_1 \right|} \qquad \text{(opposite over adjacent).}$$

Using Part A and the orthonormality of the eigenvectors, show that

$$\tan^2 \theta_k = \frac{\sum_{i=2}^{d} \left(\lambda_i^k c_i \right)^2}{\left(\lambda_1^k c_1 \right)^2},$$

then bound $\lambda_i \le \lambda_2$ for every $i \ge 2$ to conclude the error contracts geometrically:

$$\tan\theta_k \le \left(\frac{\lambda_2}{\lambda_1}\right)^k \tan\theta_0.$$

Sanity check against the in-class exercise: for $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ and $\mathbf{v}^{(0)} = (1, 0)$, we computed $\mathbf{A}^2\mathbf{v}^{(0)} \propto (5, 4)$ by hand. Verify that its $\tan\theta_2$ is exactly $(\lambda_2/\lambda_1)^2 = \frac{1}{9}$.

Part C: The Iteration Count

Now the payoff. Suppose we want $\tan\theta_k \le \epsilon$. Show that it suffices to run

$$k \;\ge\; \frac{\log(\tan\theta_0/\epsilon)}{\log(\lambda_1/\lambda_2)}$$

iterations, and simplify to $k \approx \log(1/\epsilon)/\log(\lambda_1/\lambda_2)$ for a generic start with $\theta_0 = 45°$. (Since both logarithms sit in a ratio, the base does not matter.) Then put in numbers, using $\epsilon = 10^{-6}$ and $\theta_0 = 45°$ in both cases:

- the demo's matrix, with $\lambda_1 = 3$ and $\lambda_2 = 1$, and compare your count against the straight line from the demo;
- a matrix with a narrow eigengap, $\lambda_2/\lambda_1 = 0.99$.

Each iteration costs one matrix–vector product, $O(d^2)$ time, while computing the full eigendecomposition costs $O(d^3)$. Using your two counts, conclude in one sentence when the power method is the right tool, and why the eigengap λ_2/λ_1, rather than the size of the matrix, is its clock speed.

Where this goes next. This problem is the first payoff of the course's spectral thread. The "expand in the eigenbasis and let each coefficient scale" move returns when we analyze gradient descent, where the ratio of the largest eigenvalue to the smallest (the *condition number*) will set the pace of optimization exactly the way λ_1/λ_2 sets the pace here.