Power iteration

From Wikipedia, the free encyclopedia

In mathematics, the **power iteration** is an eigenvalue algorithm: given a matrix *A*, the algorithm will produce a number λ (the eigenvalue) and a nonzero vector *v* (the eigenvector), such that *Av* = λ*v*. The algorithm is also known as the Von Mises iteration.[1]

The power iteration is a very simple algorithm. It does not compute a matrix decomposition, and hence it can be used when *A* is a very large sparse matrix. However, it will find only one eigenvalue (the one with the greatest absolute value) and it may converge only slowly.

Contents [hide]

The method [edit]

The power iteration algorithm starts with a vector b_0, which may be an approximation to the dominant eigenvector or a random vector. The method is described by the recurrence relation

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}.$$

So, at every iteration, the vector b_k is multiplied by the matrix *A* and normalized.

If we assume *A* has an eigenvalue that is strictly greater in magnitude than its other eigenvalues and the starting vector b_0 has a nonzero component in the direction of an eigenvector associated with the dominant eigenvalue, then a subsequence (b_k) converges to an eigenvector associated with the dominant eigenvalue.

Without the two assumptions above, the sequence (b_k) does not necessarily converge. In this sequence,

$$b_k = e^{i\phi_k} v_1 + r_k,$$

where v_1 is an eigenvector associated with the dominant eigenvalue, and $\|r_k\| \rightarrow 0$. The presence of the term $e^{i\phi_k}$ implies that (b_k) does not converge unless $e^{i\phi_k} = 1$. Under the two assumptions listed above, the sequence (μ_k) defined by

$$\mu_k = \frac{b_k^* A b_k}{b_k^* b_k}$$

converges to the dominant eigenvalue.

This can be run as a simulation program with the following simple algorithm:

```
for each(''simulation'') {
    // calculate the matrix-by-vector product Ab
    for(i=0; i<n; i++) {
         tmp[i] = 0;
         for (j=0; j<n; j++)
              tmp[i] += A[i][j] * b[j];
                        // dot product of i-th row in A with the column vector b
    }

    // calculate the length of the resultant vector
    norm_sq=0;
    for (k=0; k<n; k++)
         norm_sq += tmp[k]*tmp[k];
    norm = sqrt(norm_sq);
```

```
        // normalize b to unit vector for next iteration
        b = tmp/norm;
    }
```

The value of *norm* converges to the absolute value of the dominant eigenvalue, and the vector *b* to an associated eigenvector.

Note: The above code assumes real A,b. To handle complex; A[i][j] becomes conj(A[i][j]), and tmp[k]*tmp[k] becomes conj(tmp[k])*tmp[k]

This algorithm is the one used to calculate such things as the Google PageRank.

The method can also be used to calculate the spectral radius of a matrix by computing the Rayleigh quotient

$$\frac{b_k^\top A b_k}{b_k^\top b_k} = \frac{b_{k+1}^\top b_k}{b_k^\top b_k}.$$

Analysis [edit]

Let A be decomposed into its Jordan canonical form: $A = VJV^{-1}$, where the first column of V is an eigenvector of A corresponding to the dominant eigenvalue λ_1. Since the dominant eigenvalue of A is unique, the first Jordan block of J is the 1×1 matrix $[\lambda_1]$, where λ_1 is the largest eigenvalue of *A* in magnitude. The starting vector b_0 can be written as a linear combination of the columns of V: $b_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$. By assumption, b_0 has a nonzero component in the direction of the dominant eigenvalue, so $c_1 \neq 0$.

The computationally useful recurrence relation for b_{k+1} can be rewritten as:

$b_{k+1} = \frac{Ab_k}{\|Ab_k\|} = \frac{A^{k+1}b_0}{\|A^{k+1}b_0\|}$, where the expression: $\frac{A^{k+1}b_0}{\|A^{k+1}b_0\|}$ is more amenable to the following analysis.

$$\begin{aligned} b_k &= \frac{A^k b_0}{\|A^k b_0\|} \\ &= \frac{\left(VJV^{-1}\right)^k b_0}{\|(VJV^{-1})^k b_0\|} \\ &= \frac{VJ^kV^{-1}b_0}{\|VJ^kV^{-1}b_0\|} \\ &= \frac{VJ^kV^{-1}(c_1v_1+c_2v_2+\cdots+c_nv_n)}{\|VJ^kV^{-1}(c_1v_1+c_2v_2+\cdots+c_nv_n)\|} \\ &= \frac{VJ^k(c_1e_1+c_2e_2+\cdots+c_ne_n)}{\|VJ^k(c_1e_1+c_2e_2+\cdots+c_ne_n)\|} \\ &= \left(\frac{\lambda_1}{|\lambda_1|}\right)^k \frac{c_1}{|c_1|} \frac{v_1+\frac{1}{c_1}V\left(\frac{1}{\lambda_1}J\right)^k(c_2e_2+\cdots+c_ne_n)}{\|v_1+\frac{1}{c_1}V\left(\frac{1}{\lambda_1}J\right)^k(c_2e_2+\cdots+c_ne_n)\|} \end{aligned}$$

The expression above simplifies as $k \to \infty$

$$\left(\frac{1}{\lambda_1}J\right)^k = \begin{bmatrix} [1] & & & \\ & \left(\frac{1}{\lambda_1}J_2\right)^k & & \\ & & \ddots & \\ & & & \left(\frac{1}{\lambda_1}J_m\right)^k \end{bmatrix} \to \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} \text{ as } k \to \infty.$$

The limit follows from the fact that the eigenvalue of $\frac{1}{\lambda_1}J_i$ is less than 1 in magnitude, so $\left(\frac{1}{\lambda_1}J_i\right)^k \to 0$ as $k \to \infty$

It follows that:

$$\frac{1}{c_1}V\left(\frac{1}{\lambda_1}J\right)^k(c_2e_2+\cdots+c_ne_n) \to 0 \text{ as } k \to \infty$$

Using this fact, b_k can be written in a form that emphasizes its relationship with v_1 when k is large:

$$b_k = \left(\frac{\lambda_1}{|\lambda_1|}\right)^k \frac{c_1}{|c_1|} \frac{v_1+\frac{1}{c_1}V\left(\frac{1}{\lambda_1}J\right)^k(c_2e_2+\cdots+c_ne_n)}{\|v_1+\frac{1}{c_1}V\left(\frac{1}{\lambda_1}J\right)^k(c_2e_2+\cdots+c_ne_n)\|} = e^{i\phi_k}\frac{c_1}{|c_1|}v_1 + r_k \text{ where}$$

$e^{i\phi_k} = (\lambda_1/|\lambda_1|)^k$ and $\|r_k\| \to 0$ as $k \to \infty$

The sequence (b_k) is bounded, so it contains a convergent subsequence. Note that the eigenvector corresponding to the dominant eigenvalue is only unique up to a scalar, so although the sequence (b_k) may not converge, b_k is nearly an eigenvector of *A* for large k.

Alternatively, if A is diagonalizable, then the following proof yields the same result

Let $\lambda_1, \lambda_2, \ldots, \lambda_m$ be the m eigenvalues (counted with multiplicity) of A and let $v_1, v_2, \ldots, v_m$ be the corresponding eigenvectors. Suppose that λ_1 is the dominant eigenvalue, so that $|\lambda_1| > |\lambda_j|$ for $j > 1$.

The initial vector b_0 can be written:

$$b_0 = c_1 v_1 + c_2 v_2 + \cdots + c_m v_m.$$

If b_0 is chosen randomly (with uniform probability), then $c_1 \neq 0$ with probability 1. Now,

$$\begin{aligned} A^k b_0 &= c_1 A^k v_1 + c_2 A^k v_2 + \cdots + c_m A^k v_m \\ &= c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_m \lambda_m^k v_m \\ &= c_1 \lambda_1^k \left(v_1 + \frac{c_2}{c_1} \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + \frac{c_m}{c_1} \left(\frac{\lambda_m}{\lambda_1}\right)^k v_m \right). \end{aligned}$$

The expression within parentheses converges to v_1 because $|\lambda_j/\lambda_1| < 1$ for $j > 1$. On the other hand, we have

$$b_k = \frac{A^k b_0}{\|A^k b_0\|}.$$

Therefore, b_k converges to (a multiple of) the eigenvector v_1. The convergence is geometric, with ratio

$$\left| \frac{\lambda_2}{\lambda_1} \right|,$$

where λ_2 denotes the second dominant eigenvalue. Thus, the method converges slowly if there is an eigenvalue close in magnitude to the dominant eigenvalue.

Applications [edit]

Although the power iteration method approximates only one eigenvalue of a matrix, it remains useful for certain computational problems. For instance, Google uses it to calculate the PageRank of documents in their search engine,[2] and Twitter uses it to show users recommendations of who to follow.[3] For matrices that are well-conditioned and as sparse as the Web matrix, the power iteration method can be more efficient than other methods of finding the dominant eigenvector.

Some of the more advanced eigenvalue algorithms can be understood as variations of the power iteration. For instance, the inverse iteration method applies power iteration to the matrix A^{-1}. Other algorithms look at the whole subspace generated by the vectors b_k. This subspace is known as the Krylov subspace. It can be computed by Arnoldi iteration or Lanczos iteration. Another variation of the power method that simultaneously gives n eigenvalues and eigenfunctions, as well as accelerated convergence as $|\lambda_{n+1}/\lambda_1|$, is "Multiple extremal eigenpairs by the power method" in the Journal of Computational Physics Volume 227 Issue 19, October, 2008, Pages 8508-8522 (Also see pdf below for Los Alamos National Laboratory report LA-UR-07-4046)

See also [edit]

- Rayleigh quotient iteration
- Inverse iteration

References [edit]

1. ^ Richard von Mises and H. Pollaczek-Geiringer, *Praktische Verfahren der Gleichungsauflösung*, ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik 9, 152-164 (1929).
2. ^ Ipsen, Ilse, and Rebecca M. Wills (5–8 May 2005). "7th IMACS International Symposium on Iterative Methods in Scientific Computing" (PDF). Fields Institute, Toronto, Canada.
3. ^ Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF: The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web

External links [edit]

- Power method, part of lecture notes on numerical linear algebra by E. Bruce Pitman, State University of New York.
- Module for the Power Method
- [1] Los Alamos report LA-UR-07-4046 ""Multiple extremal eigenpairs by the power method"

v · t · e	Numerical linear algebra [hide]
Key concepts	Floating point · Numerical stability
Problems	Matrix multiplication (algorithms) · Matrix decompositions · Linear equations · Sparse problems
Hardware	CPU cache · TLB · Cache-oblivious algorithm · SIMD · Multiprocessing
Software	BLAS · Specialized libraries · General purpose software

Categories: Numerical linear algebra