How to Compute Eigenvalues & Eigenvectors

Zvi Rosen
Department of Mathematics

October 26, 2016

Idea

We want to solve $Ax = \lambda x$ for a square $n \times n$ matrix A, a column vector x and a scalar λ.
λ is an eigenvalue of A, and x is an eigenvector of A.

Power method

The power method is an iterative algorithm that uses $Ax = \lambda x$ to find the **largest** eigenvalue of A.
Idea: Assume that all vectors $v \in \mathbb{R}^n$ can be expressed as a weighted sum of eigenvectors. [Not always true, but we can work in this case.] Order the eigenvalues as $|\lambda_1| \geq \cdots \geq |\lambda_n|$, and associated eigenvectors $x_1, \ldots, x_n$.

Power method II

Then if v is chosen at random, we will have
$v = a_1 x_1 + \cdots + a_n x_n$, with all a_i's nonzero, and
$Av = a_1 Ax_1 + \cdots + a_n Ax_n$ by linearity.
$\Rightarrow Av = a_1\lambda_1 x_1 + \cdots + a_n\lambda_n x_n \Rightarrow A^N v = a_1\lambda_1^N x_1 + \cdots + a_n\lambda_n^N x_n$.
If you normalize this vector, you obtain

$$\frac{A^N v}{||A^N v||} = C\left(x_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^N x_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^N x_N\right).$$

As N gets large, x_1 dominates the right-hand side.

MATLAB demonstration

We use the following loop to execute the power method until we have sufficiently low error:

```
while err > .001
w = A*x/norm(A*x);
err = norm(w - x)/norm(x);
x = w;
end
```

Deflation for Symmetric Matrices

Symmetric matrices have a special property that all eigenvectors are orthogonal to each other. I.e. the dot product of two eigenvectors is zero.
Once we have the largest eigenvalue λ_1 for A, and its corresponding eigenvector x_1, we can redefine the matrix:

$$B = A - \lambda_1 x_1 x_1^T.$$

This new matrix has all the same eigenvalues, except λ_1 is replaced with zero. Repeating the power method here finds λ_2.

Polynomial method

The equation $Ax = \lambda x$ can be reformulated as:

$$(A - \lambda I)x = 0.$$

Since multiplying $A - \lambda I$ by nonzero x gives 0, this means that the matrix $A - \lambda I$ is not invertible. If it were, we would have

$$(A - \lambda I)^{-1}(A - \lambda I)x = (A - \lambda I)^{-1}0 \iff x = 0.$$

We know that a matrix has no inverse if and only if its determinant is zero, so we can think instead about:

$$\det(A - \lambda I) = 0.$$

This is a degree n polynomial in λ.

Polynomials & Eigenvalues in MATLAB

MATLAB can compute eigenvalues much quicker than it can solve polynomials. In fact, it solves polynomials using eigenvalue computation. If

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

Then the eigenvalues of the matrix

$$\begin{pmatrix} \frac{a_{n-1}}{a_n} & \cdots & \frac{a_1}{a_n} & \frac{a_0}{a_n} \\ 1 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & \cdots & 1 & 0 \end{pmatrix}$$

are the roots of the polynomial $p(x)$.

MATLAB's `eig` function

`eig` uses the QR decomposition of a matrix which factors a matrix as the product of an orthogonal matrix and an upper-triangular matrix.
It computes a sequence of decompositions until the upper-triangular matrix has the eigenvalues on the diagonal.
The details of the algorithm will not be covered on the quiz (but would make a nice research project!)