2010

Reverse-Engineering Linear Algebra

Billy Wonderly
University of Puget Sound

Reverse-Engineering Linear Algebra

Billy Wonderly

Department of Mathematics and Computer Science
University of Puget Sound

Introduction

The problem of generating matrices with desirable properties is a difficult one. These matrices are sought after for a variety of reasons, one being students looking for practice in mastering the skills they need to succeed in Linear Algebra. A solution to this problem is to do Linear Algebra in reverse. Theorems and procedures used to produce answers from questions can often in effect be 'undone', meaning that we can use a solution with specific properties, to produce a problem whose answer will retain those properties. This idea was put in motion by developing and modifying Sage, open source mathematics software.

1. Generating Solutions

In reverse engineering problems the first and most important step is finding the ideal solution. In general, solutions in linear algebra are extracted from the reduced row-echelon form of a matrix. The function created allows for inputs of number of rows, columns, and pivot columns for a matrix.

$$\left.\begin{matrix}\text{row=4}\\ \text{col=5}\\ \text{rank=3}\end{matrix}\right\} \Longrightarrow \begin{bmatrix} 1 & 0 & 5 & 0 & 4 \\ 0 & 1 & -3 & 0 & 2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Matrices are in reduced row-echelon form.
- Output matrices have only integer entries.
- Leading one's are placed skewed to the left side of the matrix.
- Matrices can be generated over a variety of finite algebraic rings.

2. Matrices with nice Echelon Forms

Another In order to get a matrix into rref, if not careful, the computations necessary can produce very ugly and hard to work with fractions. A matrix with a desirable reduced row-echelon form is achieved by 'reversing' an algorithm to put a matrix in reduced row-echelon form.

$$A = \begin{bmatrix} 1 & 0 & 5 & 0 & 4 \\ 0 & 1 & -3 & 0 & 2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{Reverse RREF}} \begin{bmatrix} -1 & -6 & 23 & -8 & -24 \\ -1 & 6 & -23 & 10 & -28 \\ 2 & -9 & 37 & -16 & -42 \\ 0 & 1 & -3 & -1 & 0 \end{bmatrix}$$

$$\xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 5 & 0 & 4 \\ 0 & 1 & -3 & 0 & 2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = A$$

- Uses the output of the routine 1.
- Reverses the rref process by randomly performing addition of the scalar multiple of one row to another row of the matrix.
- Has an additional entry size control feature because a large number of scalar row multiples of rows being added together can make entry sizes very large and unmanageable.
- Matrices can be generated over a variety of finite algebraic rings.

3. Unimodular Matrices

Unimodular matrices are nonsingular and have the special property that their determinant is one. This makes many computations turn out very nicely.

$$A = \begin{bmatrix} -3 & -15 & 4 & 4 \\ -3 & -11 & 4 & 3 \\ -5 & -24 & 2 & 19 \\ -4 & -19 & 5 & 6 \end{bmatrix}$$

$$det(A) = 1$$

- This funcion is essentially is the a specific case of the routine 2 where the input is a square, full rank, matrix.
- Because 2 only uses scalar row addition, even after a large number of row operations, the determinant remains unchanged.
- The series of row operations is performed on the identity matrix, which has determinant one.
- Unimodular matrices are useful because a unimodular matrix that contains only integer entries will have an inverse containing only integer entries.
- Multiplying a matrix B by a unimodular matrix U results in hiding a problem, and can be easily undone bye multiplying the resulting matrix by U^{-1}

4. Random Four Subspaces Routine

The Four Subspaces routine utilizes the 1 and 2 routines to produce matrices whose extended echelon form is 'nice'.

$$\left.\begin{matrix} B = \begin{bmatrix} 1 & 0 & 0 & 3 & 1 \\ 0 & 1 & 0 & -1 & -2 \\ 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ K = \begin{bmatrix} 1 & 10 & -47 \\ -1 & 3 & -15 \\ 0 & 4 & -19 \end{bmatrix} \\ L = \begin{bmatrix} 1 & -4 & -2 & 0 \end{bmatrix} \end{matrix}\right\} \Longrightarrow \left[\begin{array}{ccccc|cccc} 1 & 0 & 5 & 0 & 4 & 0 & 1 & 10 & -47 \\ 0 & 1 & -3 & 0 & 2 & 0 & -1 & 3 & -15 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 4 & -19 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & -4 & -2 & 0 \end{array}\right]$$

- All submatrices contain only integer entries.
- The B and L matrices are generated by the 1 routine, and the K matrix is called from the 3 routine. These matrices are then pushed together to produce the solution.
- The matrix formed by stacking K and L is created to be unimodular because its inverse is multiplied on the left side of B. This matrix product is the output of the routine.

5. Eigenspaces

The final routine generates a diagonalizable matrix whose eigenspaces have nice basis vectors if computed by hand. Inputs of matrix size, eigenvalues, and associated eigenspace dimensions are allowable.

$$\left.\begin{matrix}\text{size=4}\\ \text{eigenvalues=[-2,1,4]}\\ \text{dimensions=[1,1,2]}\end{matrix}\right\} \Longrightarrow A = \begin{bmatrix} -20 & 18 & -36 & -18 \\ -6 & 16 & 6 & 18 \\ 12 & -3 & 34 & 27 \\ -6 & -3 & -24 & -23 \end{bmatrix}$$

Characteristic polynomial: $p_A(x) = (x+2)(x-1)(x-4)^2$

Matrix of eigenvectors: $S = \begin{bmatrix} 4 & -6 & -3 & -3 \\ 1 & -4 & -2 & -3 \\ -2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

Giving: $S^{-1}AS = D = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$

- Built by reversing the similarity transformation $D = S^{-1}AS$ to give $SDS^{-1} = A$
- Eigenvectors are given a nice pattern of zeros and ones to show linear independence.
- Matrices built from eigenvectors are determinant one to ensure all matrix inverses and products of matrix multiplication contain only integers.
- To preserve determinant one, the matrix of eigenvectors was build from the identity matrix using addition of both column and row multiples.
- Matrices with complex eigenvalues cannot be produced.

Acknowledgements

This research was made possible by the University of Puget Sound Summer Research Grants in Science and Mathematics. I would like to thank Professor Robert Beezer for his help throughout this project. I would also like to thank all of the Sage developers who devote their time and effort to developing free and open source mathematics software.