Sage and Linear Algebra Worksheet

FCLA Section EE

Robert Beezer
Department of Mathematics and Computer Science
University of Puget Sound

Spring 2019

1 Eigenvalues and Eigenvectors

A 6×6 matrix with "nice" eigenvalues.

```
A = matrix(QQ, [
    [-31, -23, -16,  12, 120, -17],
    [ -3,   7,   0, -12,  60, -21],
    [-28, -14,  -9,  -4, 152, -30],
    [-36, -20, -16,  -1, 192, -32],
    [ -9,  -5,  -4,   0,  47,  -8],
    [ -1,   1,   0,  -4,  20,  -3]
    ])
A
```

```
p = A.characteristic_polynomial()
p
```

```
p.factor()
```

Eigenvalues are the roots of the characteristic polynomial (Theorem EMRCP), which should be obvious from the factored version, including their (algebraic) multiplicities. Of course, it can be very easy to get these in Sage.

```
A.eigenvalues()
```

Demonstration 1 Create the singular matrices $A-\lambda I_6$ for each eigenvalue (we will choose to do two with "random" choices for the eigenvalue). Row-reducing these matrices will exhibit their nonzero nullity.

```
(A-( )*identity_matrix(6)).rref()
```

```
(A-( )*identity_matrix(6)).rref()
```

Demonstration 2 Examine the eigenspace for the eigenvalue $\lambda = 3$, using Sage's right kernel command and the pivot basis.

Use a basis for the eigenspace to create eigenvectors of `A` for the eigenvalue $\lambda = 3$ at will.

```
E6 = (A-3*identity_matrix(6)).right_kernel(basis='pivot')
E6
```

```
B = E6.basis()
B
```

```
v = *B[0] + *B[1]
v
```

We can check this. Compare Av with $3v$.

```
A*v
```

```
3*v
```

Here's an easy check:

```
A*v - 3*v
```

Can you make more eigenvectors?

2 Eigenspaces, Eigenmatrices

Continuing with `A` from above, we can get eigen-stuff quickly from Sage, once we understand what is really happening (according to the definitions).

As always we want the "right" versions of the relevant commands. Eigenspaces are in the second parts of pairs, where the first part of each pair is the eigenvalue. Notice that they are vector spaces (with bases, etc). The basis vectors are Sage's version of a basis, with vectors from an echelonized matrix, typically with lots of zeros and ones in the first part of the vectors.

```
A.eigenspaces_right()
```

The `eigenmatrix` commands return pair of matrices. The first is a diagonal matrix with the eigenvalues on the diagonal. The second is a square matrix with linearly independent eigenvectors in the columns, and the order of the eigenvectors is the same as the order of the eigenvalues. That is, the eigenvector in column `i` of the second matrix is a basis vector for the eigenspace of the eigenvalue in column `i` of the first matrix.

```
A.eigenmatrix_right()
```

Sometimes the dimension of an eigenspace (the geometric multiplicity) is strictly less than the number of times the eigenvalue appears as a root of the characteristic polynomial. This is the case with `C` next, but was not the case with `A` above.

```
C = matrix(QQ, [
    [128,   20,  44, -50,  236, -18, -330, -565],
    [ -23, -16,  -5,   6,  -40,  27,   62,  128],
    [ -44, -12, -15,  16,  -78,  20,  110,  207],
    [ -2,   10,  -4,   3,  -10, -23,   20,   -9],
    [ -61,   5, -25,  27, -116, -26,  153,  225],
    [ -12, -12,  -1,   2,  -20,  24,   34,   82],
    [ -23,  -3,  -8,   9,  -42,   2,   57,   99],
    [  13,   6,   3,  -4,   23, -12,  -35,  -72]
    ])
C
```

```
C.eigenmatrix_right()
```

3 Fancy Footwork

A totally random matrix is unlikely to have a characteristic polynomial that factors if we restrict ourseves to the rationals. But we can find all the roots over $\overline{Q}$, the set of all algebraic numbers. (This is the set of all real roots of all possible polynomials with integer coefficients.)

```
D = random_matrix(QQ, 10)
D
```

```
p = D.characteristic_polynomial()
p.factor()
```

```
p.roots(ring=QQbar, multiplicities=False)
```

If we make a "block diagonal" matrix, then the characteristic polynomial will definitely factor some

```
E = block_diagonal_matrix( [random_matrix(QQ, 5),
    random_matrix(QQ, 5)])
E
```

```
p = E.charpoly()
p.factor()
```

Finally a large example, illustrating how fast Sage is at making characteristic polynomials and at factoring.

```
F = block_diagonal_matrix( [random_matrix(QQ, 50),
    random_matrix(QQ, 50)])
p = F.charpoly()
p.factor()
```

This is such a common operation, that Sage has a shorthand method for the **Factored Characteristic Polynomial**, namely `.fcp()`.

```
F.fcp()
```

Nothing short of amazing!

4 Numerical Matrices

If we use `CDF` (Complex Double Field) for the number system of the entries of our matrix, we get (good) approximate values for eigenvalues. (If we are OK with the approximate nature, these routines are very, very fast.)

```
G = random_matrix(QQ, 300)
H = G.change_ring(CDF)
H.eigenvalues()
```