逆行列の計算

経済動学 2016Q1
資料番号: 16EDH2S

mail@**kenjisato**.jp

April 14, 2016

1 逆行列の計算

Python の練習問題として, numpy.linalg.inv を使わずに逆行列を計算してみよう. numpy.ndarray は利用する. ただし, myinv は非効率的なので応用上は役に立たない.

以下のコードは行基本変形の積として正則行列を表現できるという事実を素朴に実装したものである.

```
In [1]: import numpy as np
        EPS = 1e-10
```

1.1 行の交換

```
In [2]: def switch(a, i, j):
            """switch a's i-th and j-th rows"""
            for k, (aik, ajk) in enumerate(zip(a[i], a[j])):
                a[i, k], a[j, k] = ajk, aik

        def test_switch():

            a = np.array([[1, 1, 1], [0, 0, 0], [3, 3, 3]])
            b = np.array([[3, 3, 3], [0, 0, 0], [1, 1, 1]])

            switch(a, 0, 2)
            assert (a == b).all()

        def test_noswitch():
            a = np.array([[1, 1, 1], [0, 0, 0], [3, 3, 3]])
            b = a[:]

            switch(a, 0, 0)
            assert (a == b).all()

        test_switch()
        test_noswitch()
```

1.2 1 行を定数倍して別の行に加える

```
In [3]: def add(a, c, i, j):
            """add i-th row multiplied by c to j-th row"""
            for k, aik in enumerate(a[i]):
                a[j, k] += c * a[i, k]

        def test_add():

            a = np.array([[1, 1, 1], [0, 0, 0], [3, 3, 3]])
            b = np.array([[1, 1, 1], [6, 6, 6], [3, 3, 3]])

            add(a, 2, 2, 1)
            assert (a == b).all()

        test_add()
```

1.3 1 行を定数倍する

```
In [4]: def multiply(a, c, i):
            """multiply i-th row by c != 0"""
            for k in range(len(a[i])):
                a[i, k] *= c

        def test_multiply():
            a = np.array([[1, 1, 1], [0, 0, 0], [3, 3, 3]])
            b = np.array([[1, 1, 1], [0, 0, 0], [30, 30, 30]])

            multiply(a, 10, 2)
            assert (a == b).all()

        test_multiply()
```

1.4 (A, I) に上の変換を適用して (I, A^{-1}) を得る

```
In [5]: def myinv(a):
            """matrix inversion

            INPUT
            -----
            a: nxn list of floats

            Output
            ------
            b: inverse matrix of a
            """

            n = a.shape[0]

            # extended coefficient matrix
            A = np.empty((n, 2*n))
            A[:, 0:n] = a[:]
            A[:, n:] = np.eye(n)

            for icol in range(n):
                irow = icol
                # Make a_ii = 1
                for i in range(irow, n):
                    if np.abs(A[i, icol]) > EPS:
                        c = 1/A[i, icol]
                        multiply(A, c, i)
                        switch(A, irow, i)
                        break

                # Make a_ij = 0 for i != j
                for i in range(n):
                    if not i == irow:
                        c = - A[i, icol]
                        add(A, c, irow, i)

            return A[:, n:]
```

```
In [6]: a = np.diag([1, 3, 3])
```

```
In [7]: myinv(a)
```

```
Out[7]: array([[ 1.        ,  0.        ,  0.        ],
               [ 0.        ,  0.33333333,  0.        ],
               [ 0.        ,  0.        ,  0.33333333]])
```

```
In [8]: size = 100

        b = np.random.random((size, size))
        c = myinv(b)

        np.allclose(b.dot(c), np.eye(size))
```

```
Out[8]: True
```