$\|y\| \left\| \frac{\vec{x}}{\|x\|} \right\| \cos\theta = \langle y_1$

$\|y_1\| = \|y\| \cos\theta =$

$\frac{\|y\| \cos\theta}{\|x\|} \vec{x}$) = vector pro

$\frac{x \cdot y}{|y|} \cdot \frac{y}{|y|} = \frac{x \cdot y}{\|y\|^2} y$

Homework #4 EE16A

#1) a)

$$\vec{y}_1 = \frac{\vec{y} \cdot \vec{v}_1}{\|\vec{y}\|^2} \vec{y}$$

$$\vec{y}_2 = \frac{\vec{y} \cdot \vec{v}_2}{\|\vec{y}\|^2} \vec{y}$$

b) $\|\vec{y}_1\| = \|\vec{y}\| \cos\theta_1$

$\|\vec{y}_2\| = \|\vec{y}\| \cos\theta_2$

c) $\vec{\varepsilon}_1 = \vec{y} - \vec{y}_1$

$\vec{\varepsilon}_1 = \vec{y} - \frac{\vec{y} \cdot \vec{v}_1}{\|\vec{y}\|^2} \vec{y}$

$$\|\vec{\varepsilon}_1\|^2 = \left\| \vec{y} - \frac{\vec{y} \cdot \vec{v}_1}{\|\vec{y}\|^2} \vec{y} \right\|^2$$

$$\|\vec{\varepsilon}_2\|^2 = \left\| \vec{y} - \frac{\vec{y} \cdot \vec{v}_2}{\|\vec{y}\|^2} \vec{y} \right\|^2$$

d) $$\left\| \vec{y} - \frac{\vec{y} \cdot \vec{v}_1}{\|\vec{y}\|^2} \vec{y} \right\|^2 < \left\| \vec{y} - \frac{\vec{y} \cdot \vec{v}_2}{\|\vec{y}\|^2} \vec{y} \right\|^2$$

((A^T) = R(A)

#2

$$A = \begin{bmatrix} 1 & -1 & -3 & 4 \\ 3 & -3 & -5 & 8 \\ 1 & -1 & -1 & 2 \end{bmatrix}$$

a) Column Space

colspace(A) = span {col vectors of A} ← subspace of $\mathbb{R}^m$ ($\mathbb{R}^3$)

$$A = \begin{bmatrix} 1 & -1 & -3 & 4 \\ 3 & -3 & -5 & 8 \\ 1 & -1 & -1 & 2 \end{bmatrix}$$

dim(colspace(A)) = 2

basis for colspace = $\left\{ \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}, \begin{bmatrix} -3 \\ -5 \\ -1 \end{bmatrix} \right\}$

$$\text{colspace}(A) = \left\{ a\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} + b\begin{bmatrix} -3 \\ -5 \\ -1 \end{bmatrix} \;\middle|\; a, b \in \mathbb{R} \right\}$$

b) Null space $N(A) = \{\vec{x} \in \mathbb{R}^4 \mid A\vec{x} = \vec{0}\}$

$$\left[\begin{array}{cccc|c} 1 & -1 & -3 & 4 & 0 \\ 3 & -3 & -5 & 8 & 0 \\ 1 & -1 & -1 & 2 & 0 \end{array}\right]$$

$R3 = R3 - R1$

$$\left[\begin{array}{cccc|c} 1 & -1 & -3 & 4 & 0 \\ 3 & -3 & -5 & 8 & 0 \\ 0 & 0 & 2 & -2 & 0 \end{array}\right]$$

$R2 = R2 - 3(R1)$

$$\left[\begin{array}{cccc|c} 1 & -1 & -3 & 4 & 0 \\ 0 & 0 & 4 & -4 & 0 \\ 0 & 0 & 2 & -2 & 0 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & -1 & -3 & 4 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

dim(null(A)) = 2

$x_1 \quad x_2 \quad x_3 \quad x_4$

$x_1 - x_2 - 3x_3 + 4x_4 = 0$

$x_3 = x_4$

$x_1 = x_2 - x_4$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} -1 \\ 0 \\ 1 \\ 1 \end{bmatrix} x_4$$

$$\text{nullspace}(A) = \left\{ a\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + b\begin{bmatrix} -1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \;\middle|\; a, b \in \mathbb{R} \right\}$$

c) Row space

rowspace(A) = colspace(A^T)

$$A^T = \begin{bmatrix} 1 & 3 & 1 \\ -1 & -3 & -1 \\ -3 & -5 & -1 \\ 4 & 8 & 2 \end{bmatrix} \sim$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ -3 & -5 & -1 \\ 4 & 8 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ 0 & 4 & 2 \\ 4 & 8 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 4 & 2 \\ 0 & 0 & 0 \\ 4 & 8 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -2 & -1 \end{bmatrix}$$

$$\text{row}(A) = \left\{ a\begin{bmatrix} 1 \\ -1 \\ -3 \\ 4 \end{bmatrix} + b\begin{bmatrix} 3 \\ -3 \\ -5 \\ 8 \end{bmatrix} \;\middle|\; a, b \in \mathbb{R} \right\}$$

d) Left nullspace

(aka nullspace of A^T)

$$A^T = \left[\begin{array}{ccc|c} 1 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -2 & -1 & 0 \end{array}\right]$$

$x_1 \quad x_2 \quad x_3$ (free var)

$x_2 = -\frac{1}{2}x_3$

$x_1 = -3x_2 - x_3$

$x_3 = x_3$

$x_1 = \frac{1}{2}x_3$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_3 \begin{bmatrix} 1/2 \\ -1/2 \\ 1 \end{bmatrix}$$

$$N(A^T) = \text{span}\left\{ \begin{bmatrix} 1/2 \\ -1/2 \\ 1 \end{bmatrix} \right\}$$

$A\vec{v} = \lambda\vec{v}$

$$\begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} \vec{v}_1^T \\ \vdots \\ \vec{v}_n^T \end{bmatrix} = m \times m$$

$m \times n$ $\quad n \times n$ $\quad n \times m$

#3 a)

$$U = \begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_n \end{bmatrix} \quad V = \begin{bmatrix} \vec{v}_1^T \\ \vdots \\ \vec{v}_n^T \end{bmatrix} \quad \Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

$m \times n$ $\quad n \times m$ $\quad n \times n$

b) 400 eigenvalues, but since the image is symmetric we'll only need 200.

c) [In ipynb]

d) I'd say the value of k can't dip below 20 w/out losing significant quality.

#4 a) $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ b) one hop $A^1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 1 one-hop path

$$A^2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(a→a, b→a, a→b, b→b)

0 two-hop paths

$$A^3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

(a→b)

1 3-hop path

c) $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

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

$\det\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = 0$

$\det\begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix} = 0$

$\lambda^2 - 1 = 0$

$\lambda = 1, -1$

(1)

$(A - \lambda I)\vec{x} = \vec{0}$

$$\left[\begin{array}{cc|c} -1 & 0 & 0 \\ 0 & -1 & 0 \end{array}\right]$$

d)

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

e)

$$B^2 = \begin{bmatrix} 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

1 ~~two~~-hop path 1→3

$$B^3 = \begin{bmatrix} 2 & 1 & 3 & 3 \\ 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 2 & 0 & 3 & 2 \end{bmatrix}$$

1 3-hop path 1→2

#4 Cont.

f) $B_{Transition} = \begin{bmatrix} 0 & 1 & 1/3 & 1/3 \\ 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 \\ 1 & 0 & 1/3 & 0 \end{bmatrix}$

used iPython

if $\lambda = 1$, Eigen vector = $[-0.614 \quad -0.307 \quad -0.230 \quad -0.690]^T$

Scaled → $[0.334 \quad 0.167 \quad 0.125 \quad 0.375]$

g) $C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$

h) No paths from 1 → 3, obviously.

i) $C_{transition} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1 & 0 \end{bmatrix}$

λ values: $1, 1, -1, -0.5+0.5j, -0.5-0.5j$ (iPython)

Eigen vectors: $[0.7 \quad 0.7 \quad 0 \quad 0 \quad 0]^T$ $[0 \quad 0 \quad 0.6 \quad 0.3 \quad 0.6]^T$

w/ $\lambda = 1$ ↓ ↓

$[0.5 \quad 0.5 \quad 0 \quad 0 \quad 0]^T$ $[0 \quad 0 \quad 0.4 \quad 0.2 \quad 0.4]^T$

#5 Dhruv Kathuria # 3031790620
Divya Vijayan # 3032175290

We really collaborated on the first and third questions. They taught me about eigenvalues.

EE16A Homework 4

Image Compression

In [1]:
```
%pylab inline
```

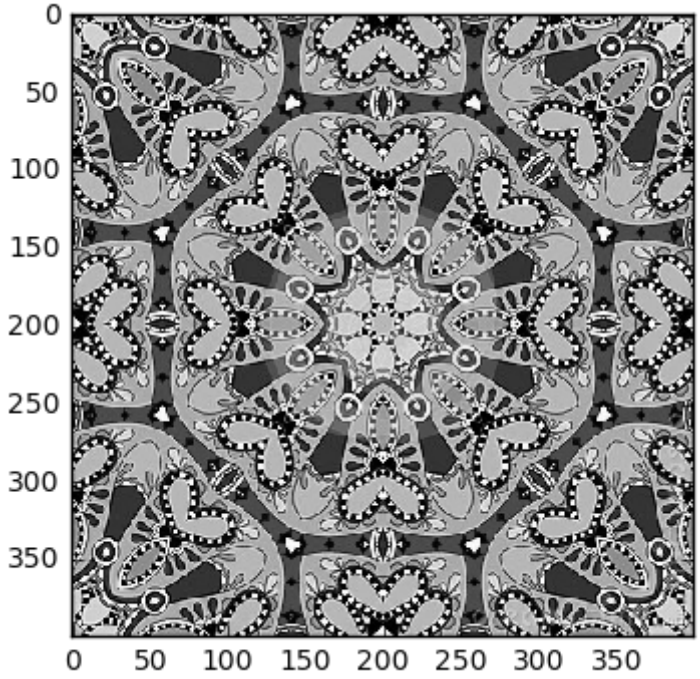
```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import numpy as np
from scipy import ndimage as nd
from scipy import misc
from scipy import io
```

Part b

In [3]:
```
#Load Pattern Image
pattern = np.load('pattern.npy')
plt.imshow(pattern, cmap='gray', interpolation='nearest')
```

Out[3]: <matplotlib.image.AxesImage at 0x1108b7320>



Use the command shape (http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html) to find the dimensions of the image. How many eigenvalues do you expect?

Run the code below to find the eigenvector and eigenvalues of `pattern` and sort them in descending order (first eigenvalue/vector corresponds to the largest eigenvalue)

In [7]:
```
print(shape(pattern))
eig_vals, eig_vectors = np.linalg.eig(pattern)
idx = (abs(eig_vals).argsort())
idx = idx[::-1]
eig_vals = eig_vals[idx]
eig_vectors = eig_vectors[:,idx]
```

```
(400, 400)
```

Part c

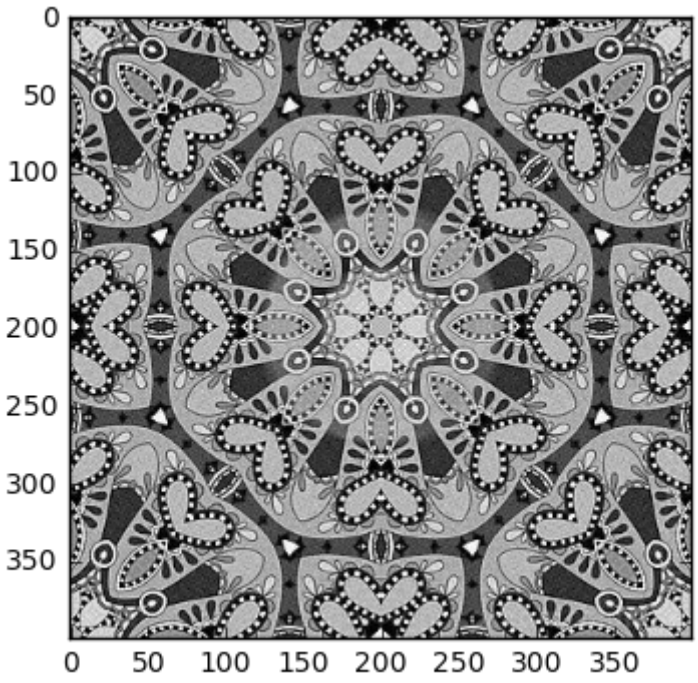
Find the pattern approximation using 100 largest eigenvalues/eigenvectors.

- Index into above variables to choose the first 100 eigenvalues and eigenvectors.
- You can use the command np.outer (http://docs.scipy.org/doc/numpy/reference/generated/numpy.outer.html) to find the outer product of two vectors

In [20]:
```
rank = 100
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += (val_i*np.outer(vec_i,vec_i.T))

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[20]: <matplotlib.image.AxesImage at 0x118b277b8>



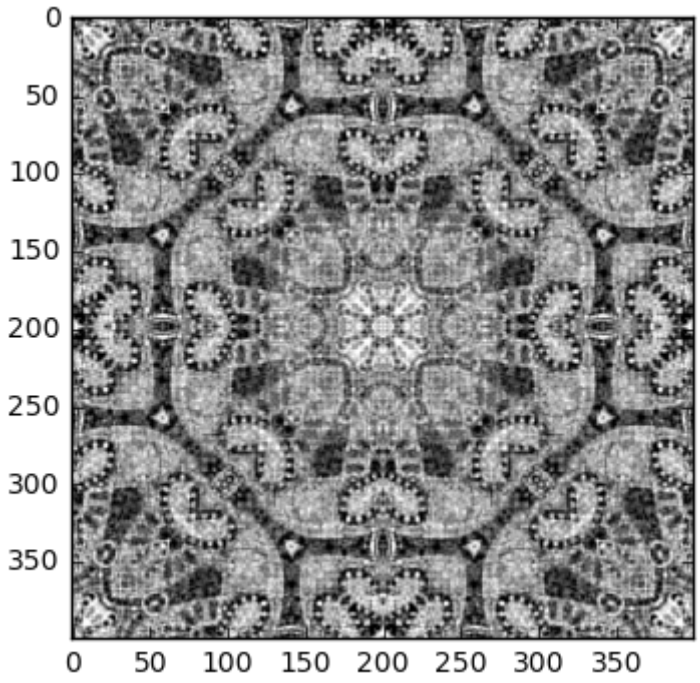
Part d

Find the pattern approximation using 50 largest eigenvalues/eigenvectors

In [35]:
```
rank = 20
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += (val_i*np.outer(vec_i,vec_i.T))

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[35]: <matplotlib.image.AxesImage at 0x11c7f9278>



Paths of a Surfer

In [38]:
```
# There is no required ipython component, but you may wish to use iPytho
n for calculations.
B = np.array([[0, 1, 1/3, 1/3], [0, 0, 1/3, 1/3], [0, 0, 0, 1/3], [1, 0
,1/3, 0]])
np.linalg.eig(B)
```

Out[38]:
```
(array([ 1.00000000+0.j        , -0.33333333+0.47140452j,
        -0.33333333-0.47140452j, -0.33333333+0.j        ]),
 array([[ -6.13571991e-01+0.j        ,  -1.57134840e-01+0.44444444j,
          -1.57134840e-01-0.44444444j,   3.55817481e-16+0.j        ],
        [ -3.06785996e-01+0.j        ,  -3.14269681e-01-0.11111111j,
          -3.14269681e-01+0.11111111j,   4.19634415e-16+0.j        ],
        [ -2.30089497e-01+0.j        ,  -2.35702260e-01-0.33333333j,
          -2.35702260e-01+0.33333333j,   7.07106781e-01+0.j        ],
        [ -6.90268490e-01+0.j        ,   7.07106781e-01+0.j        ,
           7.07106781e-01-0.j        ,  -7.07106781e-01+0.j        ]]))
```

In [50]:
```
A = np.array([-0.614, -0.307, -0.230, -0.690])
scalar = 1 / abs(np.sum(A))
A = numpy.around((scalar * A), 3)
A
```

Out[50]:
```
array([-0.334, -0.167, -0.125, -0.375])
```

In [51]:
```
Ctrans = np.array([[0,1,0,0,0], [1,0,0,0,0],[0,0,0,0,1],[0,0,1/2,0,0],
[0,0,1/2,1,0]])
np.linalg.eig(Ctrans)
```

Out[51]:
```
(array([ 1.0+0.j , -1.0+0.j ,  1.0+0.j , -0.5+0.5j, -0.5-0.5j]),
 array([[ 0.70710678+0.j        , -0.70710678+0.j        ,
          0.00000000+0.j        ,  0.00000000-0.j        ,  0.00000000+
0.j         ],
        [ 0.70710678+0.j        ,  0.70710678+0.j        ,
          0.00000000+0.j        ,  0.00000000-0.j        ,  0.00000000+
0.j         ],
        [ 0.00000000+0.j        , -0.00000000+0.j        ,
         -0.66666667+0.j        ,  0.70710678+0.j        ,  0.70710678-
0.j         ],
        [ 0.00000000+0.j        , -0.00000000+0.j        ,
         -0.33333333+0.j        , -0.35355339-0.35355339j,
         -0.35355339+0.35355339j],
        [ 0.00000000+0.j        , -0.00000000+0.j        ,
         -0.66666667+0.j        , -0.35355339+0.35355339j,
         -0.35355339-0.35355339j]]))
```

In [ ]: