CS216 HW1 Name: Yifan Tian ID:78921267

A: Matlab Warmup

1.

a. x = randperm(5);
X is a vector whose value is a permutation of 1-5, so 5 elements.

b.
```
>> a = [1 2 3;4 5 6;7 8 9];
>> b = a(2,:);
```
a is a 3x3 matrix, b is the second row of a.

c.
```
>> f = [1501:2000];
>> g = find(f > 1850);
>> h = f(g);
```
g is the index set of elements which is greater than 1850.
So h is the array of elements which is greater than 1850.

d.
```
>> x = 22.*ones(1,10);
>> y = sum(x);
```

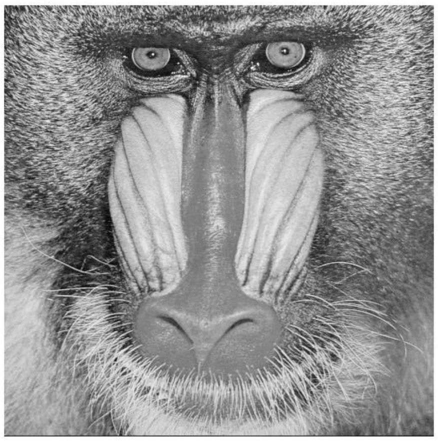
x is a diagonal matrix whose diagonal elements is 22, y is going to sum the elements of y.

e.
```
>> a = [1:100];
>> b = a([end:-1:1]);
```

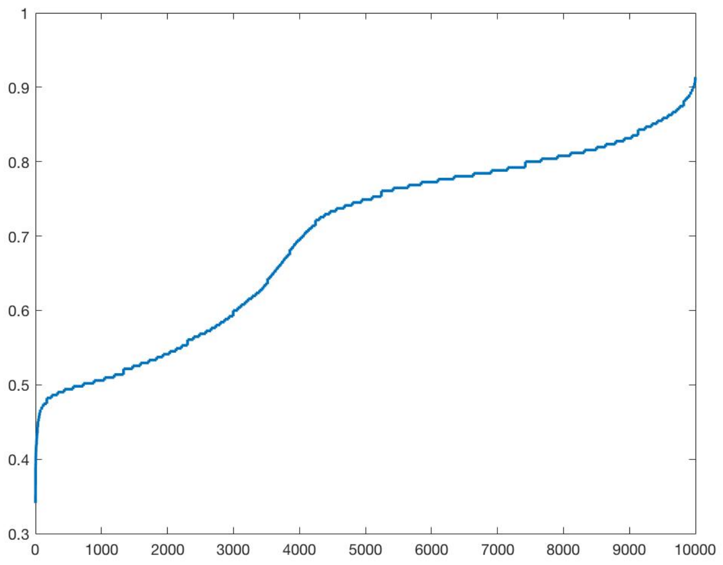
a is a single 100-dimensional vector from 1 to 100, b reverses a.

2.

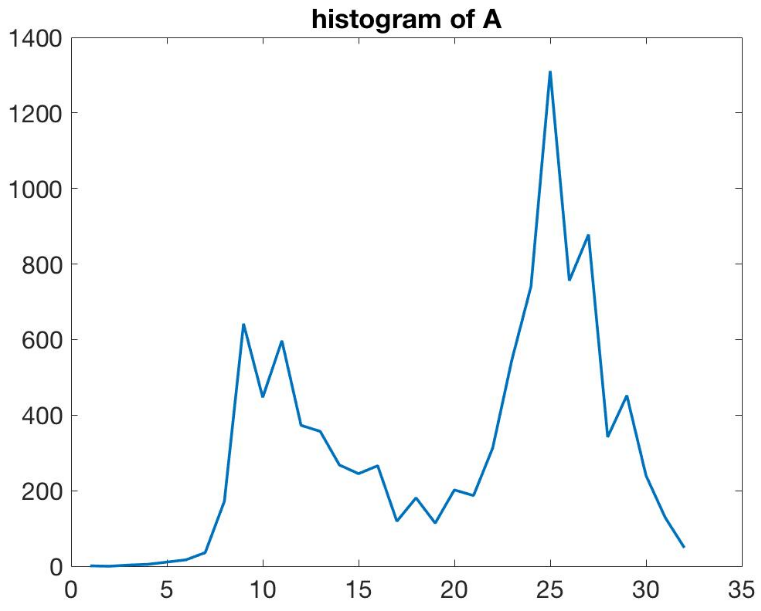
Image for question 2:



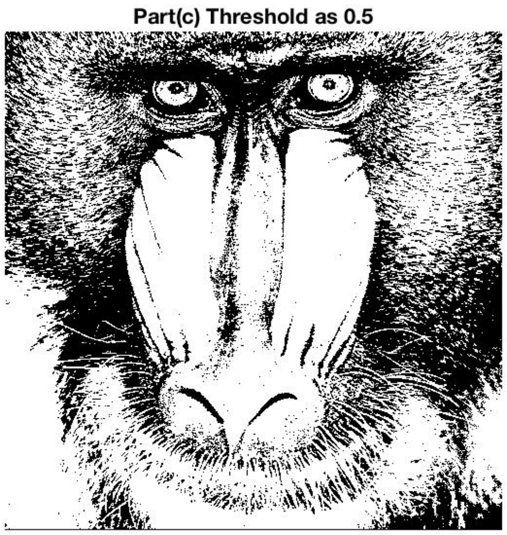
a. Sort all the intensities in A, put the result in a single 10,000-dimensional vector x, and plot the values in x.



b. Display a figure showing a histogram of A's intensities with 32 bins using the *hist* function.

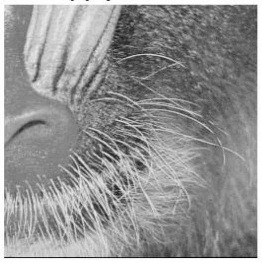


c. Create and display a new binary image the same size as *A*, which is white wherever the intensity in A is greater than a threshold *t*, and black everywhere else.



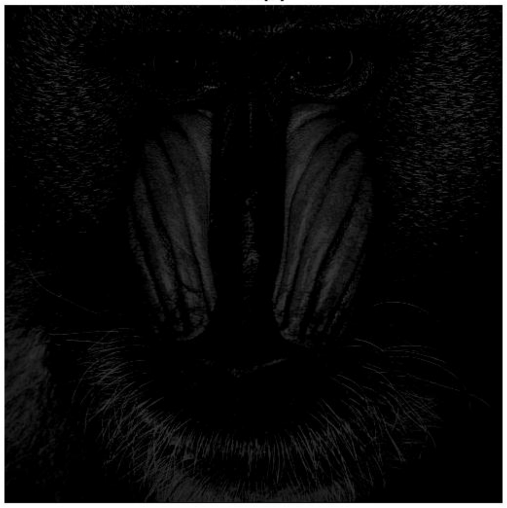
d. Display the bottom right quadrant of *A*.

Part(c) quadrant of A

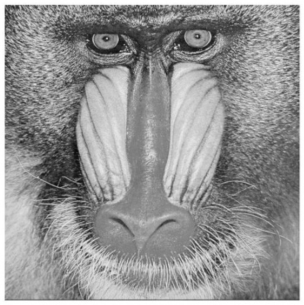


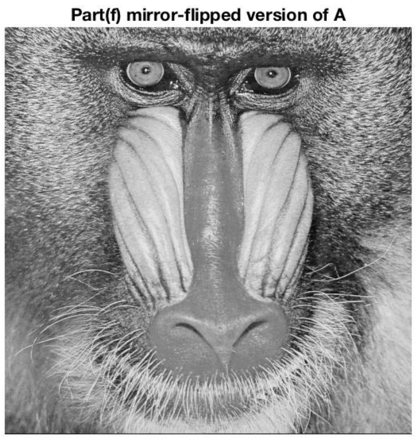
e. Generate a new image (matrix), which is the same as *A*, but with *A*'s mean intensity value subtracted from each pixel. Set any negative values to 0.

Part(d)



f. Display the mirror-flipped version of image A. That is, if A was an image of right-facing car, transform A to be a left-facing car.





g. Use the *min* and *find* functions to set *x* to the single minimum value that occurs in *A*, and set *r* to the row it occurs in and *c* to the column it occurs in.

```
[row,col] = find(A == min(min(A)));
```

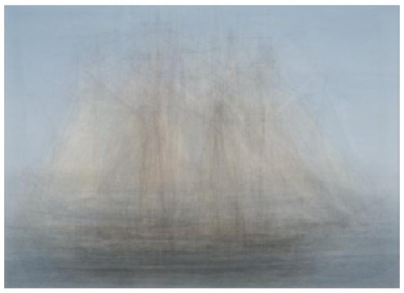
h. Let *v* be the vector: v = [1 8 8 2 1 3 9 8]. Use the *unique* function to compute the total number of unique values that occur in *v*.

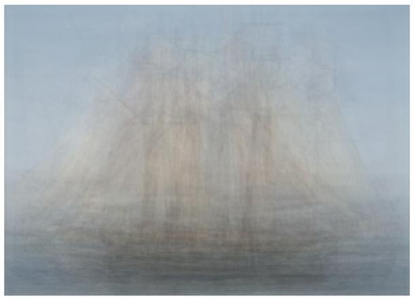
```
v = [1 8 8 2 1 3 9 8];
uni = length(unique(v));

uni = 5
```

B: Computing average images

The average images for image set 1.



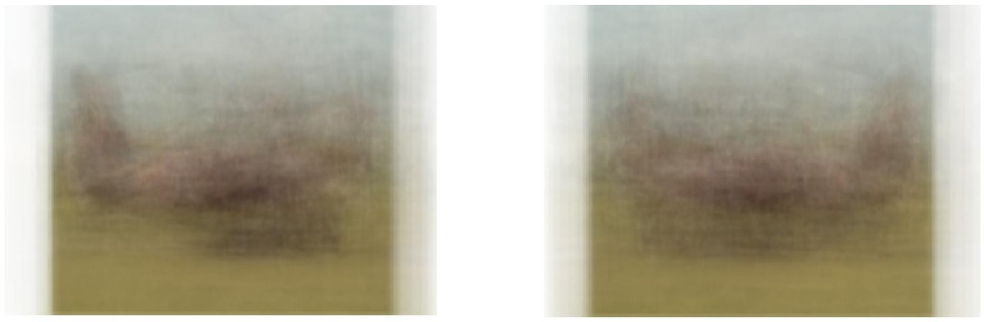


Left: without randomization, right: after randomization.

The average image still has an outline of a ship. Because the horizontal is mixed, but the feature along y direction preserved.

Difference between left and right is small, because the randomization of left and right has no effect on average figure. In the original image set, every ship is oriented to left and right evenly, so the randomization of left and right will not change the average image too much.

The average images for image set 2.



Left: without randomization, right: after randomization.

The left average image still has an outline of an airplane, and the plane seems is **rising from left to righ**t. But the average image in left **is unclear in the orientation**. I think it is because in the image set, more plane is oriented to right. The average figure still contains this information. But after horizontal randomization, the feature of horizontal orientation is lost. (every plane has even chance pointing to right or left).

C: Image classification

10 classes:

Airplane(1) automobile(2) bird(3) cat(4) deer(5) dog(6) frog(7) horse(8) ship(9) truck(10).

For confusion matrix (i,j), i for real class, j for prediction.

Following are results from different classification strategies:

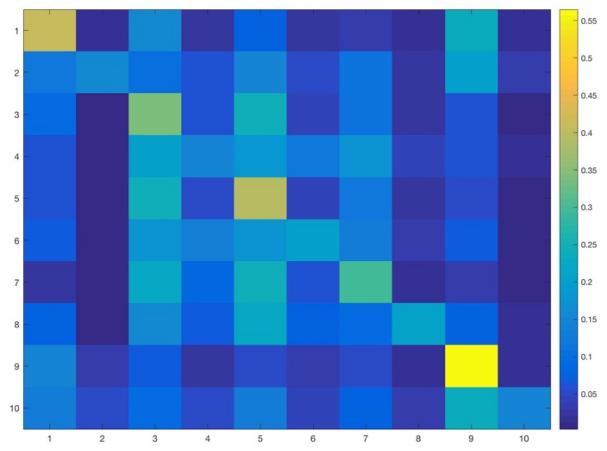
1. **Display the first plane:**

2. **Constructing the nearest-neighbor classifier**
   (1) Using NN (k nearest-neighbor) method. Here the distance metric is Euclidian distance:

   The matlab code for computing distance between the test data vector and batch vector:

```
distance = pdist2(double(data1(j,:)),double(testdata(k,:)));
```

3. Compute a class confusion matrix

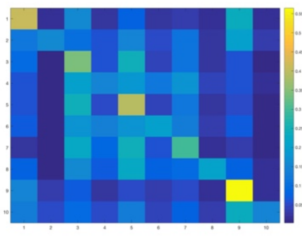


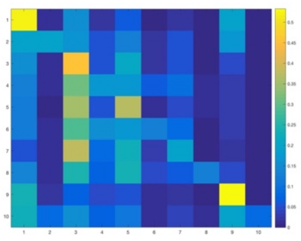
The matrix in detail:

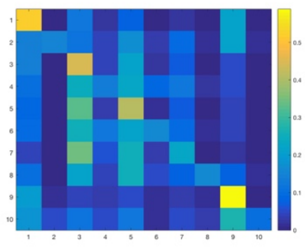
0.413	0.012	0.154	0.027	0.075	0.021	0.032	0.014	0.239	0.013
0.121	0.157	0.103	0.059	0.145	0.049	0.11	0.021	0.204	0.031
0.099	0.005	0.345	0.064	0.245	0.045	0.109	0.021	0.062	0.005
0.061	0.008	0.197	0.148	0.182	0.119	0.173	0.039	0.058	0.015
0.062	0.004	0.248	0.055	0.396	0.04	0.119	0.022	0.051	0.003
0.065	0.006	0.17	0.139	0.173	0.203	0.134	0.033	0.067	0.01
0.026	0.004	0.225	0.089	0.241	0.063	0.298	0.012	0.037	0.005
0.077	0.005	0.157	0.068	0.228	0.076	0.091	0.207	0.079	0.012
0.149	0.032	0.065	0.029	0.049	0.03	0.048	0.015	0.565	0.018
0.127	0.052	0.097	0.053	0.131	0.046	0.079	0.038	0.232	0.145

The diagonal elements:
(1, 1) 0.413000
(2, 2) 0.157000
(3, 3) 0.345000
(4, 4) 0.148000
(5, 5) 0.396000
(6, 6) 0.203000
(7, 7) 0.298000
(8, 8) 0.207000
(9, 9) 0.565000
(10, 10) 0.145000
The average classification rate is 0.2877.

4. Construct a K NN-classifier that computes the K closest training images, and returns the most common label from this set.

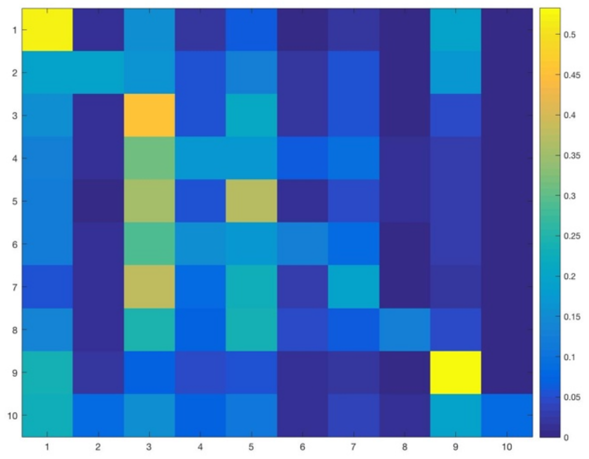

K = 1


K = 3


K= 5

The above are results of class-confusion matrix of KNN-classifier for different K. **So K = 3 is better than other two Ks**, as it has better classification rate on class 1 and class 3.

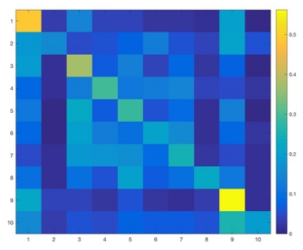
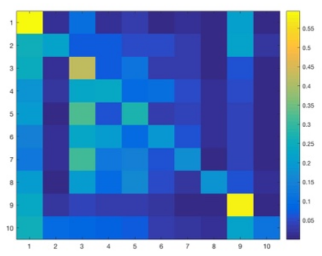
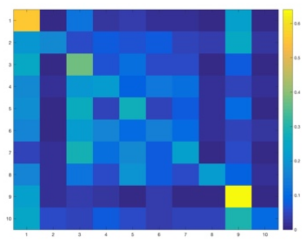
Class-confusion matrix for K = 3



The average classification rate (the diagonal element):
(1, 1) 0.518000
(2, 2) 0.198000
(3, 3) 0.453000
(4, 4) 0.170000
(5, 5) 0.370000
(6, 6) 0.129000
(7, 7) 0.199000
(8, 8) 0.130000
(9, 9) 0.533000
(10, 10) 0.085000
The average classification rate is 0.2785.

5. Construct a NN-classifier that uses normalized correlation (or cosine similarity) instead of Euclidian distance. Report the best missclassification rate (find the K that has the best performance) and display its class-confusion matrix.

   Class-confusion matrices:

K = 1 k = 3 k = 5

From above three Class-confusion matrices, **the k = 3 seems have better performance as it has better classification in most classes.** We can see a clear ridge and also a much better classification rate in class1 (airplane).

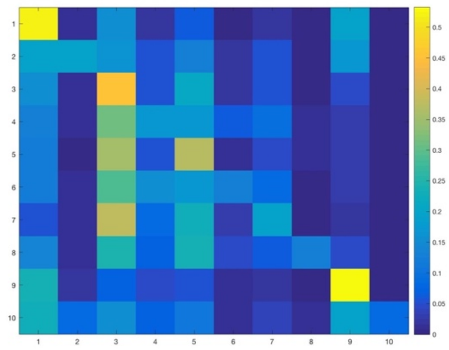
The classification rate for k = 3:

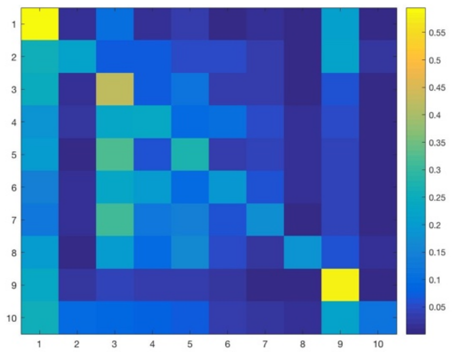
(1, 1) 0.595000
(2, 2) 0.217000
(3, 3) 0.422000
(4, 4) 0.238000
(5, 5) 0.279000
(6, 6) 0.194000
(7, 7) 0.170000
(8, 8) 0.182000
(9, 9) 0.581000
(10, 10) 0.117000
The average classification rate is 0.2995**.**

6. What is your best-performing system (the value of K, and the distance metric)? Provide some explanations as to why you think this combination performed best.

   First let's pick up the best system among two distance metric. So we have k = 3 for both Euclidian distance and normalized correlation.

   The we compare their class-confusion matrices:





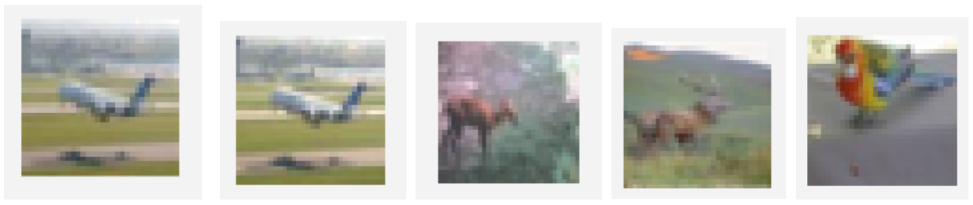
(Euclidian distance, k =3) (Normalized correlation k =3)

Euclidian distance is better in some classes like 3,5 while Normalized correlation is slightly better in more other classes like 1,2,4..
So let's compare their total classification rate:
Total:

Euclidian distance: 0.2785
Normalized correlation: 0.2995
So Normalized correlation is better in average performance.

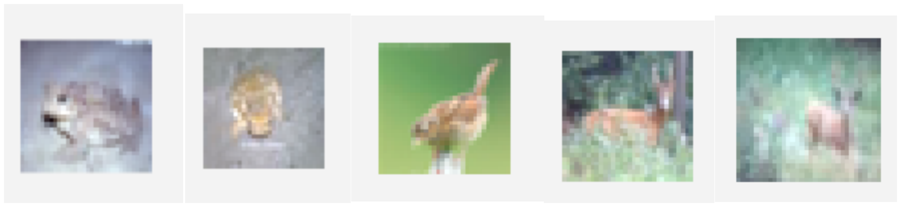
The classes that tend to confuse with others are 6(**dog**),7(**frog**), 8(**horse**), 10(**truck**).

Example for truck:



The system predict it as airplanes, or deers or birds.
While no one looks like the truck above. I guess each figure has some part that is close to the truck above. Like the distribution of color(The bird has a blue tail) and the distribution of dark and light. (The first two, the shadow of airplanes)

Example for Frog:



The test image is a frog, while the system predict as frogs (2), bird(1), and deer(2). It is easy to understand why it is to confuse. As we can see the last three figures has the same distribution of colors, **the bird and deer are brown,**

while the background is green. This is consistent with the result from confusion matrix as frog is easy to be predicted as deer and bird. The entry (7,3) and (7,5) is large.

Observation: We can see that (4,3) (5,3) (6,3) (7,3) entry has high value, means it is easy for class 4,5,6,7(cat, deer, dog, frog) to be predicted as class 3(bird). Also frog and bird, deer and bird are easy to confuse. 2,4,6,8,10 is hard to classify. Airplane and ship are easy to classify.

Reasoning: We can also find that it is easy to be predicted as airplane(1), bird(3), ship(9), but hard to be predicted as automobile(2), horse(8), truck(10), I think it is because the distribution of color and shadow. Birds have many kinds of colors, while plane has many kinds of shadows.

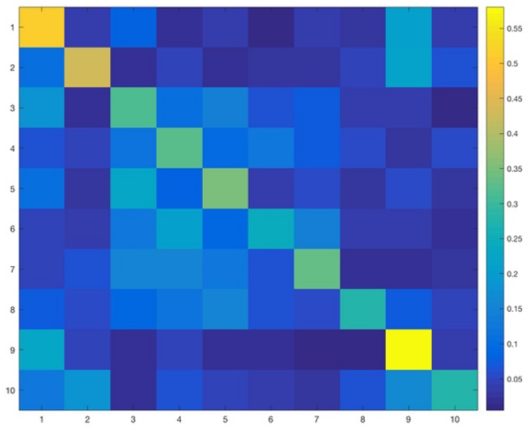
Other tries or modifications:

1. Use correlation distance:

$$d_{st} = 1 - \frac{(x_s - \bar{x}_s)(y_t - \bar{y}_t)'}{\sqrt{(x_s - \bar{x}_s)(x_s - \bar{x}_s)'}\sqrt{(y_t - \bar{y}_t)(y_t - \bar{y}_t)'}}$$

As in matlab it is defined as pdist(vec1,vec2,'correlation');

Class-confusion matrix for K = 3:



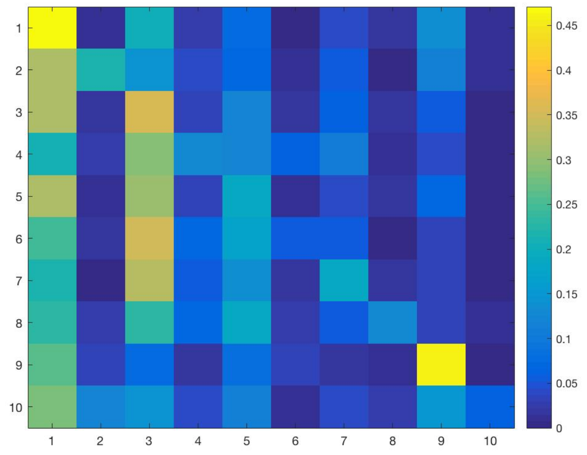
The average classification rate (the diagonal element):
(1, 1) 0.561224

(2, 2) 0.429293
(3, 3) 0.358974
(4, 4) 0.366834
(5, 5) 0.257576
(6, 6) 0.210811
(7, 7) 0.310185
(8, 8) 0.264249
(9, 9) 0.497696
(10, 10) 0.231527
Average classification rate = 0.3488

It is clear as the correlation distance has better performance in almost all classes

2. **Using gray image instead color image to classify:**
Color image: **Gray image:**

K = 3, using Euclid distance

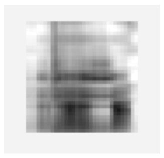


Here, using gray image is not as good as using color image, it is reasonable as color contains some information.
(1, 1) 0.479592
(2, 2) 0.196970
(3, 3) 0.415385
(4, 4) 0.195980
(5, 5) 0.171717
(6, 6) 0.059459

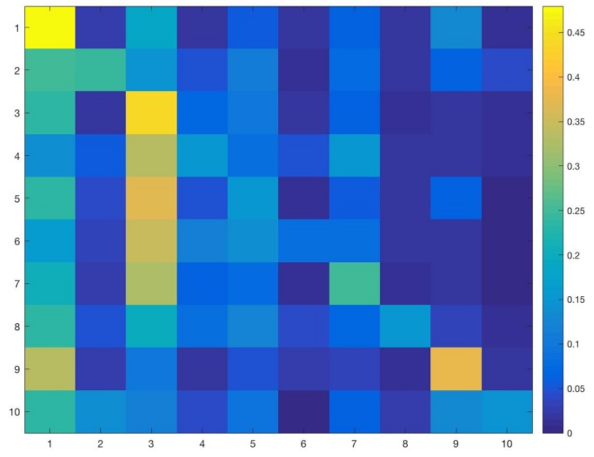
(7, 7) 0.157407
(8, 8) 0.150259
(9, 9) 0.437788
(10, 10) 0.073892
The average error for this classification is 0.2338.

Due to the reduction of color dimension. Figures become similar. So most classes have some similarity with airplane(1) and class 4,5,6,7 has similarities with bird(3).

3. **Using SVD to compress data:**
Original color: original gray: Image after SVD as dimension of V is 3:



The error of energy is 0.0195



A classification after I do SVD for both data set and target.

(1, 1) 0.479592
(2, 2) 0.242424
(3, 3) 0.441026
(4, 4) 0.150754
(5, 5) 0.156566
(6, 6) 0.086486
(7, 7) 0.250000

(8, 8) 0.155440
(9, 9) 0.377880
(10, 10) 0.142857

The average error for this classification is 0.2483. Slightly better than use original gray image. I guess it is because SVD remove the noise.