MATH3290 Mathematical Modeling 2018/2019
Assignment 2
Suggested Solutions

1. On the popular TV show The Price Is Right, at the end of each half hour, the three winning contestants face off in a spin game. The game consists of spinning a large wheel with 20 spaces on which the pointer can land, scored from \$0.05 to \$1.00 in 5−cents increments. The objective of the game is to obtain as close to \$1.00 as possible without going over that amount with a maximum of two spins. If a player's total score goes beyond 1.00, it would be reseted as 0.

   Every player can decide whether or not he want a second spin. The spinning sequence of these three players are random. Naturally, if the first player does not go over, the other two will use one or two spins in an attempt to overtake his score.

   If you are chosen as the first player, what's your strategy to maximize your expected total score? That is, what's your expected value on your first spin so that a second spin is not needed?

   Simulate the game by computer coding and use your result to support your strategy.

   Solution 1. *When we choose to reject a second spin if the first spin is greater or equal to C, the simulation result is shown in Table 1. As we can see, when the simulation times is large enough, say $K = 20000000$, the total score attains it's maximum* 0.630017 *when $C = 0.45$. Therefore, the player is suggested to stop if his or her first spin is greater or equal to* 0.45.

Table 1: Simulation result

C	$K = 200000$	$K = 2000000$	$K = 20000000$
0.05	0.525660	0.524839	0.524997
0.10	0.548573	0.548580	0.548733
0.15	0.569256	0.569354	0.569481
0.20	0.587350	0.587156	0.587497
0.25	0.602286	0.602465	0.602569
0.30	0.613515	0.614138	0.614440
0.35	0.623019	0.623034	0.622898
0.40	0.626795	0.628398	0.628275
0.45	0.628967	0.629782	0.630017
0.50	0.628554	0.628404	0.628107
0.55	0.623708	0.622371	0.622493
0.60	0.613005	0.613496	0.612948
0.65	0.599309	0.599814	0.599481
0.70	0.580791	0.582042	0.581745
0.75	0.561107	0.560081	0.559903
0.80	0.535121	0.533852	0.533818
0.85	0.502291	0.503300	0.503133
0.90	0.468385	0.467404	0.467568
0.95	0.428410	0.427608	0.427489
1.00	0.381769	0.382466	0.382459

2. In this exercise, you will use k-means to compress an image by reducing the number of colors it contains. In a straightforward 24-bit color representation of image, each pixel is represented as three 8-bit integers (ranging from 0 to 255) that specify red, green and blue intensity values (`https://en.wikipedia.org/wiki/RGB_color_model`). One photo may contain thousands of colors, but we would like to reduce that number to a small number k. By making this reduction, it would be possible to represent the photo in a more efficient way by storing only the RGB values of the k colors present in the image. We may achieve this by the following steps.

Step 1 Download **William.zip** and unpack its contents into your Matlab working directory. There are two photos in this package. The one named **William-big.png** contains 983×616 pixels, while they are only 150×93 pixels in the image named **"William-small.png"**. We will run k-means on the 150×93 image **"William-small.png"**.

Step 2 In Matlab, load the small image into your program with the following command:

```
A = double(imread('William-small.png'));
```

This creates a three-dimensional matrix A whose first two indices identify the pixel position and whose last index represents red, green, or blue. For example, $A(50, 33, 3)$ gives you the blue intensity of the pixel at position $y = 50, x = 33$.

Step 3 To initialize, pick k colors randomly from the small image. These are the k-means in the beginning.

Step 4 Compute k cluster centroids from this image, with each centroid being a vector of length three that holds a set of RGB values.

Step 5 After k-means has converged, load the large image into your program and replace each of its pixels with the nearest of the centroid color you found from the small image. You can set the maximum of the number of iterations to be 100.

For $k = 4, 8, 16$, repeat the above steps and draw the picture respectively. Compare your picture with **William-big.png**.

Solution 2. *Plese refer to the **q2.m**. For* $k = 4, 8, 16$*, the pictures we get can be shown as following:*

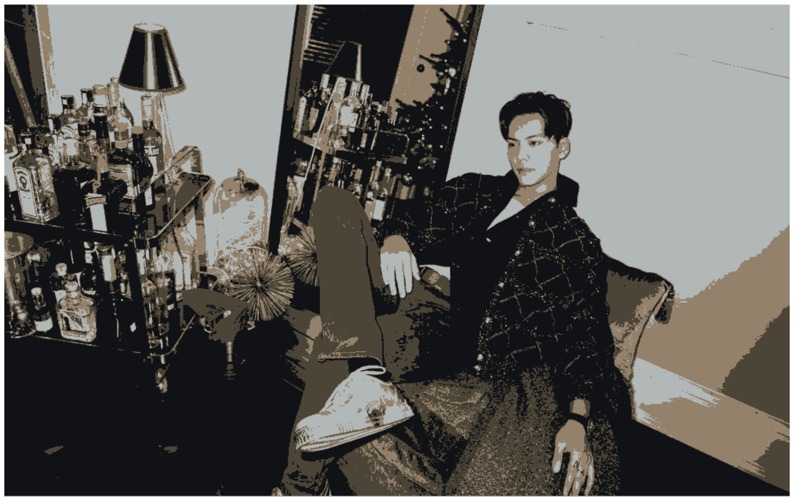


Figure 1: Compressed pictures, $k = 4$.

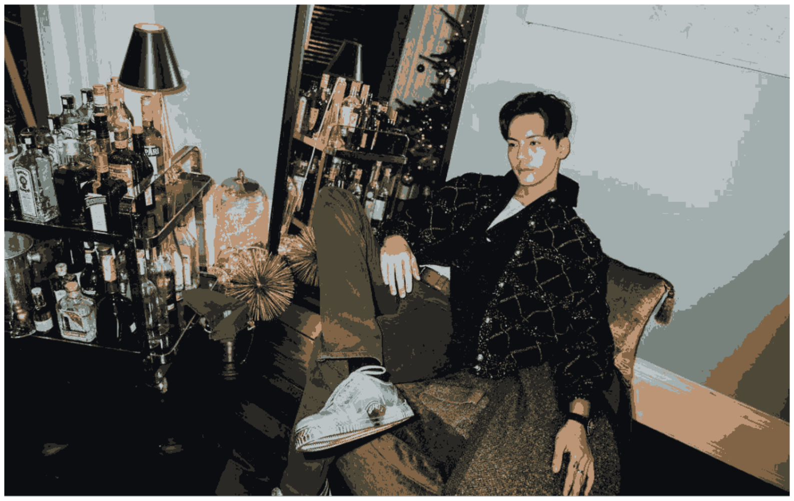


Figure 2: Compressed pictures, $k = 8$.

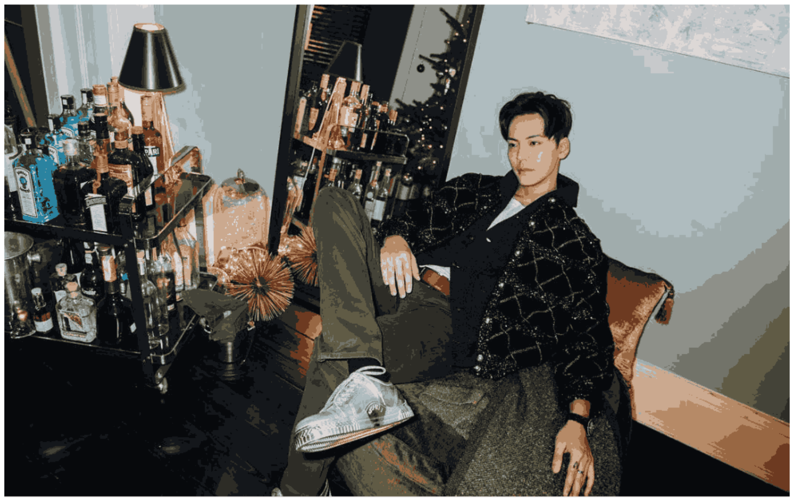


Figure 3: Compressed pictures, $k = 16$.

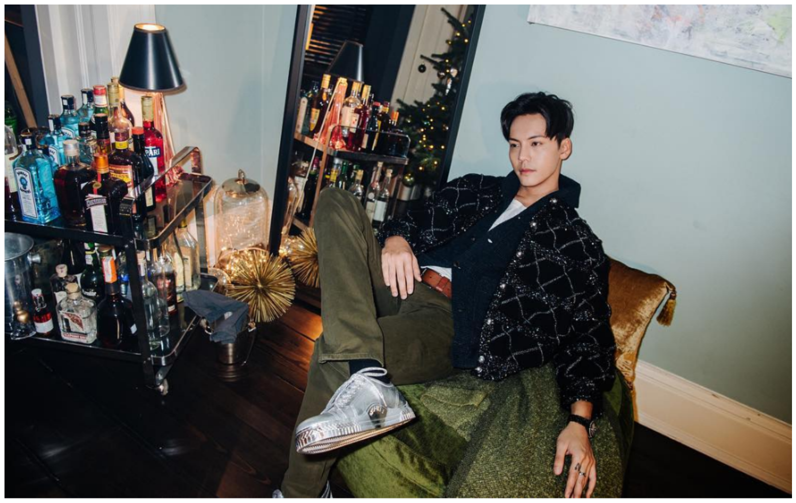


Figure 4: Original sample picture.

Hence, it is clear that the picture get more and more vivid as the number of colors increases.

3. In **yalefaces.zip**, there are 10 images of dimensions 61×80 and a Matlab file **Q2.m**. In the Matlab file, we rescale each pixel value to the interval [0,1] and reshape the image to a $(61 \times 80) \times 1$ vector. First 9 images are stored in the matrix X. Do the following in **Q2.m**:

   (a) Following the lecture notes, perform *principle component analysis* on X.

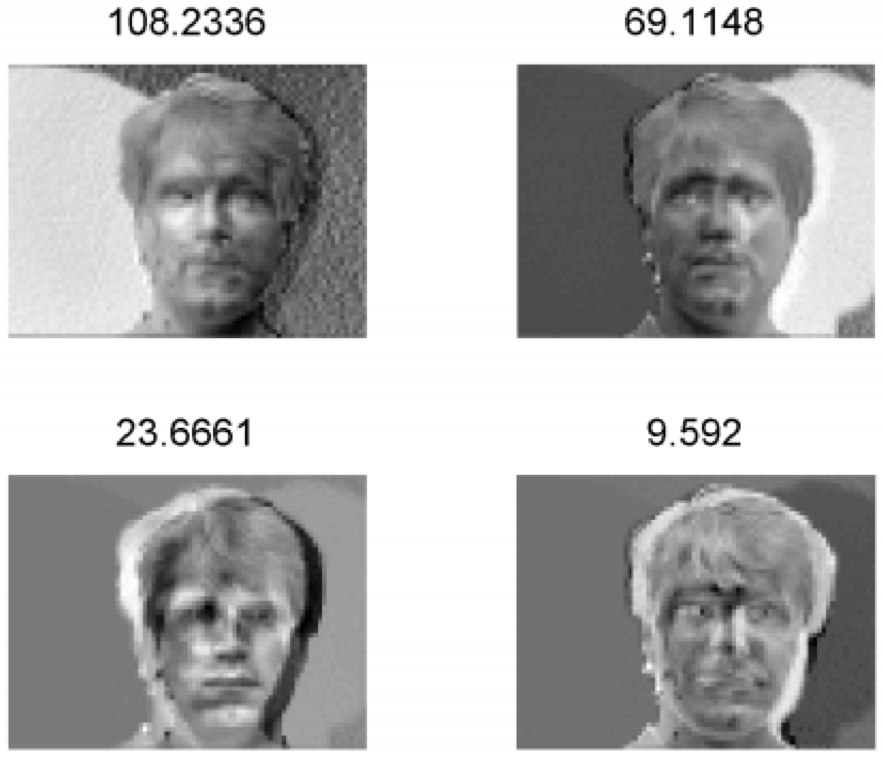
   (b) Find the four largest eigenvalues and show the corresponding eigenvectors (reshape it into the dimensions of the original image).

   (c) Compute the relative error for data compression using only the eigenvectors in part (b).

   (d) Use the eigenvectors in part (b), express the 10^{th} image (**10.gif**). Explain why the representation has bad quality.

Solution 3. *Please refer to* ***q3.m****.*

(b) The largest four eigenvalues and corresponding eigenvectors are shown below.



(c) Let $\lambda_1 \geq \lambda_2 \geq \dots \lambda_9$ nr the eigenvalues. The relative error for the data compression is

$$Error = \sqrt{\frac{\lambda_5}{\lambda_1 + \cdots + \lambda_9}} = 15\%.$$

(d) The 10^{th} image and the synthetic image are shown below. The 10^{th} images are shifted towards the right and tilted comparing to the first 9 pictures. Therefore the location of the features (like eyes, mouth etc) does not match the previous 9 images.