MATH3290 Mathematical Modeling

Tutorial 6

25th October 2017

Outline

1. **Simulation**
   - Generate Discrete Random Variable
   - Example: Blackjack
2. **Image processing using k-means**
   - k-means
   - k-means Example
3. **Data Compression using PCA**
   - Quick Review of PCA
   - Hints for Assignment 2

Generate Discrete Random Variable

Assume that Z is a random variable (r.v.) uniformly distributed in $[0,1]$, one can generate other discrete r.v. ω_i using Z, where the sample space of ω_i is $S = \{\omega_1, \cdots, \omega_N\}$. Suppose that the probability is $p_i := P(X = x_i)$ satisfying $\sum_{i=1}^{N} p_i = 1$. Next, define the following function g

$$g(z) = \begin{cases} \omega_1, & 0 < z \leq p_1, \\ \omega_2, & p_1 < z \leq p_1 + p_2, \\ \vdots & \vdots \\ \omega_N, & p_1 + \cdots + p_{N-1} < z \leq 1. \end{cases}$$

Then $g(Z)$ is our desired random variable.

Generate Discrete Random Variable (Cont.)

Some examples

1. To simulate the outcome from rolling a fair die, one can take $g(z) = \lceil 6 \times z \rceil$ and the function g would be

$$g(z) = \begin{cases} 1, & 0 < z \leq 1/6, \\ 2, & 1/6 < z \leq 1/3, \\ \vdots & \vdots \\ 6, & 5/6 < z \leq 1. \end{cases}$$

   and $N = 6, p_i = 1/6$.
2. Similarly, one can simulate the outcome from picking a deck of poker. ($g(z) = \lceil 13 \times z \rceil$, $N = 13, p_i = 1/13$)

Example: Blackjack

To simulate a game, do the following steps

1. Understand the rules of the game.
2. Simulate the probabilistic behavior.
3. Implement the game using the flows of control (`if-else`, `while`, `for`).
4. Repeat the game a number of times, and approximate the probability desired.

Quick Review of k-means

Goal: compute the centers $c_1, \cdots, c_k$.

1. Initial step: choose $c_1, \cdots, c_k$ and set $\mathtt{SSE} = \infty$.
2. Assign the points to the closest clusters c_i only if the distance strictly decreases.
3. For each non-empty cluster, re-compute the center c_i.
4. Compute the $\mathtt{SSE}$.
5. If the $\mathtt{SSE}$ decrease, continue. (If not, STOP)

Problem Descriptions

Given a colorful image with different size. Your task is to sample the elements of color from the small image and replace each of the pixel from the big image with the nearest of the centroid color.

1. Load the small image.
2. Pick k colors randomly from the small image initially.
3. Perform k-means algorithm until it converges. (max number of iteration: 100)
4. Load the big image and replace each of its pixels with the nearest of the centroid color found from the small image.

Quick Review of PCA

Given a set of data points $x_1, \cdots, x_n \in \mathbb{R}^d$. Do the following steps to perform PCA.

1. Define the following $d \times d$ matrix:

$$Q = \frac{1}{n}\sum_{j=1}^{n} \tilde{x}_j \tilde{x}_j^T, \quad \tilde{x}_j = x_j - m, \quad m = \frac{1}{n}\sum_{j=1}^{n} x_j.$$

2. Compute the eigenvalues λ_k of Q ($k = 1, \cdots, d$). Assume that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

and find the corresponding eigenvectors are $u_1, u_2, \cdots, u_d$, respectively.

Quick Review of PCA (Cont.)

3. Choose k principal directions $u_1, \cdots, u_k$, $1 \le k \le d$.
4. Compute the projection of $\tilde{x}_j$ to these eigenvectors, where $j = 1, \cdots, n$, that is

$$c_{js} := \tilde{x}_j^T u_s, \quad s = 1, \cdots, k.$$

   Stroage is $n \times k + d \times k + d$.
5. Recover the data using

$$\hat{x}_j = \sum_{s=1}^{k} c_{js} u_s + m, \quad j = 1, \cdots, n$$

6. Compute the relative error e_i as follow:

$$e_i = \frac{\|x_j - \hat{x}_j\|}{\|x_j\|}, \quad j = 1, \cdots, n.$$

Hints for Assignment 2

Consider the 10 images of dimension 61×80. We want to complete the MATLAB file **Q2.m**. First 9 images are stored in the matrix X.

(a) Perform **PCA** 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 **(a)**.

(a) Read the raw data and form the matrix X. To obtain Q, we need to calculate the mean vector m and subtract it from X.

```
X = zeros(61*80,9);
for i = 1:9
    I = double(imread([num2str(i,'%02d') '.gif']))/256;
    X(:,i) = I(:);
end
av = mean(X')';
X_tilde = X-repmat(av,1,9);
% Use the built-in function eig
Q = (X_tilde) * (X_tilde)' /9;
[U,L] = eig(Q);
L = sort(diag(L));
```

Figure: Read data and forming Q.

(b) Since the build-in function **eig** has already sorted the eigenvalues, we take the last four largest entries from vector L. Also, we plot the corresponding eigenvectors as images. Those eigenvalues are:

$$\lambda_1 = 108.2336, \quad \lambda_2 = 69.1148,$$

$$\lambda_3 = 23.6661 \quad \text{and} \quad \lambda_4 = 9.5920.$$

```
f4 = L(end-3:end)
V  = U(:,end-3:end);
for i = 1:4
    subplot(2,2,i);
    imagesc(reshape(V(:,i),[61 80]));
end
```

Figure: Choose principal directions.

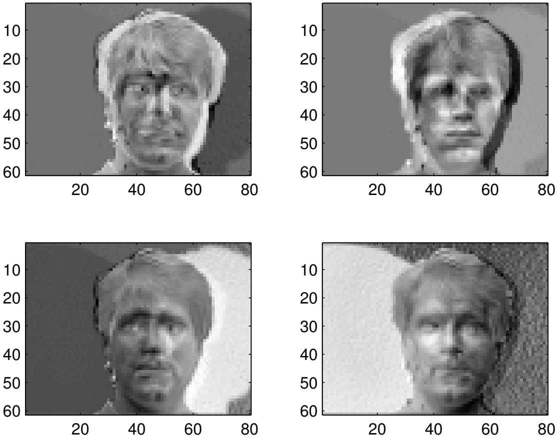

Figure: Corresponding eigenvectors.

(c) We calculate the projection of *X_tilde*, that is the matrix of coefficients *C*. Also, recover the data and denote it as *X_appro*. Then, we compuate the relative error.

```
C = (X_tilde)' * V;
X_appro = V*C' + repmat(av,1,9);
error = zeros(1,9);
for i = 1:9
    error(1,i) = norm(X(:,i)-X_appro(:,i))/norm(X(:,i));
end
```

Figure: Obtain projection and error.