Matlab 5: K-Means Clustering

Cheng-Hsin Hsu

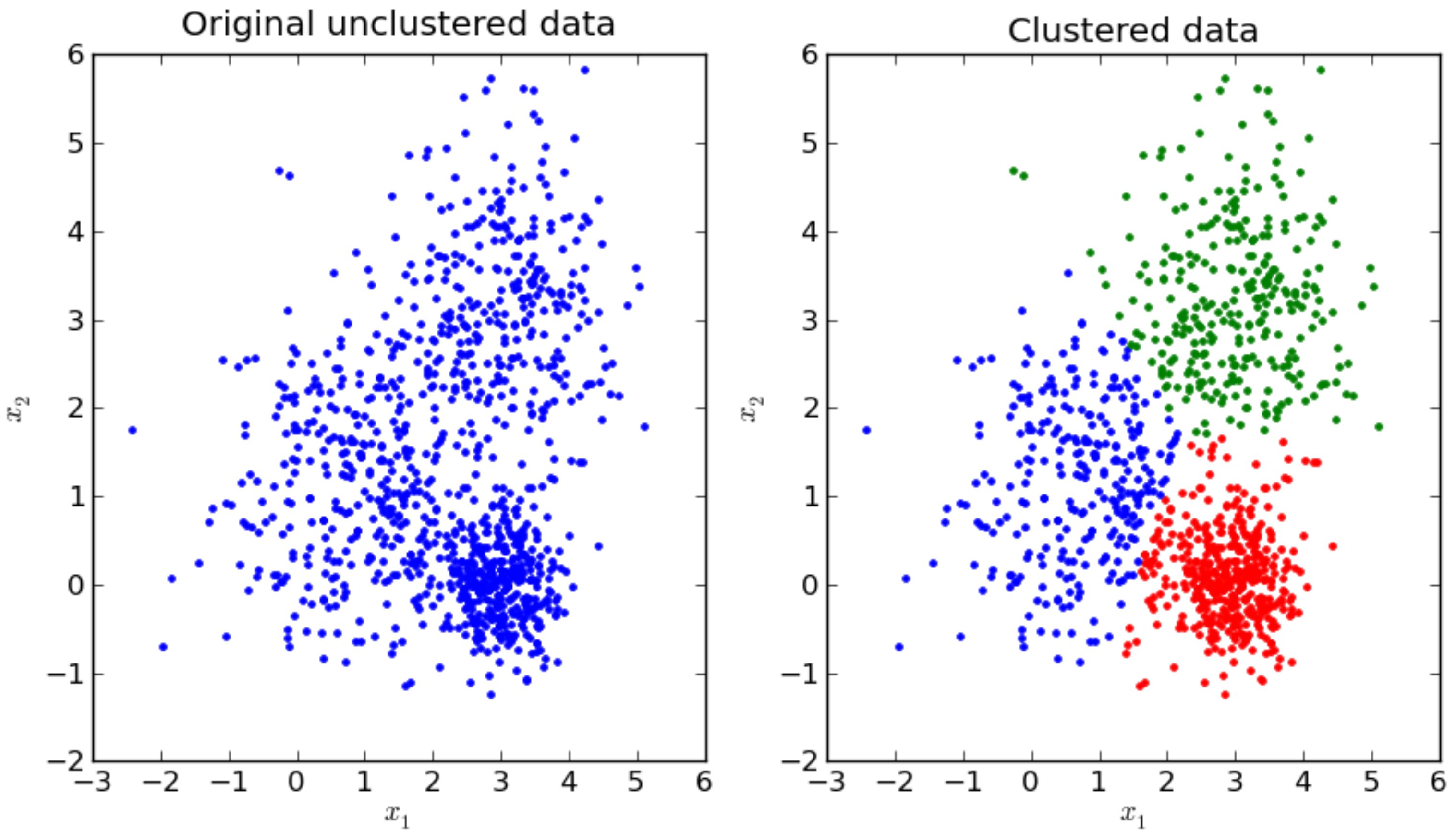
National Tsing Hua University

Department of Computer Science

Slides and sample codes are based on the materials from Prof. Roger Jang

Goals

- Let's visualize what is K-means clustering



K-Means Clustering

- Find *k* points of a dataset to best represent the dataset with minimum deviation (distortion)
  - *k* is a user-specified parameter, could be chosen using validation
- These chosen points are called cluster centers
  - Or prototypes, centroids, and codewords

Sample Applications

- Data classification: remove noisy data and reduce computational complexity
- Data compression: use the cluster centers to represent the original dataset ← fewer possibilities, easier to code
  - Homework: better indexed colors for the minion picture, chosen by your K-mean code

High-Level Idea

- Objective function: the sum of square distances between each data point and its nearest cluster centers ← called distortion
- Have to make two crucial decisions
  - Where are the cluster centers?
  - Which cluster does each data point belong to?
- Approach: we iteratively find the optimal of a decision while having the other decision fixed ← Coordinate optimization

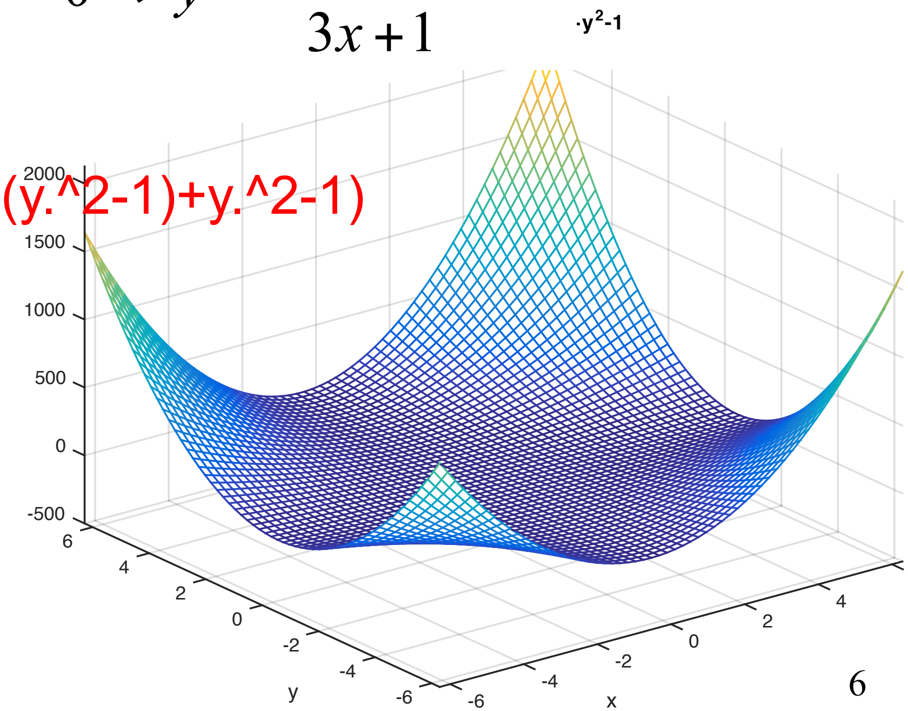
Example of Coordinate Optimization

$$f(x,y) = x^2\left(y^2 + y + 1\right) + x\left(y^2 - 1\right) + y^2 - 1$$

$$\frac{\partial f(x,y)}{\partial x} = 2x\left(y^2 + y + 1\right) + \left(y^2 - 1\right) = 0 \Rightarrow x = -\frac{y^2 - 1}{2\left(y^2 + y + 1\right)}$$

$$\frac{\partial f(x,y)}{\partial y} = 2x(2y + 1) + x(2y) + 2y = 0 \Rightarrow y = -\frac{x}{3x + 1}$$

ezmesh(@(x,y) x.^2*(y.^2+y+1)+x.*(y.^2-1)+y.^2-1)



Math Notations

Input:

- $X = \{x_1, x_2, \ldots, x_n\}$ A data set in d-dim. space
- *m*: Number of clusters (we avoid using *k* here to avoid confusion with other summation indices)

Output:

- m cluster centers: $c_j, 1 \le j \le m$
- Assignment of each x_i to one of the m clusters:

$$a_{ij} \in \{0,1\}, 1 \le i \le n, 1 \le j \le m$$

$$\sum_{j=1}^{m} a_{ij} = 1, \forall i$$

Math Notations (cont.)

$$e_j = \sum_{x_i \in G_j} \left\| x_i - c_j \right\|^2$$

Objective function, we aim to minimize it

$$J(X;C,A) = \sum_{j=1}^{m} e_j = \sum_{j=1}^{m} \sum_{x_i \in G_j} \left\| x_i - c_j \right\|^2 = \sum_{j=1}^{m} \sum_{i=1}^{n} a_{ij} \left\| x_i - c_j \right\|^2, where$$

$$X = \{x_1, x_2, \ldots, x_n\}$$

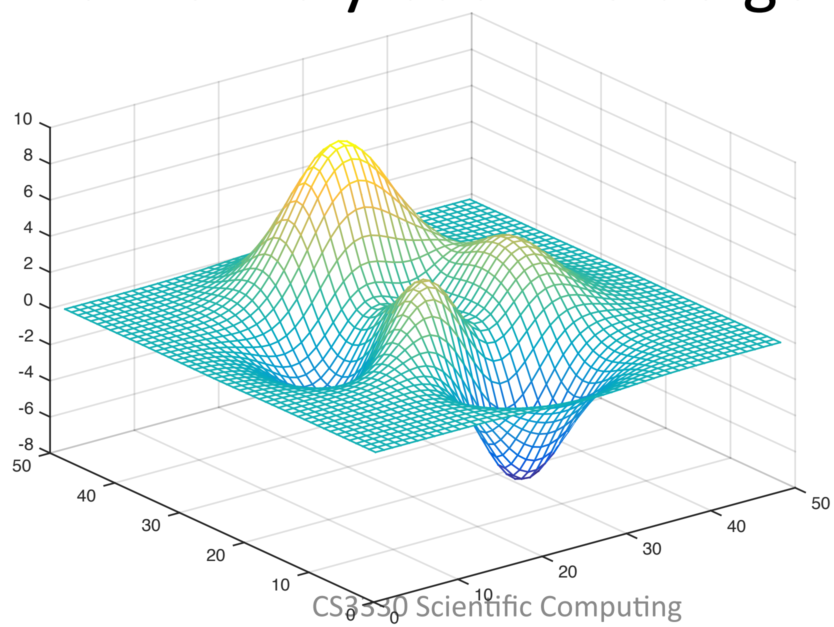
$$C = \{c_1, c_2, \ldots, c_m\}$$

Decision variables, notice that C has a dim of d x m and A has a dim of n x m

$$a_{ij} = 1 \; iff \; x_i \in G_j, \; with \sum_{j=1}^{m} a_{ij} = 1, \forall i$$

Minimizing J(X; C,A)

- Turns out to be NP-Hard
- Fall back to coordinate optimization
  - It's not perfect: we don't get global optimum
  - Yet it's not terribly bad: we do get local optimum



Step 1: Finding the Best A (Association)

- Analytic (closed-form) solution exists
- Intuition: $\frac{\partial J(X,C,A)}{\partial a_{ij}} = \|x_i - c_j\|^2 \; \forall a_{i,j}$
- Therefore: $\hat{a}_{ij} = \begin{cases} 1 \; if \;\; j = \arg\min_q \left\| x_i - c_q \right\|^2 \\ 0, \; otherwise \end{cases}$ Optimal for this step
- Or formally:

$$\hat{A} = \arg\min_A J(X;C,A) \Leftrightarrow J(X;C,A) \geq J(X;C,\hat{A}), \forall C$$

Step 2: Finding the Best C (Centers)

- Analytic (closed-form) solution also exists
- Intuition: $$\frac{\partial J(X,C,A)}{\partial c_j} = \sum_{i=1}^{n} a_{ij}[-2\|x_i - c_j\|]$$

Optimal for this step

- To get the extreme value, we have:

$$\hat{c}_j = \frac{\sum_{i=1}^{n} a_{ij} x_i}{\sum_{i=1}^{n} a_{ij}}$$

- Or formally:

$$\hat{C} = \arg\min_{C} J(X;C,A) \Leftrightarrow J(X;C,A) \geq J(X;\hat{C},A), \forall A$$

K-Mean Algorithm

1. Initialize
   - Select initial *m* cluster centers
2. Find associations
   - For each x_i, assign the cluster with nearest center
   - ➔ Find A to minimize J(X; C, A) with fixed C
3. Find centers
   - Compute each cluster center as the mean of data in the cluster
   - ➔ Find C to minimize J(X; C, A) with fixed A
4. Stopping criterion
   - Stop if clusters stay the same. Otherwise go to step 2.

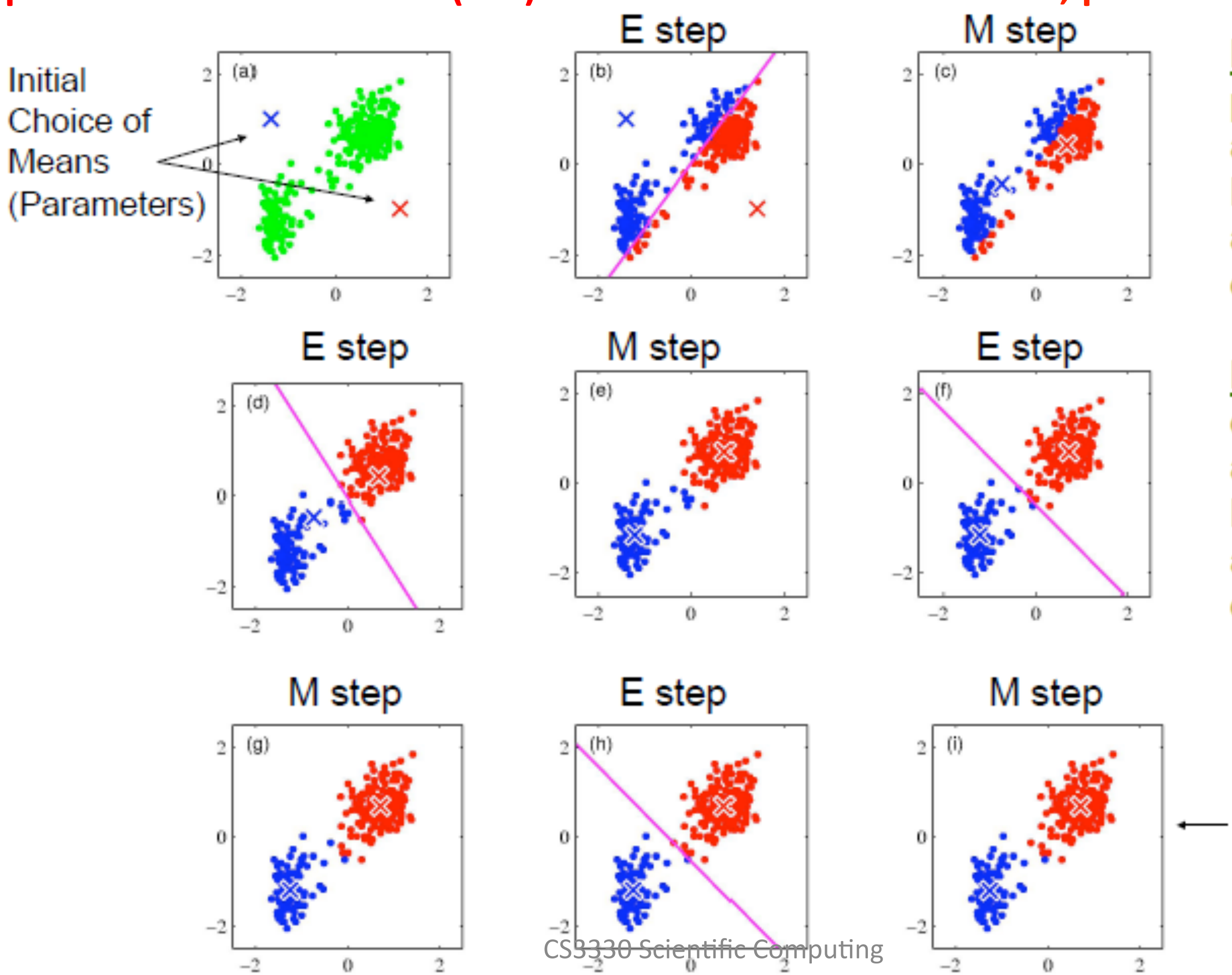
Stopping Criteria

- Two stopping criteria
  - Repeating until no more change in cluster assignment
  - Repeat until distortion improvement is less than a threshold
- Fact: Convergence is assured since J is reduced repeatedly ← Distortion is monotonically nonincreasing

$$J(X;C_1,\_) \geq J(X;C_1,A_1) \geq J(X;C_2,A_1) \geq J(X;C_2,A_2) \geq J(X;C_3,A_2) \geq J(X;C_3,A_3) \geq \cdots$$

How K-Means Works

Expectation Maximization (EM): distributions ← association, parameters ← centers



E step:
parameters
are fixed
Distributions
are
optimized

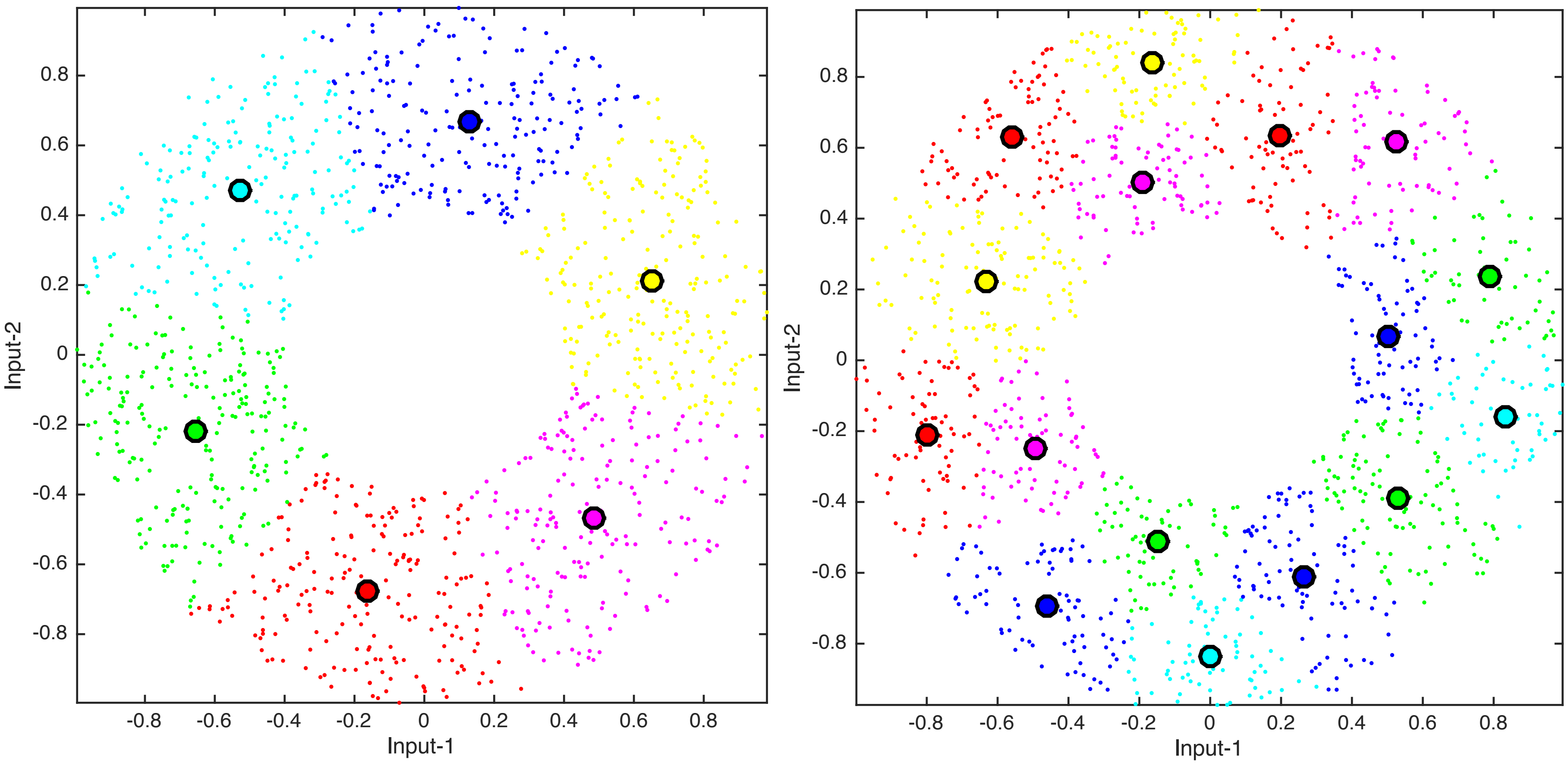
M step:
distributions
are fixed
Parameters
are
optimized

Final
Clusters
And
Means

Demo of K-Mean Clustering

- Download the (demo version of) the Machine Learning Toolbox from Prof. Jang's website
  - http://mirlab.org/jang/matlab/toolbox/machineLearning/
- Try the two demos
  - kMeansClustering.m ← animations of k-means algorithm
  - vecQuantize.m ← clustering versus quantization

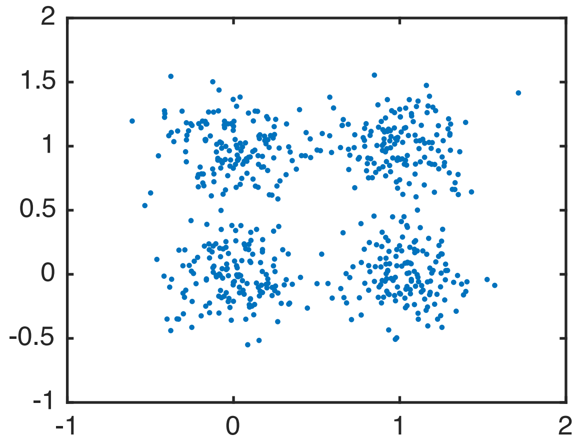
Demo of K-Mean Clustering (cont.)

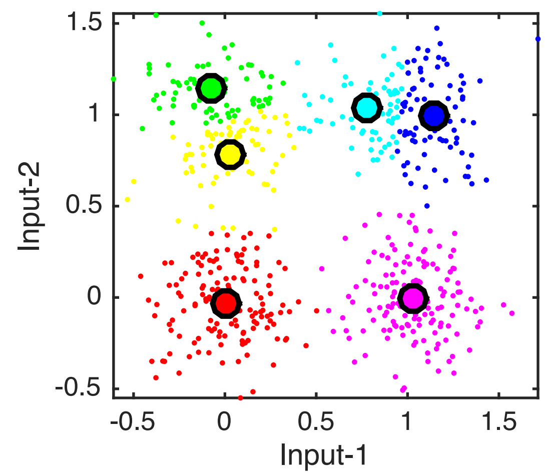


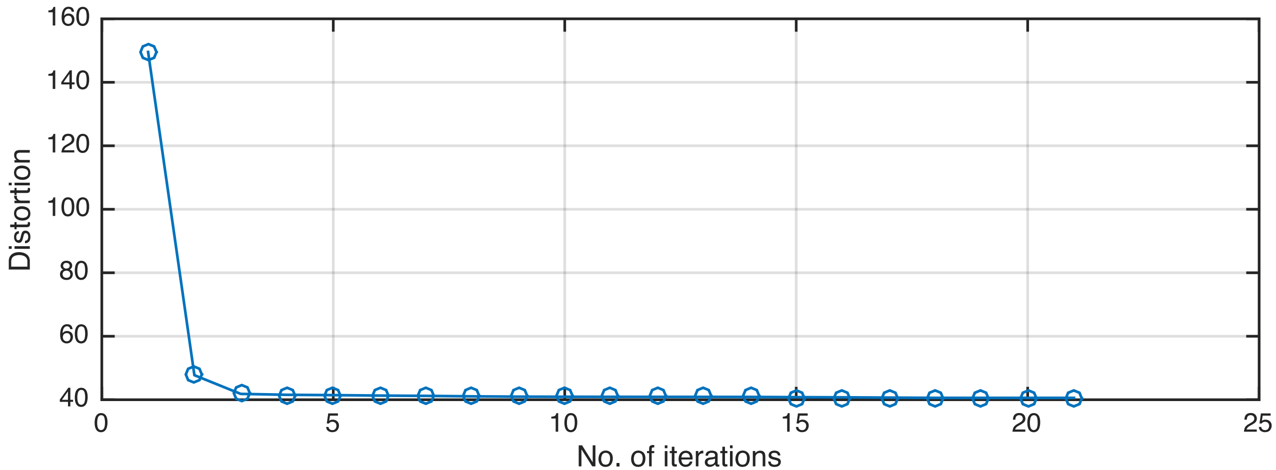
Sample Code

```
% ====== Get the data set
DS = dcData(5);
subplot(2,2,1);
plot(DS.input(1,:), DS.input(2,:), '.');
% ====== Run kmeans
centerNum=6;
[center, U, distortion, allCenters] = kMeansClustering(DS.input, centerNum);
% ====== Plot the result
subplot(2,2,2);
vqDataPlot(DS.input, center);
subplot(2,1,2);
plot(distortion, 'o-');
xlabel('No. of iterations'); ylabel('Distortion'); grid on
```

Sample Code #1







Discussions

- While the distortion is monotonically nonincreasing, we don't always get the global minimum
  - Solution: try a few random initial centers
  - Alternate solution: select initial centers as the dataset points with the largest sum of pairwise squared distance ← intuitively good, but still no guarantees

Discussions (cont.)

- It is possible that during the K-means iterations, one of the clusters has zero dataset point
  - Solution: split a cluster into two, different heuristics are possible, e.g., cluster with the maximal number of dataset points
- What we introduced is called batch K-means algorithm
  - There is also an online version existing, also known as sequential K-means algorithm

Image Compression: An Application

- Convert a image from true colors to indexed colors with minimum distortion
- Steps:
  - Collect data from a true-color image
  - Perform k-means clustering to obtain cluster centers as the indexed colors
  - Map each pixel's true color into indexed color

Recap: True- versus Indexed-Colors

True-color image

- Each pixel is represented by a vector of 3 components [R, G, B]

Index-color image

- Each pixel is represented by an index into a color map

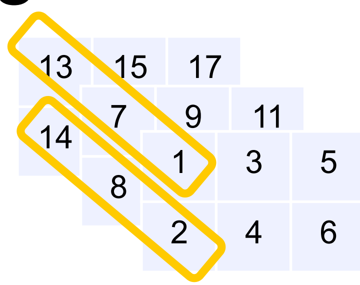
Read the Image, Check the Size

```
X = imread('minion.jpg');
image(X);
[m, n, p]=size(X)
```

- 640 x 640 x 3 matrix
- Check the color
  - dec2hex(X(200,200,:))
  - dec2hex(X(300,300,:))

How to Apply K-Means?

- (x, y, :) are the RGB values of a single pixel ← A sample in a 3-dim space!
- Have to convert a pixel into a column of a 2-D array
- Example: Indexing of pixels for an 2 x 3 x 3 image



$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \end{bmatrix}$$

- Related command (exercise): reshape

How to Apply K-Means? (cont.)

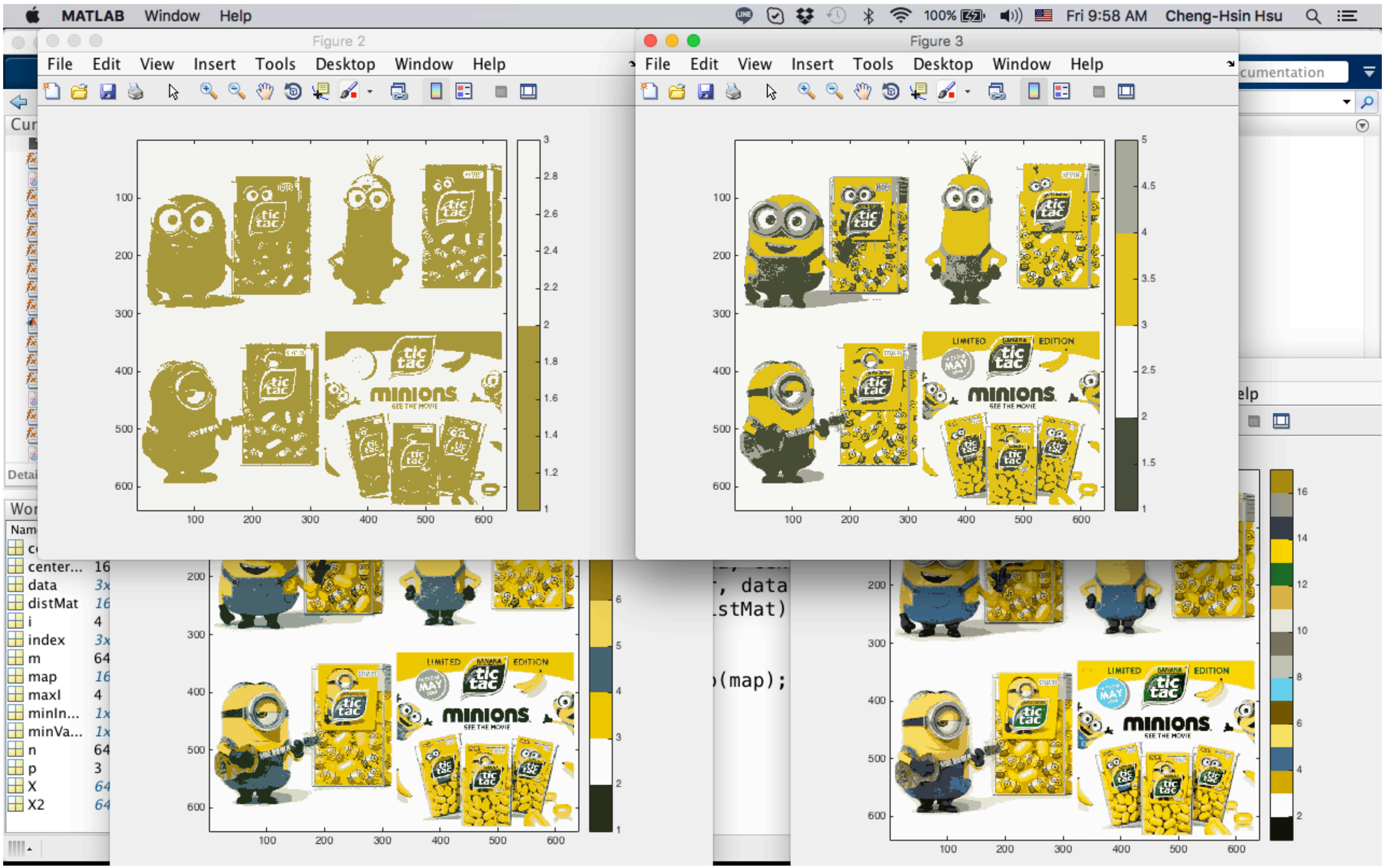
- index=reshape(1:m*n*p, m*n, 3)';
- >> size(index)
- ans = 3   409600

- Now we have 409600 samples, find the centers using K-means algorithm

(Partially-Working?) Code

```
X = imread('minion.jpg');
image(X)
[m, n, p]=size(X);
index=reshape(1:m*n*p, m*n, 3)';
data=double(X(index));
maxI=4;
for i=1:maxI
      centerNum=2^i;
      fprintf('i=%d/%d: no. of centers=%d\n', i, maxI, centerNum);
      center=kMeansClustering(data, centerNum);
      distMat=distPairwise(center, data);
      [minValue, minIndex]=min(distMat);
      X2=reshape(minIndex, m, n);
      map=center'/255;
      figure; image(X2); colormap(map); colorbar; axis image;
end
```

Results


MATLAB Window Help
Fri 9:58 AM Cheng-Hsin Hsu
Figure 2
Figure 3
File Edit View Insert Tools Desktop Window Help
minions
LIMITED EDITION
center... 16
data
distMat
i 4
index
m 64
map
maxI 4
minIn...
minVa...
n 64
p 3
X
X2
, data
stMat)
(map);

Compression Ratio

$$before = m * n * 3 * 8 \; bits$$

$$after = m * n * \log_2(c) + c * 3 * 8 \; bits$$

$$\rho = \frac{before}{after} = \frac{m * n * 3 * 8}{m * n * \log_2(c) + c * 3 * 8} = \frac{24}{\log_2(c) + \frac{24c}{m * n}} \approx \frac{24}{\log_2(c)}$$

Note: Compared to raw 8-bit RGB image,
not PNG (lossless) nor JPG (lossy)

Matlab #5 Homework (M5)

- (3% + 1% Bonus) A true-color image of size mxn is represented by mxn pixels, each consists of 24 bits of RGB colors. On the other hand, each pixel of an mxn index-color image is represented by an p-bit unsigned integer, which serves as an index into a color map of size 2^p by 3. In this exercise, we use k-means clustering to convert a true-color image into an index-color image. In particular, the goal of this exercise is to display the images after data compression using k-means clustering.

Matlab #5 Homework (M5) cont.

- (2%) Convert the minions true-color image into an index-color one using k-means clustering on each pixel represented as a vector of 3 elements of RGB intensity. Let k take the values of 2, 4, 8, 16, 32, 64 and display the converted index-color images in your report
- (0.5%) How long does it take to run each k-means clustering?
- (0.5%) What are the distortion of each images?
- (0.5%) What is the compression ratio of the 6 resulting images?
- (0.5%) Is the compression lossy or lossless?

Matlab #5 Homework (M5) cont.

- Hints:
  - You need to read the original image and convert it into a 3x307200 data matrix of "double", where each column is the RGB vector of an pixel
  - Use *imread* to read an image, *image* to display an image, and *colorbar* to display the color map
  - Use *kMeansClustering* for k-means clustering
  - Use *distPairwise* to find the distance between two sets of vectors