



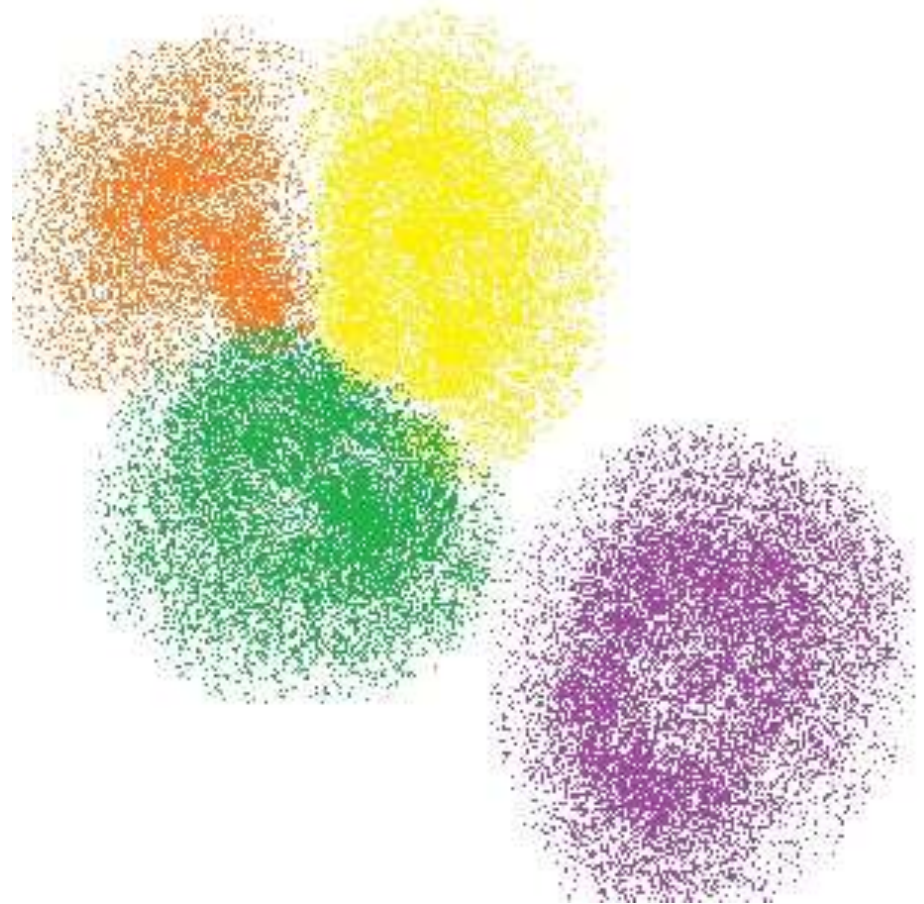
k-means Clustering

Mr. Bryan Bagnall
SPAWAR Systems Center, Pacific
Phone: 619-553-4061
Email: bryan.bagnall@navy.mil

Mr. Sparta Cheung
SPAWAR Systems Center, Pacific
Phone: 619-553-5927
Email: sparta.cheung@navy.mil

Overview of Talk

- Description
- Algorithm
- Notes
- OpenCV documentation
- Coding an example

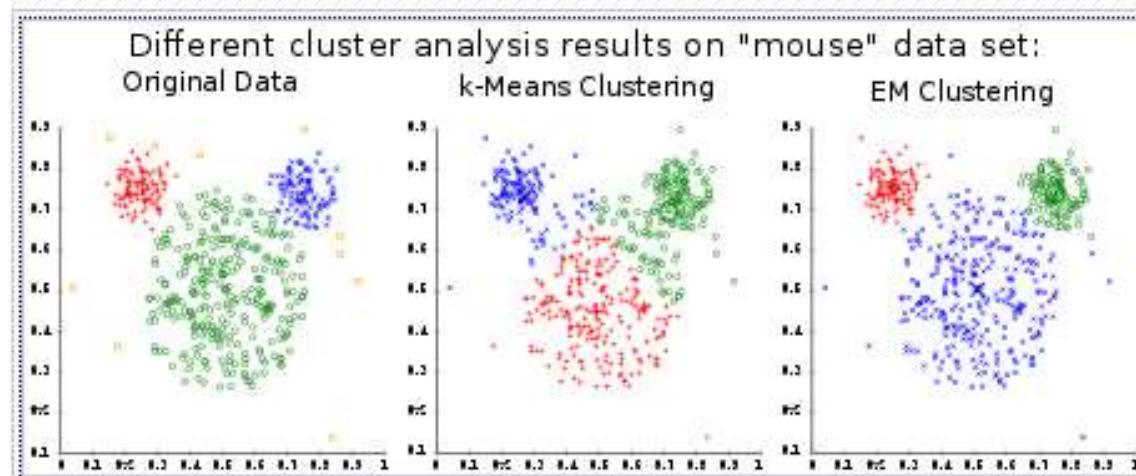


k-means Clustering

Method of cluster analysis, also known as *Lloyd's Algorithm*

The goal of *k-means* clustering are:

- Partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean



k-means Clustering

Given a set of observations (x_1, x_2, \dots, x_n) where each observation is a d -dimensional real vector

Then partition the n observations into k sets ($k \leq n$), $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within cluster sum of squares

$$\arg \min_S \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2, \text{ where } \mu_i \text{ is the mean of points in } S_i$$

k-means Clustering

The most common algorithms use an iterative refinement technique

Step #1: Assign each observation (x) to the cluster with the closest mean

$$S_i^{(t)} = \{x_j : \|x_j - m_i^{(t)}\| \leq \|x_j - m_{i^*}^{(t)}\| \text{ for all } i^* = 1, \dots, k\}$$

Step #2: Calculate the new means to be the centroid of the observations in the cluster

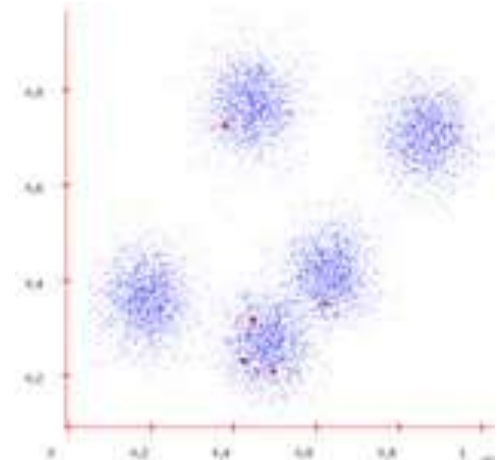
$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

The algorithm converges when the assignments no longer change

k-means Clustering

Step 1:

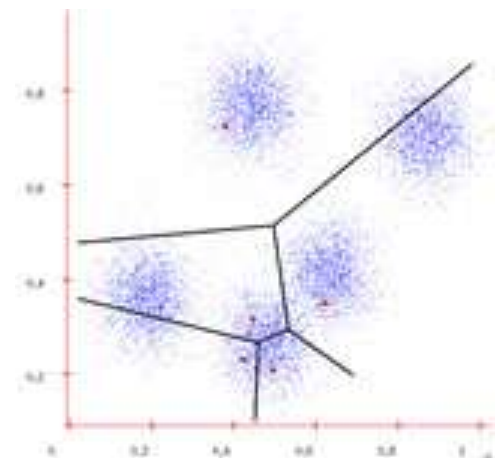
k initial means, $k = 5$ randomly selected *means* from the data set



Step 2:

k clusters are created by associating every observation (x) with the nearest mean

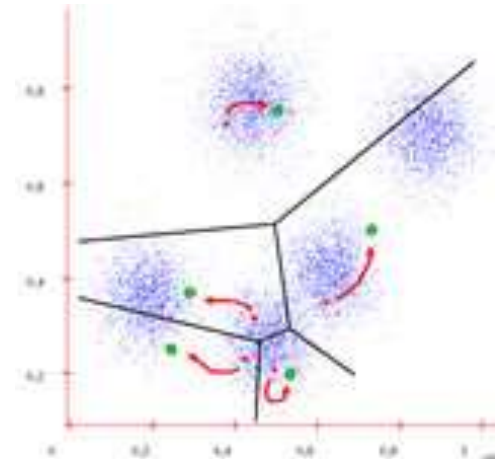
Each datapoint checks to see which center it is closest to



k-means Clustering

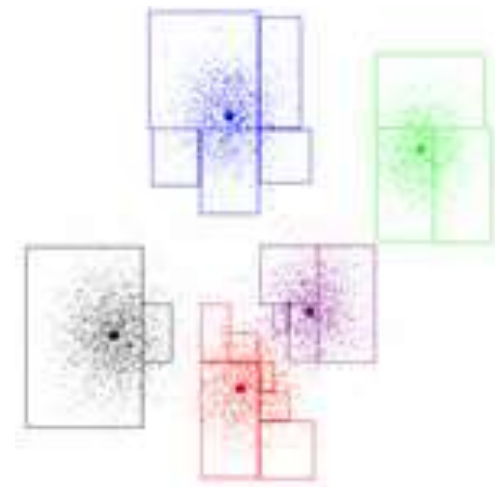
Step 3:

The center uses each of the points it “owns” to calculate a new center, then it shifts to that new center



Step 4...:

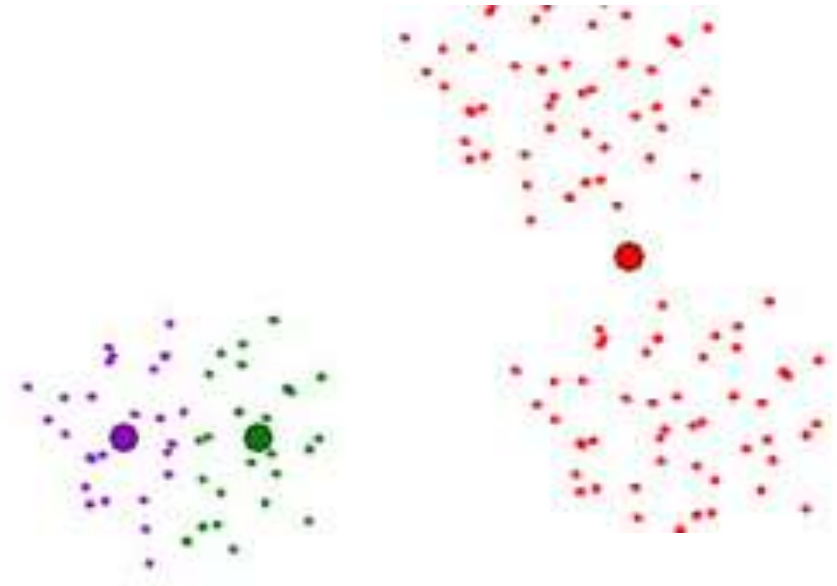
Steps 2 and 3 are repeated until the algorithm converges



k-means Clustering

Some notes about using *k-means*

The number of clusters is typically an input parameter, the wrong parameter will yield poor results



The cluster model is based on spherical clusters, clusters are expected to be of similar size

It is often used as a preprocessing step for other algorithms

cv::kmeans Function Documentation

Finds the centers of clusters and groups the input samples around the clusters

http://opencv.willowgarage.com/documentation/cpp/clustering_and_search_in_multi-dimensional_spaces.html

```
double kmeans( const Mat& samples, int clusterCount, Mat& labels,  
               TermCriteria termcrit, int attempts, int flags, Mat* centers)
```

The function `kmeans` implements a k-means algorithm that finds the centers of *clusterCount* clusters and groups the input *samples* around the clusters.

cv::Canny Function Documentation

Finds the centers of clusters and groups the input samples around the clusters

http://opencv.willowgarage.com/documentation/cpp/clustering_and_search_in_multi-dimensional_spaces.html

```
double kmeans( const Mat& samples, int clusterCount, Mat& labels,  
               TermCriteria termcrit, int attempts, int flags, Mat* centers)
```

Parameters:

samples – Floating-point matrix of input samples, one row per sample

clusterCount – The number of clusters to split the set by

labels – The input/output integer array that will store the cluster indices per sample

termcrit – Specifies the maximum number of iterations

attempts – The number of times that the algorithm is executing using different
initial labelings

cv::Canny Function Documentation

Finds the centers of clusters and groups the input samples around the clusters

http://opencv.willowgarage.com/documentation/cpp/clustering_and_search_in_multi-dimensional_spaces.html

```
double kmeans( const Mat& samples, int clusterCount, Mat& labels,  
               TermCriteria termcrit, int attempts, int flags, Mat* centers)
```

Parameters:

flags –

KMEANS_RANDOM_CENTERS – use random centers for each attempt

KMEANS_PP_CENTERS – use kmeans++ center initialization

KMEANS_USE_INITIAL_LABELS – use user supplied initial labels

centers– The output matrix of the cluster centers, one row per cluster center

cv::Canny Function Documentation

Finds the centers of clusters and groups the input samples around the clusters

http://opencv.willowgarage.com/documentation/cpp/clustering_and_search_in_multi-dimensional_spaces.html

```
double kmeans( const Mat& samples, int clusterCount, Mat& labels,  
               TermCriteria termcrit, int attempts, int flags, Mat* centers)
```

There is an OpenCV kmeans example in

..\Day02\codigo\opencv_samples\samples\kmeans.c

Questions?