SPAWAR
Systems Center
PACIFIC

k-means Clustering

SPAWAR Systems Center PACIFIC

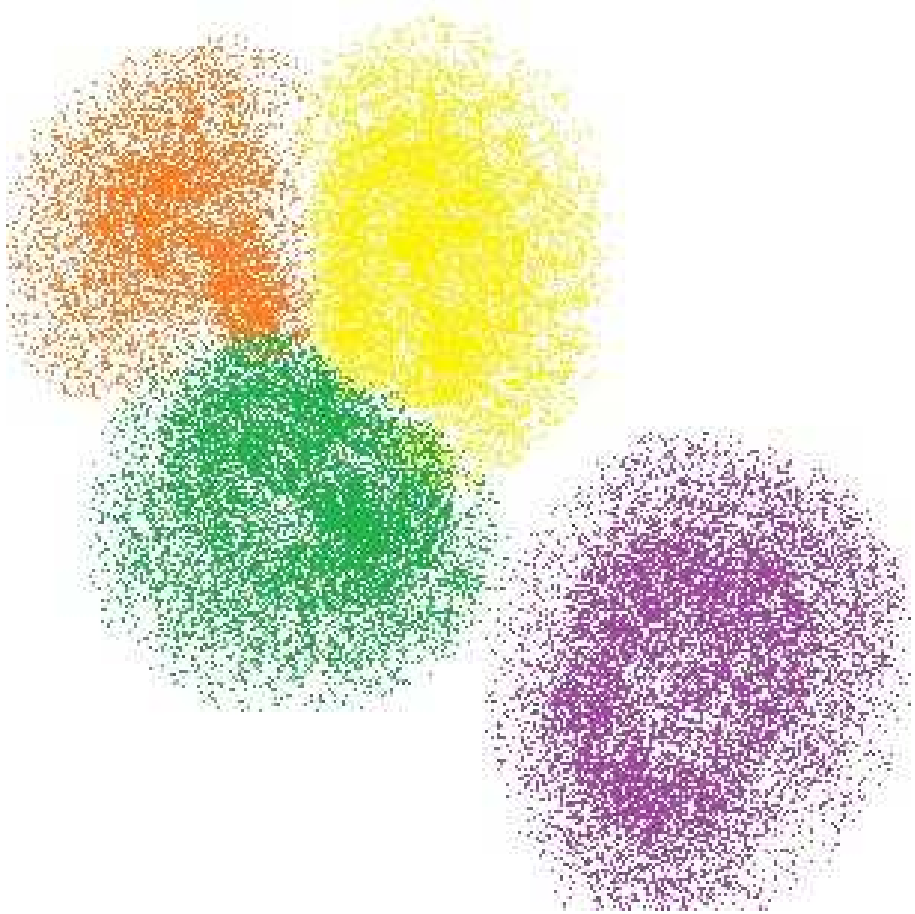
Disclaimer

- Though SSC Pacific makes every effort to perform quality assurance on its training materials, the material in this presentation may inadvertently include technical inaccuracies or other errors. We would be grateful if users notify us of any errors or inaccuracies they may find.
- The presentation contains references to links and to third-party websites. These are provided for the convenience and interest of users and this implies neither responsibility for, nor approval of, information contained in these websites on the part of the U.S. Government. The USG makes no warranty, either express or implied, as to the accuracy, availability or content of information, text, graphics in the links/third party websites. The USG has not tested any software located at these sites and does not make any representation as to the quality, safety, reliability or suitability of such software, nor does this presentation serve to endorse the use of such sites.

SPAWAR Systems Center PACIFIC

Overview of Talk

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

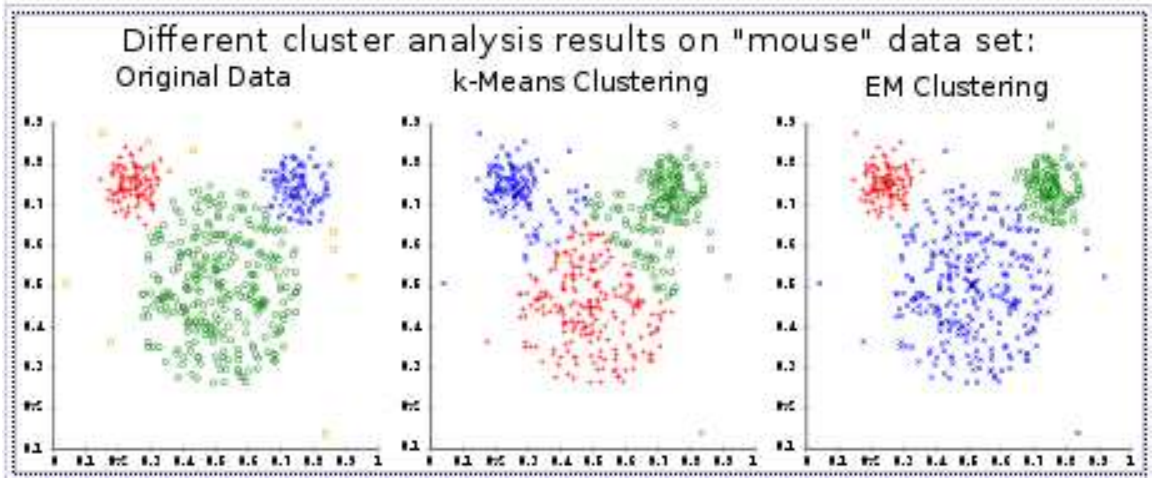


k-means Clustering

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



SPAWAR Systems Center PACIFIC

k-means Clustering

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

Then parition the *n* observations into *k* sets (k<=n), **S** = {$S_1, S_2, \ldots, S_k$} so as to minimize the within cluster sum of squares

$$\arg\min_{S} \sum_{x=1}^{k} \sum_{x_j \epsilon S_i} \|x_j - \mu_j\|^2, \quad where\ \mu_i\ is\ the\ mean\ of\ points\ in\ S_i$$

SPAWAR Systems Center PACIFIC

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

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

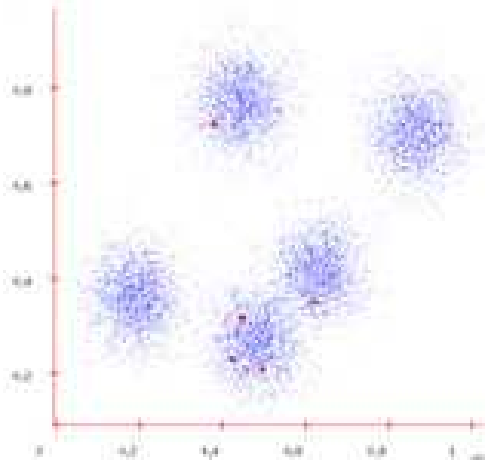
Step #2: Calculate the new means to be the centroid of the observations in the cluster

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

The algorithm converges when the assignments no longer change

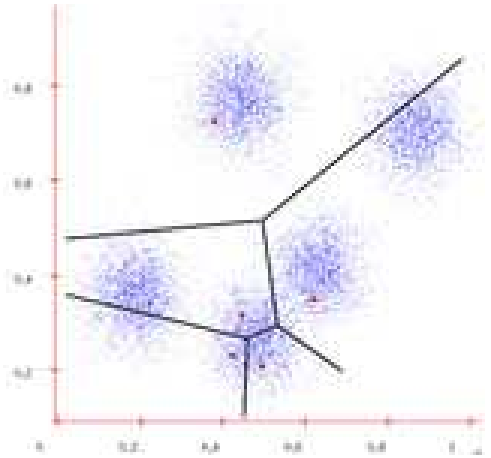
k-means Clustering

Step 1:
k initial means, *k* = 5 randomly selected *means* from the data set



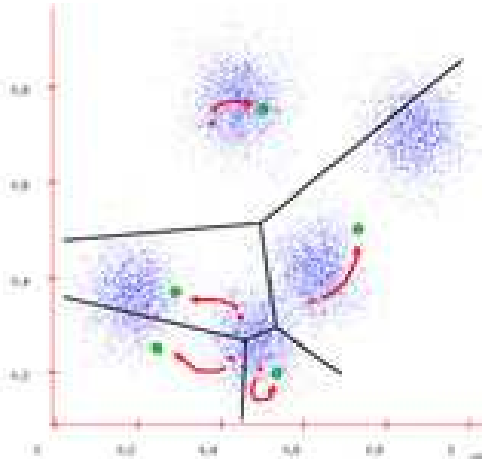
Step 2:
k clusters are created by associating every observation (*x*) with the nearest mean

Each datapoint checks to see which center it is closest to

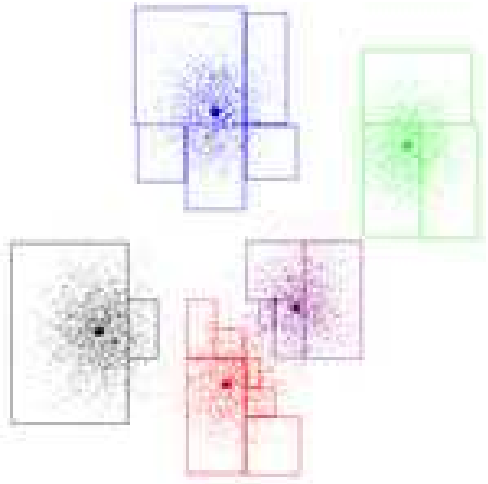


k-means Clustering

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

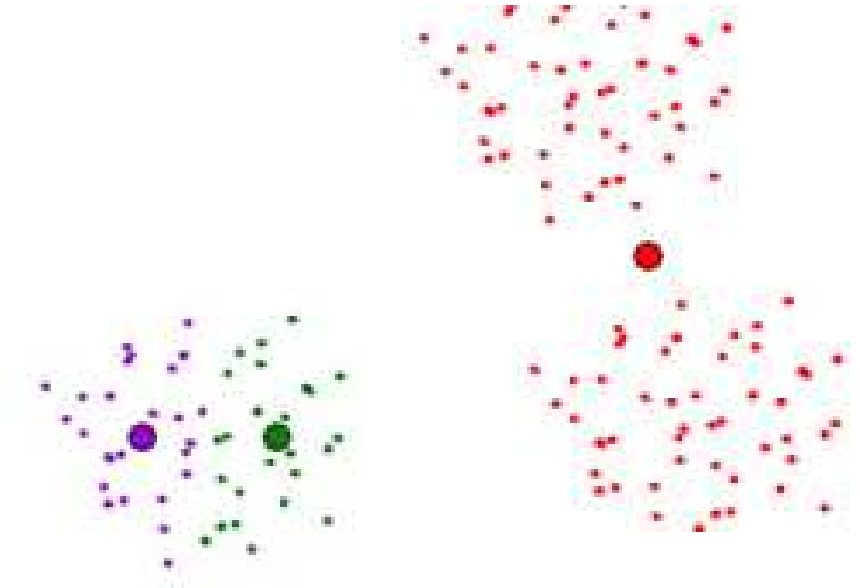


SPAWAR Systems Center PACIFIC

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

SPAWAR
Systems Center
PACIFIC

Questions?