Geometric Data Analysis

Clustering

Ioannis Emiris

NKUA, and Athena RC

Fall 2020

Contents

Outline

1. Clustering
   - Vector spaces
   - Improved k-means
   - Arbitrary (non-vector) metric spaces
   - Improvements and Evaluation

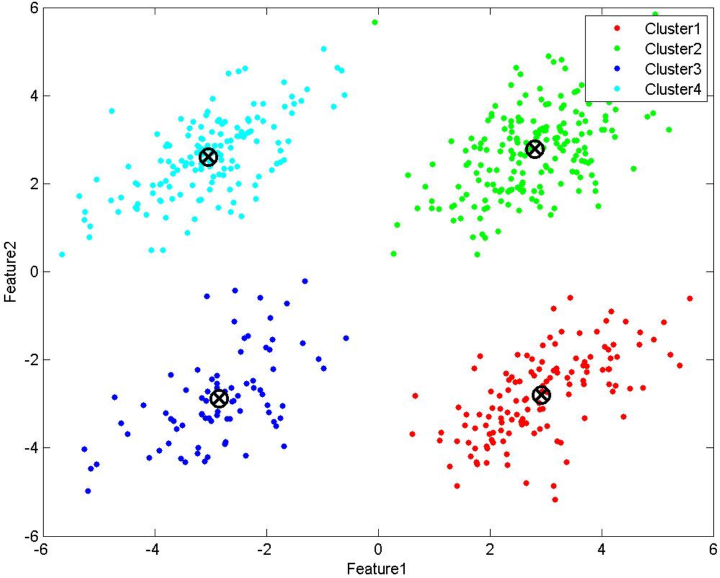
Clustering

Definition (k clusters)

Given n objects, and $k > 1$, partition the objects into k subsets (clusters) so as to optimize some objective function.

- Objects in the same cluster are more "similar" (or closer) to each other than to those in other clusters.
- Possible criteria: minimizing the total distance among all cluster points, minimizing the distance of cluster points to some center, etc.
- Variations: k is unknown and computed, e.g., by the Silhouette method. Capacitated/balanced: k given, clusters of equal cardinality.

Good Clustering, with centers



Approaches

- hierarchical (agglomerative): each point initializes a cluster, merge closest pair (define distance of clusters) until stopping criterion, e.g., predetermined number of clusters, or clusters with points too far apart (Gromos).
- point-assignment: given some initial clusters, assign points to "best" cluster; cluster represented by ``centroid" (which may not belong to input). Example: k-means.

(Ullman et al.:Mining Massive datasets)

Outline

k-means: Objective function

Typical ambient space is $\mathbb{R}^d$ but can generalize to metric space $\mathcal{Z}$.

Minimization function

In any metric space over points/vectors $\mathcal{Z}$ with distance/metric function d, let the dataset be $X = \{x_1, \ldots, x_n\} \subseteq \mathcal{X} \subseteq \mathcal{Z}$, $k > 1$. Given centroids $C \subset \mathcal{Z}$, let

$$\mathrm{d}(x_i, C) = \min_{c \in C} \mathrm{d}(x_i, c).$$

Consider vector $v(C) = (\mathrm{d}(x_1, C), \ldots, \mathrm{d}(x_n, C))$. The k-means objective is:

$$\min_{C \subseteq \mathcal{Z}, |C| = k} \|v(C)\|_2^2 = \sum_{i=1}^{n} \mathrm{d}(x_i, C)^2.$$

The k-means objective is NP-hard, but for the ℓ_2 metric, Lloyd's algorithm converges quickly to a *local* minimum.

Variations

Various minimizations

Recall $X = \{x_i\}$, $v(C) = (\mathrm{d}(x_1, C), \ldots, \mathrm{d}(x_n, C))$, $C \subset \mathcal{Z}$ are centroids.
For $\mathrm{d}(\cdot)$ denoting ℓ_2 distance, the k-means objective is:

$$\min_{C \subseteq \mathcal{Z}, |C| = k} \|v(C)\|_2^2 = \sum_{i=1}^{n} \mathrm{d}(x_i, C)^2.$$

Other standard objectives:
-- k-median: $\min_{C \subseteq \mathcal{Z}, |C|=k} \|v(C)\|_1$,
-- k-medoid: $\min_{C \subseteq X, |C|=k} \|v(C)\|_1$.
-- k-center: $\min_{C \subseteq X, |C|=k} \|v(C)\|_\infty$,

Lloyd's

Algorithm

Initialize k centers randomly (or using some strategy).

1. Assignment: Assign each object to its nearest center.
2. Update: Calculate mean $\frac{1}{T}\sum_{i=1}^{T}\vec{v}_i$ of each cluster, make it new center.

Repeat the two steps until there is no change in the assignments.

Properties

- Each distance calculation $= O(d)$ because vectors in $\mathbb{R}^d$.
- Assignment $= O(nkd)$, Update $= O(nd)$,
- $\#$iterations unknown, in practice $\ll n$.
- Converges to local minimum in Euclidean space (depends on initialization)

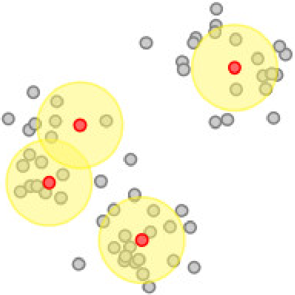
Outline

1. Clustering
   - Vector spaces
   - Improved k-means
   - Arbitrary (non-vector) metric spaces
   - Improvements and Evaluation

Inverted Quantized k-means (IQ-means)

Reverse assignment

- Fixed Data-structure for points (Dataset in memory)
- Centroids are queries; range search of increasing radius
- Resolve overlapping balls; consider ``left-overs".



(Avrithis-Anagnostopoulos-Kalantidis-E,ICCV'15)

Assignment by Range search

Reverse approach (ANN)

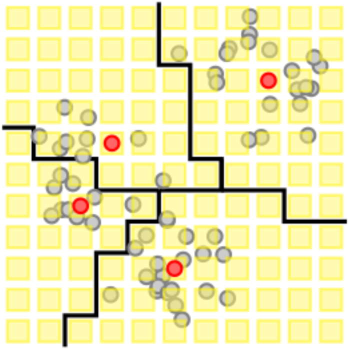
- Index n points, but only once for entire algorithm.
- At each iteration, for each centroid c, range/ball queries centered at c.
- Mark assigned points: move at end of bucket, or flag them.
- Increase radii by $\times 2$, start with min(dist between centers)/2, until all points assigned, or most ranges/balls do not assign a new point.
- For a given radius, if a point lies in ≥ 2 balls, compare its distances to the respective centroids, assign to closest centroid.
- At end: for every unassigned point, compare its distances to all centroids

- Standard method: n ANN queries, each k^ρ, hence $O(nk^\rho)$.
- Inverse: k queries, each $n^\rho+\text{OutputSize} = O(n)$; stores entire dataset
- probabilistic analysis

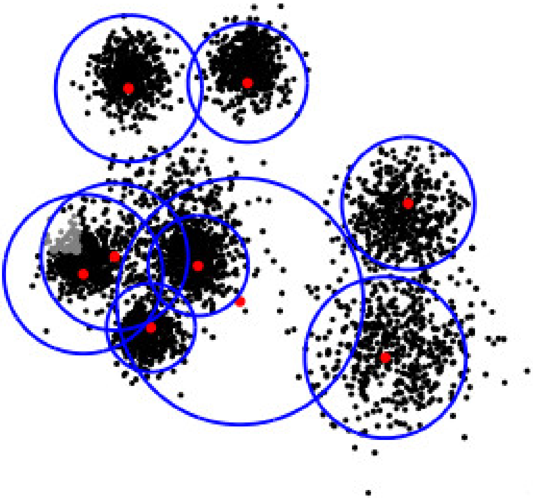
Inverted Quantized k-means (IQ-means)

The algorithm

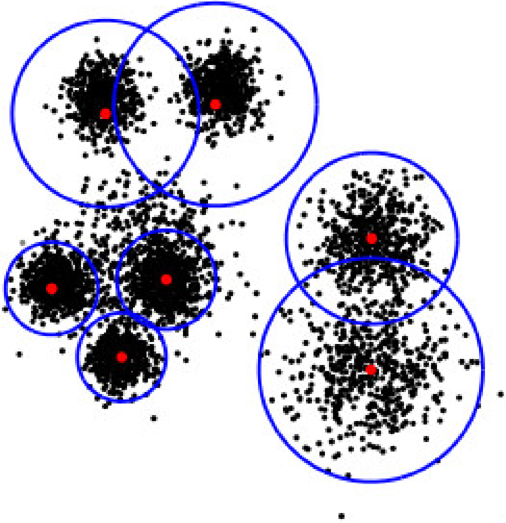
- inverse assignment (above): faster than update!
- quantization on dynamic 2d-grid (via learning) (Avrithis:ICCV'13)
- dynamic estimation of overlap hence of k (Avrithis-Kalantidis,ECCV'12)



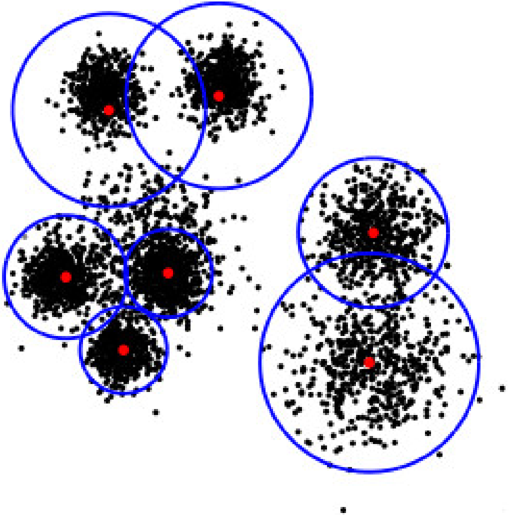
Dynamic IQ-means (k=9)



Dynamic IQ-means (k=7)



Dynamic IQ-means (k=7)

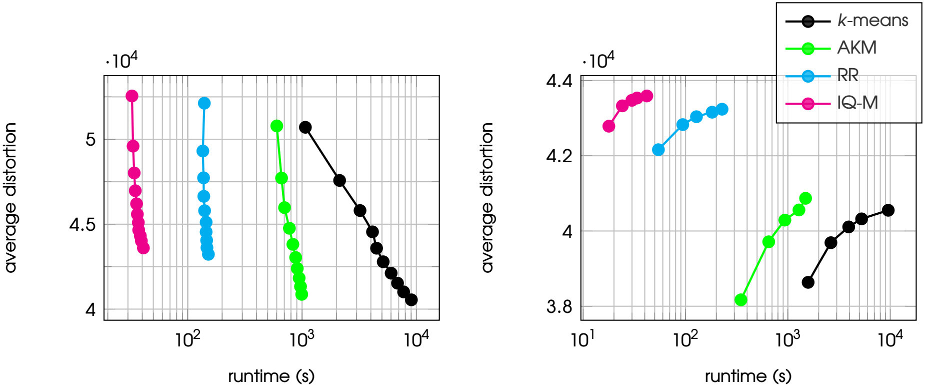


Experiments

- (Avrithis,Kalantidis,Anagnostopoulos,E:ICCV'15)
  http://github.com/iavr/iqm
- Comparison against
  -- AKM: Approximate k-means (Philbin et al. CVPR'07)
  -- RR: Ranked Retrieval (Broder et al. Web Search & Data Mining'14)
  -- **standard** k-means
- Speed: IQ-means fastest
- Accuracy: IQ-M on par with dedicated methods, worse than (approx) k-means
- clustering of 100M images, on a single machine, in <1 hour.
  Best method for a couple of years.

Performance

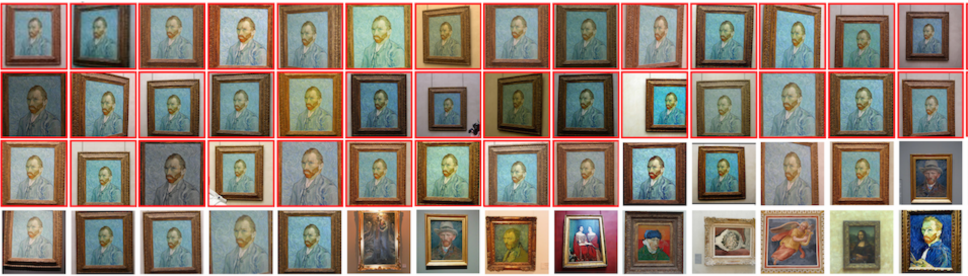
Distortion vs total time for 20 iterations on 10^6 images (SIFT1M):



Left: varying number of clusters k. Right: increasing number of points n.

Mining example

500K Paris + 100Mil YahooFlicker images



Accurate cluster despite large dataset: Paris ground truth depicted in red outline, the rest are images closest to the red ones.

Outline

1. Clustering
   - Vector spaces
   - Improved k-means
   - Arbitrary (non-vector) metric spaces
   - Improvements and Evaluation

k-medoids

Goal: Handle any distance metric; k-means only if consistent with mean.

k-medoids (PAM is simplest algorithm) use centroids that **belong** to the dataset:

Definition (Medoid)

The medoid of a set is the object of the set that minimizes total dissimilarity (distance) to all other objects in the set.

Objective function (cf. above): Minimize sum of distances to point's centroid.

vs k-means

-- k-means tends to select convex spherical clusters; k-medoids less so.

-- k-means is more sensitive to noisy data and outliers.

-- k-means is faster and easier to implement.

(Kaufman-Rousseeuw'87)

Partitioning Around Medoids (PAM)

Initialize k centroids randomly.

1. Assignment: Assign each object to nearest centroid; compute objective
2. Update:

   for each centroid m **do**
       for each non-centroid t **do**
           Swap m and t, compute new objective function value.
       end for
   end for
   Keep configuration (centroids) with min objective value.

Repeat steps 1 and 2 until there is no change of configuration (centroids).

Let distance calculation $= O(d')$. Update of Objective $= O((n-k)d')$, if 2nd best centroid known. Hence, update $= O((n-k)^2 kd') \sim n^2$.

Accelerating updates

Two faster updates, which may however lose accuracy compared to PAM.
Recall that after every swap we compute J in $O((n-k)d')$.

1. Improved Update

Instead of swapping centroid m with every point t, swap m only with every non-centroid in same cluster as m.

Complexity: $n-k$ iterations instead of $k(n-k)$, hence update $= O((n-k)^2 d')$

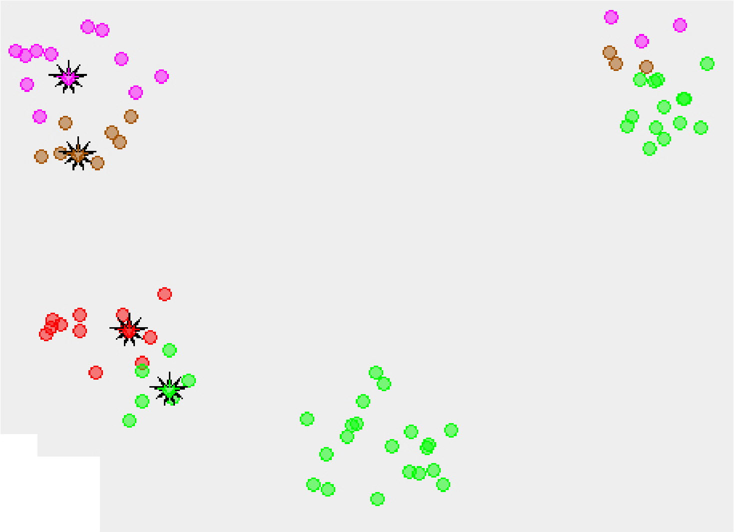
2. Update à la Lloyd's

For every cluster: (i) compute its medoid t, (ii) swap its current centroid m with t.

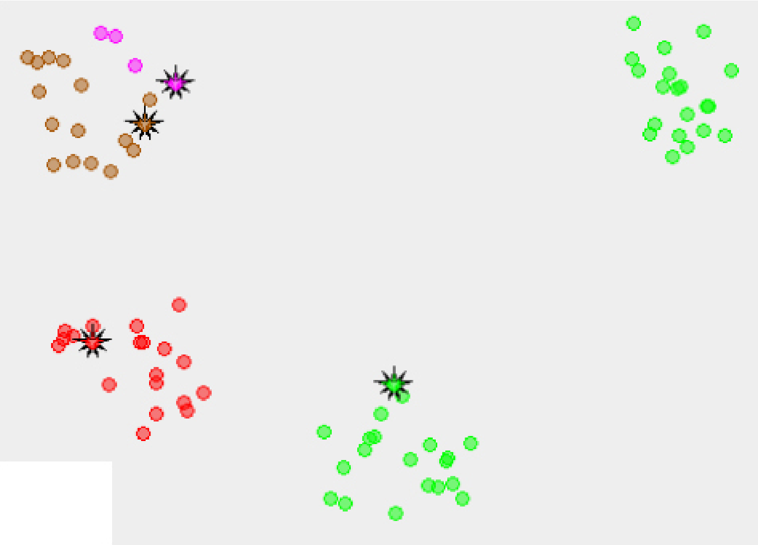
The medoid t minimizes $\sum_{i \in C} d(i,t)$ over all possible objects t in cluster C.
Computed in $O(a^2 d')$, assuming clusters have $a \simeq n/k$ items.

Total Complexity $= O((ka^2 + k(n-k))d') = O((n^2/k + nk)d') = O(n^2 d')$ (Park-Jun'09).

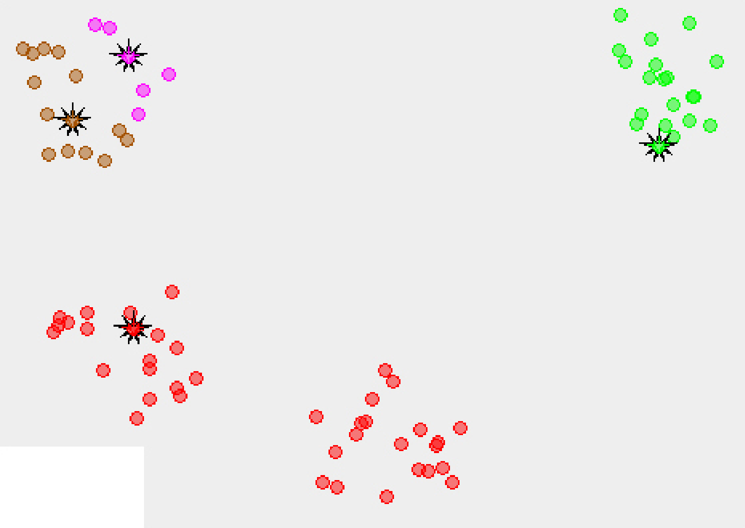
Initial



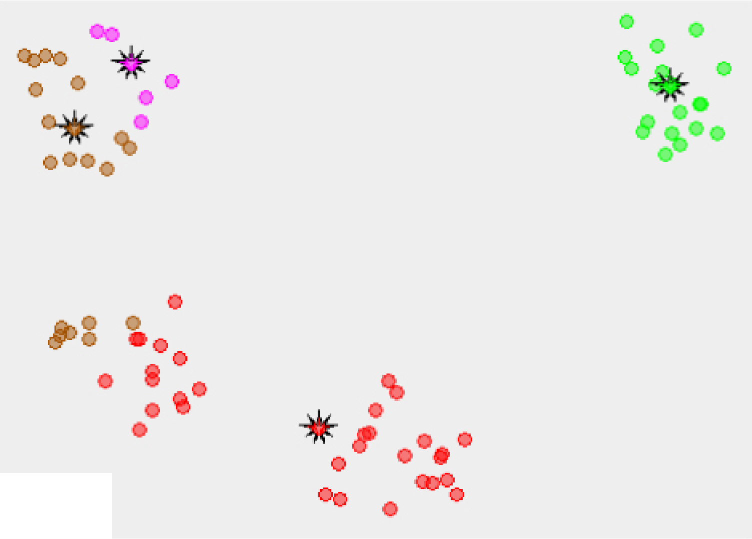
PAM 1



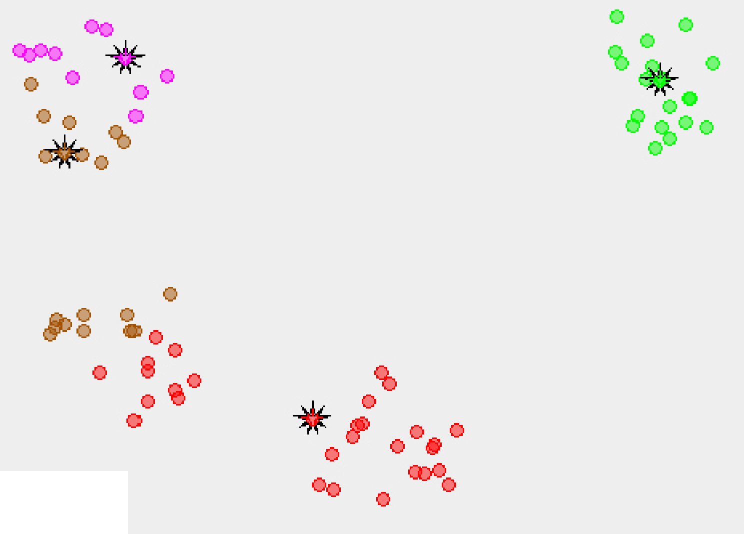
PAM 2



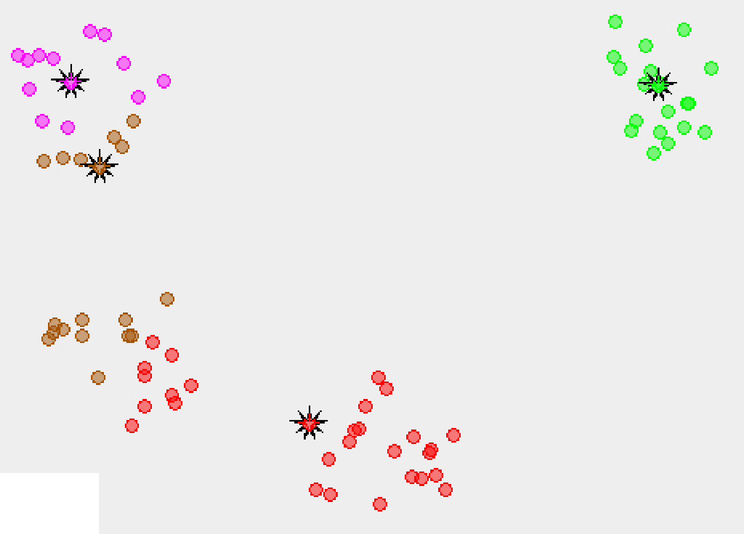
PAM 3



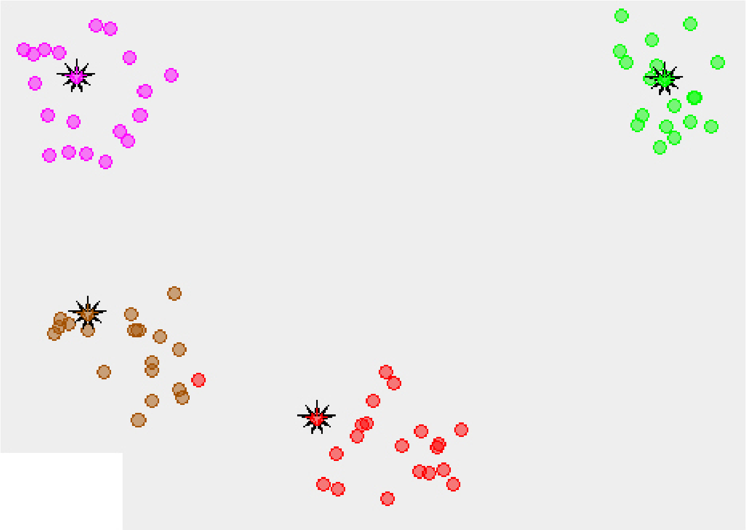
PAM 4



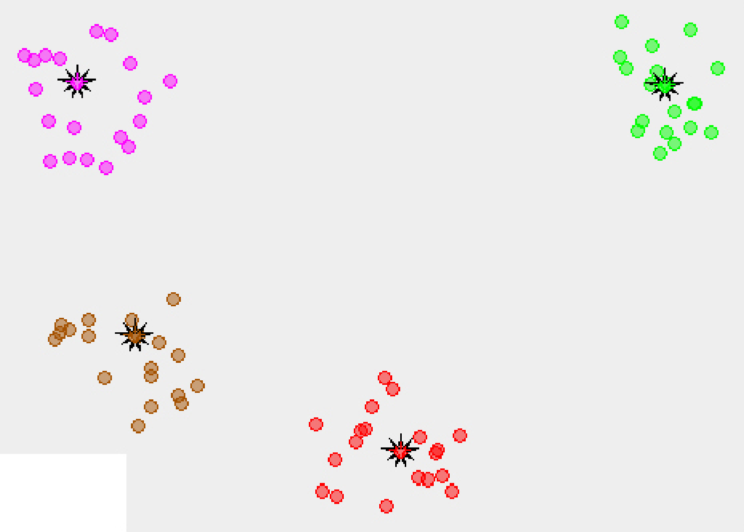
PAM 5



PAM 6



Final



Outline

1. Clustering
   - Vector spaces
   - Improved k-means
   - Arbitrary (non-vector) metric spaces
   - Improvements and Evaluation

Clustering Large Applications (CLARA)

General Idea: run entire algorithm with sample of size $n' \ll n$. Use s samples drawn independently, return best clustering.

Overall algorithm:

for $i = 1, \ldots, s$ **do**
  apply PAM on a random (uniform) sample of size n'
  assign n points to k computed centroids
  calculate the total cost of the partitioning
end for
return best partitioning

Experimental results recommend: $s = 5$, $n' = 40 + 2k$.

CLARA based on RANdomised Search (CLARANS)

- Update: swap m's with t's, for some randomly selected (m, t)'s only.
- Picking random $Q \subset \{1, \ldots, k\} \times \{1, \ldots, n-k\}$, s times.

Select k centroids by some initialization method.
for $i = 1, \ldots, s$ **do**
  Cluster $n - k$ points to k centroids by some assignment method.
  Randomly select set Q of $|Q|$ pairs (m, t), $|Q| < k(n-k)$.
  for $(m, t) \in Q$ **do**
    Swap m with t; compute new objective value.
  end for
  Keep centroids with minimum objective value over $|Q|$ choices.
end for
Output centroids yielding minimum objective value over s candidates.

Experiments recommend: $s = 2$, $|Q| = \max\{0.12 \cdot k(n-k), 250\}$.
(Ng-Han:IEEE Tran.Know.Data Eng'02, Theodoridis et al.:Patt.Recogn.,ch.14)

Improve Initialisation 1: Spread-out

initialization++ : k-means++ / k-medoids++:

(1) Choose a centroid uniformly at random; $t \leftarrow 1$.
(2) $\forall$ non-centroid point $i = 1, \ldots, n-t$, let $D(i) \leftarrow \min$ distance to some centroid, among t chosen centroids.
(3) Choose new centroid: r chosen with probability proportional to $D(r)^2$:

$$\text{prob}[\text{choose } r] = D(r)^2 / \sum_{i=1}^{n-t} D(i)^2.$$

Let $t \leftarrow t + 1$.
(4) Go to (2) until $t = k =$ given #centroids.

Expected approximation ratio $= O(\log k)$ (Arthur-Vassilvitskii:SODA'07)
Similar algo for 2-approx of k-center (NP-hard prob)

Improve Initialisation 2: Concentrate

Select centroids close to dataset's center of mass (and to each other) as follows.

(1) Calculate symmetric $n \times n$ distance matrix of all objects, i.e. all distances d_{ij} from every object $i = 1, \ldots, n$ to every other object $j = 1, \ldots, n, i \neq j$.
(2) For object i compute

$$v_i = \sum_{j=1}^{n} \frac{d_{ij}}{\sum_{t=1}^{n} d_{jt}}, \qquad i = 1, \ldots, n.$$

(3) Return the k objects with k smallest v_i values.

Algorithm proposed in (Park-Jun'09).

Evaluation: Silhouette

-- For $1 \le i \le n$, $a(i) =$ average distance of i to other objects in same cluster.
-- Let $b(i) =$ average distance of i to objects in *next best* (neighbor) cluster, i.e. cluster of 2nd closest centroid.

Silhouette of Object i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} = \left\{ \begin{array}{ll} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{array} \right\} \in [-1, 1].$$

Interpret silhouette

$s(i) \to 1$: i seems correctly assigned to its cluster;
$s(i) \simeq 0$: borderline assignment (but not worth to change);
$s(i) \to -1$: i would be better if assigned to next best cluster.

Silhouette: Cluster and clustering

Specific clusters

-- Evaluate a cluster: Compute average $s(i)$ over all i in some cluster.
-- If k is too large or too small, some clusters shall display much smaller silhouettes than the rest.
-- Silhouette plots are used to improve k: try different k's and see if clusters have roughly equal silhouettes.

Overall Clustering

Overall Silhouette coefficient = average $s(i)$ over $i = 1, \dots, n$.

High if well clustered, low may indicate bad k (or existance of outlier points).