Ανάπτυξη Λογισμικού για Δυσεπίλυτα Αλγοριθμικά Προβλήματα

Ενότητα 2: Clustering

Γιάννης Εμίρης

Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Πανεπιστήμιο Αθηνών

November 10, 2016

Outline

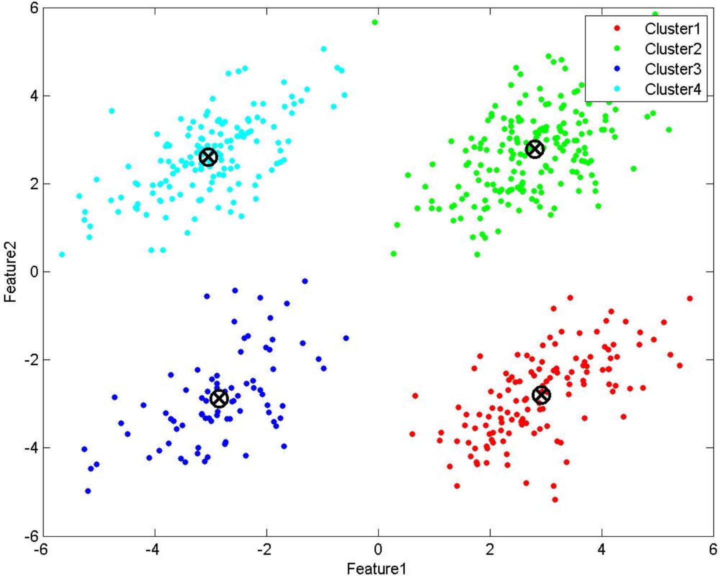
Clustering

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.
- Applications: Classification, Social Network Analysis, Recommender Systems, Market Research, Bioinformatics etc

Good Clustering, with centers



Approaches

- hierarchical (agglomerative): each point initializes a cluster, merge until stopping criterion, e.g., predetermined number of clusters, or if merging creates cluster with points too far apart.
- point-assignment: given some initial clusters, assign points to "best" cluster; might allow combining / splitting clusters, or unassign points. Example: k-means (our focus).

(Ullman et al:Mining Massive datasets)

Outline

1. Clustering
2. Vector spaces
   - Elkan´s Algorithm
3. Metric spaces
4. General Improvements
   - Swapping
   - Sampling
   - Initialization
   - Relation to LSH/DBH
5. Evaluation

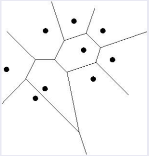
Vector spaces

Problem definition

- Clustering that minimizes objective function.
- k is given.
- Centroids do not have to be part of the dataset

k-means

- k-means is the most common problem: Main algorithms:
  -- Lloyd's algorithm is standard.
  -- Elkan's uses triangular inequality to accelerate updates.
- Also used to construct initial clusters for more sophisticated method.
- In Euclidean space, assignment is point location to the k Voronoi cells.



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/objects $\mathcal{Z}$ 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 $X = \{x_i\}$, $v(C) = (\mathrm{d}(x_1, C), \ldots, \mathrm{d}(x_n, C))$, where $C \subset \mathcal{Z}$ are the centroids, and the k-means objective is:

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

Similar 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 of each cluster, make it the 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

Properties

Lemma

For centers a, b, and point x: $d(x, a) \leq d(a, b)/2 \Rightarrow d(x, a) \leq d(x, b)$.

Proof. $x \in$ Ball centered at a with radius $d(a, b)/2$.

Corollary: x shall not be assigned to b.

Lemma (Triangular)

If center c is updated to c' then:

$|d(x, c) - d(c, c')| \leq d(x, c') \leq d(x, c) + d(c, c')$.

Maintain bounds

Initialisation

- Pick initial centers c and assign each x to closest center $c(x)$.
- Set $u(x) \leftarrow d(x, c(x))$, $l(x, c) \leftarrow d(x, c)$, $\forall c \neq c(x)$.
- Compute $d(c, c')$ for all centers $c \neq c'$.

Update bounds

Compute $d(c, c')$ for all centers $c \neq c'$.

- Upper bound $u(x)$ on distance of x to its center c;
  for new center c' of x's cluster: update $u(x) \leftarrow u(x) + d(c, c')$.
- Lower bound $l(x, c)$ to any other center $c \neq c(x)$;
  update when c changes to c': $l(x, c') \leftarrow |l(x, c) - d(c, c')|$.

These bounds follow from the triangular inequalities.

Elkan's Assignment

for each point x **do**
  if $u(x) \leq d(c(x), c')/2$, for all centers $c' \neq c(x)$ **then** Skip x; **continue**
  end if
  for each c' to which x is not assigned **do**
    if $u(x)$ not opt, compute $u(x) \leftarrow d(x, c(x))$; mark $u(x)$ as opt;
    if $u(x) \leq d(c, c')/2$, or $u(x) \leq l(x, c')$ **then** Skip c'; **continue**
    end if
    compute $d(x, c')$;
    if $d(x, c') < u(x)$ **then** assign x to c'; set $u(x) \leftarrow d(x, c')$
    end if
    update $l(x, c') \leftarrow d(x, c')$;
  end for
end for

Elkan's algorithm

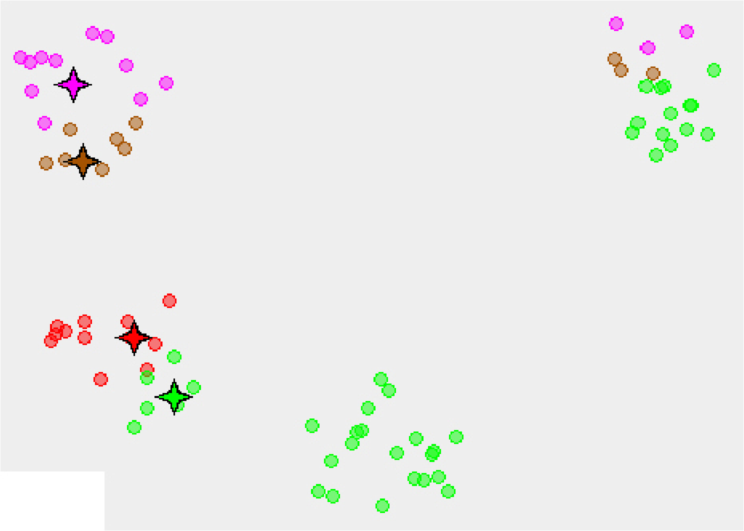
Elkan's algorithm offers, compared to Lloyd's,

-- large experimental speedup,

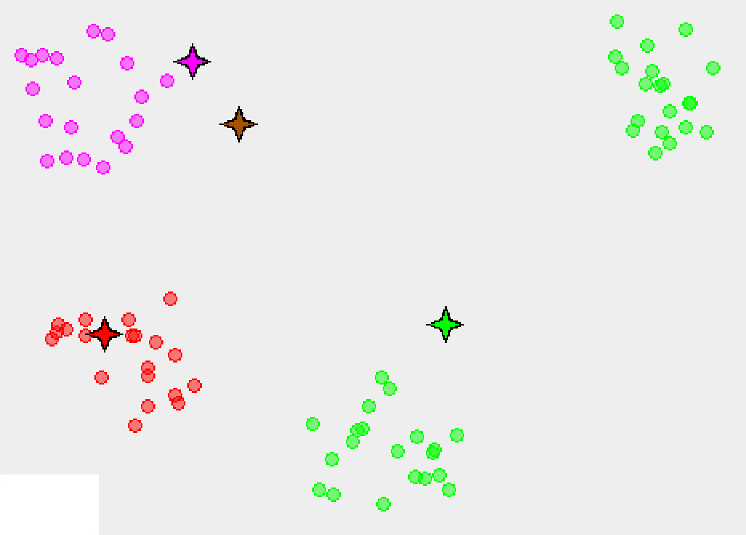
-- same output, same convergence,

-- requires much higher storage.

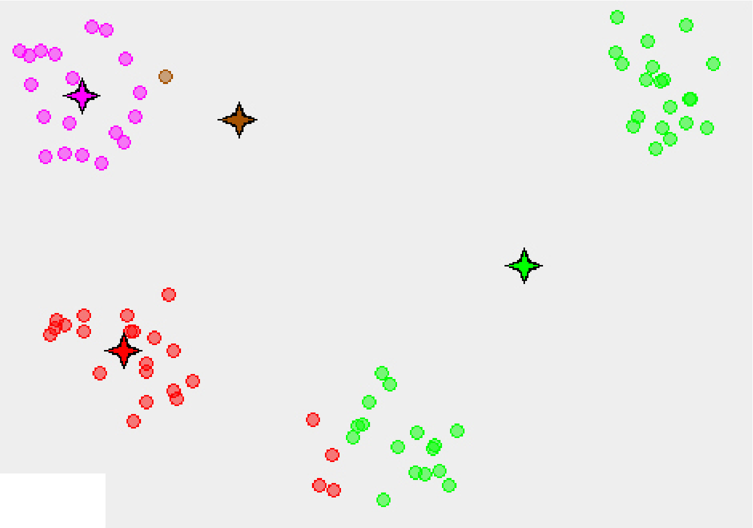
Initialization



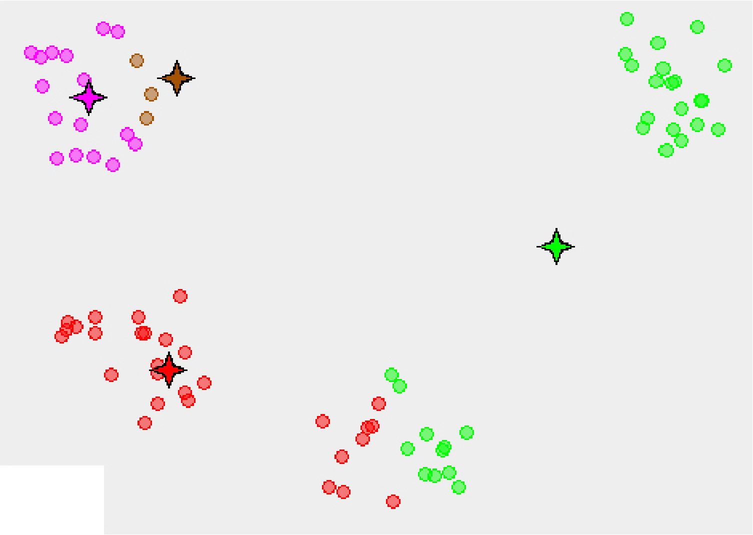
Lloyd's 1



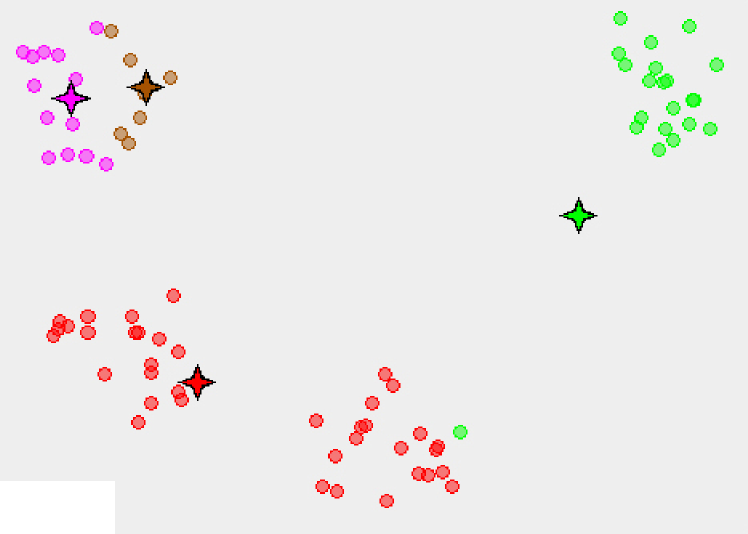
Lloyd's 2



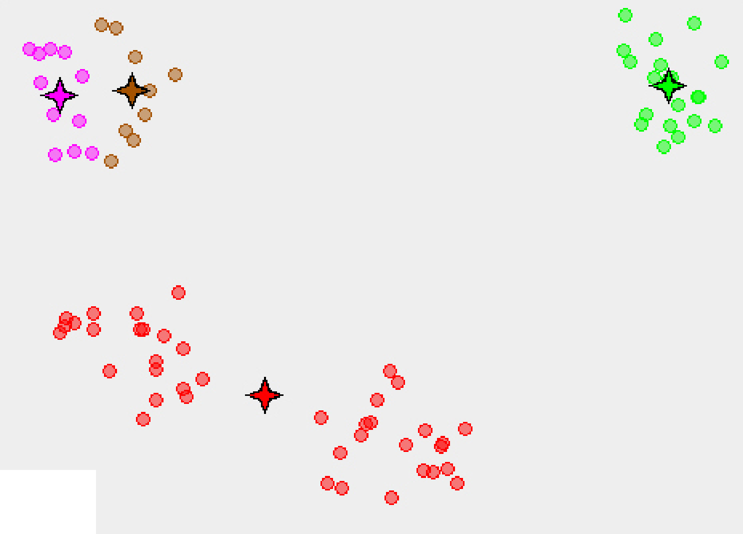
Lloyd's 3



Lloyd's 4



Final



Outline

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 value of objective function.

           Update the configuration if new objective value is lower.

       end for

   end for

Update until no new configuration selected.

Let distance calculation $= O(d')$. Update of Objective $= O((n-k)d')$ if 2nd best centroid known. Hence, update $= O((n-k)^2 kd') \sim n^2$.
(Small improve: test candidate swaps, implement best one.)

Update of Objective cost function

$$\text{Objective function: } J = \sum_{i=1}^{n-k} \text{dist}(i, c(i)).$$

Suppose for each item i we also store the 2nd best centroid c'.
If centroid m is replaced by non-centroid t, ΔJ takes 4 possible values:

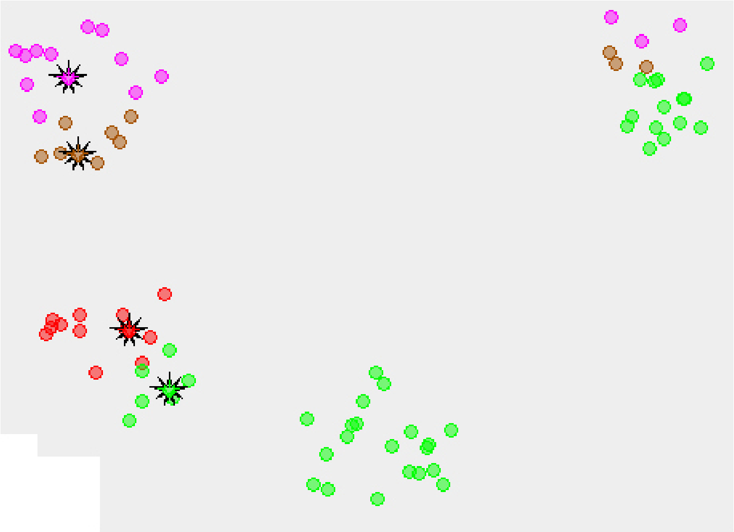
- For $i : c(i) = m$,

$$\Delta J = \begin{cases} \text{dist}(i,t) - \text{dist}(i,m), & \text{if dist}(i,t) \leq \text{dist}(i,c') : \text{ do nothing} \\ \text{dist}(i,c') - \text{dist}(i,m), & \text{if dist}(i,t) > \text{dist}(i,c') : \text{ assign } i \text{ to } c' \end{cases}$$

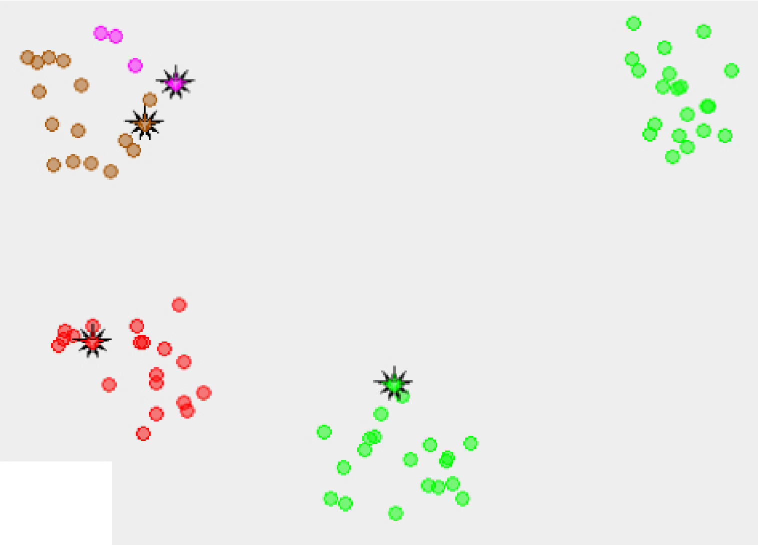
- For $i : c(i) \neq m$,

$$\Delta J = \begin{cases} 0, & \text{if dist}(i,t) \geq \text{dist}(i,c(i)) : \text{ do nothing} \\ \text{dist}(i,t) - \text{dist}(i,c(i)), & \text{if dist}(i,t) < \text{dist}(i,c(i)) : \text{ assign } i \text{ to } t \end{cases}$$

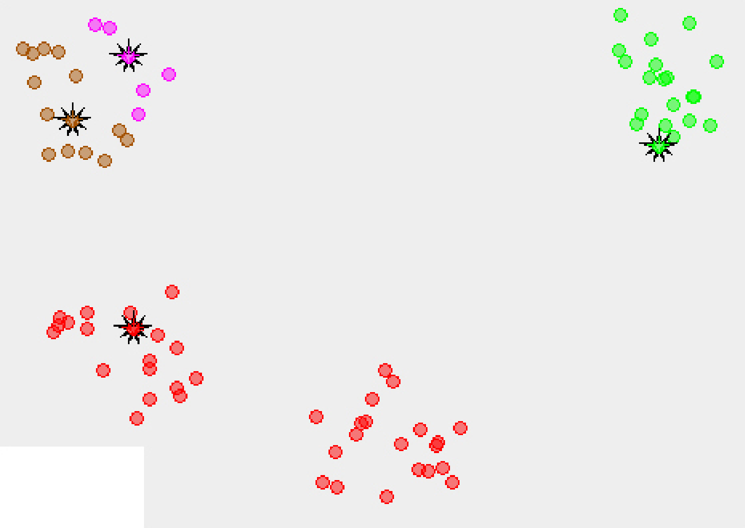
Initial



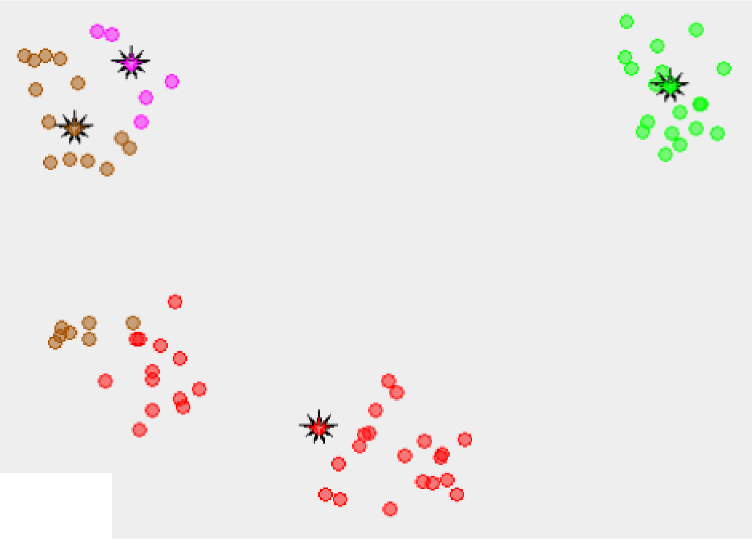
PAM 1



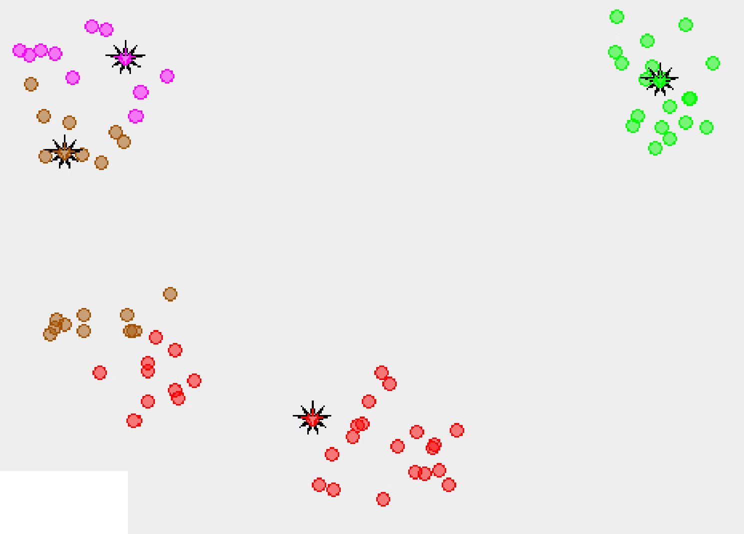
PAM 2



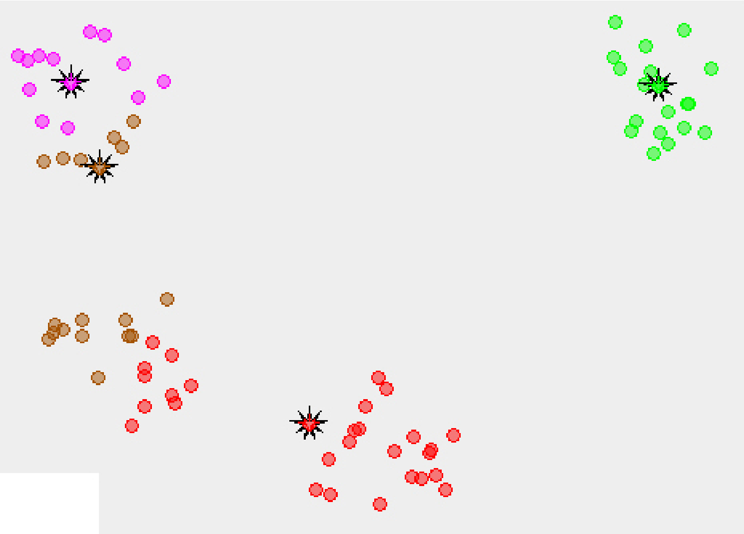
PAM 3



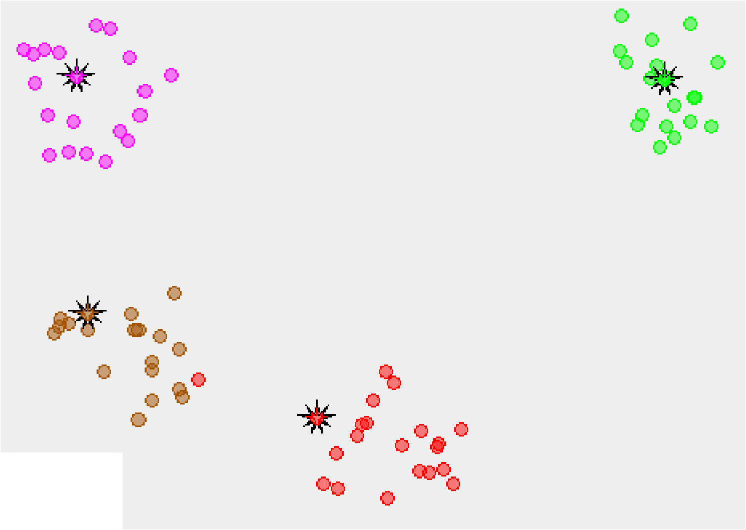
PAM 4



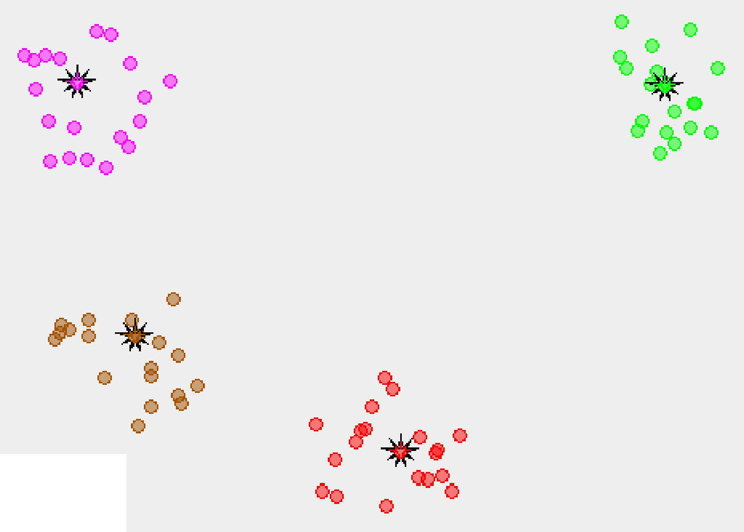
PAM 5



PAM 6



Final



Outline

1. Clustering
2. Vector spaces
   - Elkan´s Algorithm
3. Metric spaces
4. **General Improvements**
   - Swapping
   - Sampling
   - Initialization
   - Relation to LSH/DBH
5. Evaluation

Outline

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).

Outline

Clustering Large Applications (CLARA)

Idea: work with sample of size $n' < n$. Use s samples drawn independently.

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.
- Implement by picking random $Q \subset \{1,\ldots,k\} \times \{1,\ldots,n-k\}$.
- Also try s different, independent initializations.

for $i = 1,\ldots,s$ **do**

  Initialize k centroids (independently of previous initial sets).

  Randomly select set Q of $|Q|$ pairs (m,t), $|Q| \ll n(n-k)$.

  Run PAM swapping centroid m with non-centroid t, $\forall\,(m,t) \in Q$.

  Keep configuration with minimum objective value.

end for

Output clustering with minimum objective value over s candidates.

Experimental results recommend: $s = 2$, $|Q| = \max\{0.12 \cdot k(n-k), 250\}$.

(Ng-Han:VLDB'94) (Theodoridis-Koutroumbas:Pattern Recognition,ch.14)

Implement CLARANS

Consider a dataset of n points/objects with k clusters.
How to choose one uniformly distributed pair from

$$\{(m, t) : m = 0, \ldots, k-1,\ t = 0, \ldots, n-1\}.$$

Pick a uniformly distributed integer $x \in [0, kn-1]$ and return

$$\left(x \bmod k,\ \left\lfloor \frac{x}{k} \right\rfloor\right).$$

PAM vs CLARA / CLARANS

CLARA / CLARANS are approximations of PAM.

CLARANS is in $O(n^2)$, CLARA is significantly faster.

CLARANS leads to a better clustering, especially if CLARA's random sample is biased.

Both methods rely heavily on their parameters.

Outline

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

(1) Choose a centroid uniformly at random; $k \leftarrow 1$.
(2) $\forall$ point i, $D(i) \leftarrow$ min distance to centroid among k chosen centroids
(3) Choose new centroid: r chosen with probability proportional to $D(r)^2$:

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

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

Expected approximation ratio $= O(\log k)$ (Arthur-Vassilvitskii:SODA'07)

Implement initialization++

Given $D(i) > 0$, $i = 1, \ldots, n$, compute $n - k$ partial sums

$$P(r) = \sum_{j=1}^{r} D(j)^2, \qquad r = 1, \ldots, n-k,$$

and store them in an array P of n `float` numbers. To avoid the $P(r)$'s being very large, one can normalize all $D(i)$'s, e.g. by dividing them by $\max_i D(i)$.

Pick a uniformly distributed `float` $x \in [0, P(n)]$ and return

$$r \in \{1, 2, \ldots, n-k\} \; : \; P(r-1) < x \leq P(r),$$

where $P(0) = 0$: r chosen with probability proportional to $P(r) - P(r-1) \sim D(i)^2$. Can find r by binary search in array P.

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).

Outline

Assignment by ANN (LSH/DBH)

Direct approach

At each iteration:

1. Index k centroids into data-structure (e.g. LSH hashtables)
2. For every point, run ANN to find nearest centroid.

Assignment by ANN (LSH/DBH)

Reverse approach

- Index n points into L hashtables: once for entire algorithm.
- LSH/DBH TableSize $\leq n/8$: avoid buckets with very few items.
- At each iteration, for each centroid c, range/ball queries centered at c.
- Mark assigned points: either move them at end of LSH buckets (and insert "barrier"), or just flag them (using a new "flag" field).
- 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.
- If point in ≥ 2 ranges/balls compare its distances to respective centroids
- At end: for every unassigned point, compare its distances to all centroids

Outline

Internal evaluation

Evaluate clustering without reference to objective function. Try to capture meaning of clustering.

- Let k be the number of computed clusters.
- Internal evaluation considers the given pointset and the clusters, produces quality coefficient for each partition; k may be a parameter.
- External evaluation: use known class labels and benchmarks; often created by humans.

In the sequel we present internal evaluation methods, mainly Silhouette.

Silhouette

For $1 \leq i \leq n$, $a(i) =$ average distance of i to other objects in same cluster.
Let $b(i) =$ average distance of i to objects in *next best* (neighbor) cluster.

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 $i = 1, \ldots, n$.

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