Is task clustering executable

Hong Lu
luh.lewis@gmail.com

1 Reviews

How to deal with the tasks?

We can adapt several representations to the tasks by using various algorithms based on different data structures. The idea, we often bring about easily, is to connect or divide the tasks into an entity or several groups.

k-means

Within clustering algorithm, k-means is efficient and gurantees to converge.

It depends that what constitutes good clusters subjecting to various criteria, both ad-hoc or systematic.

Some experiments are shown below in Fig. 1

simple k-means is shown as algorithm 1

Yinyang k-means

Yinyang k-means, proposed @ICML in 2015, optimizes *Assignment* process of algorithm 1 mainly. It uses Triangle inequality to reduce computations which has been implemented before. Let's introduce some notations, $c_i \in C,\ i = 1, 2, 3, ..., k$, where C represents set of cluster centers and c is a cluster within the set. Let $b(x)$ ("the best cluster for") be the cluster to which the point is assigned to. Let $b'(x), c', C'$ be the entity in the next iteration. Let $\sigma(c)$ represents $d(c, c')$

Global filtering identifies whether a point x changes its cluster in an assignment step with a **single** comparison. Algorithm maintains an upper and lower bound, where $ub(x) \geq d(x, b(x))$ and $lb(x) \leq d(x, c), \forall c \in C - b(x)$

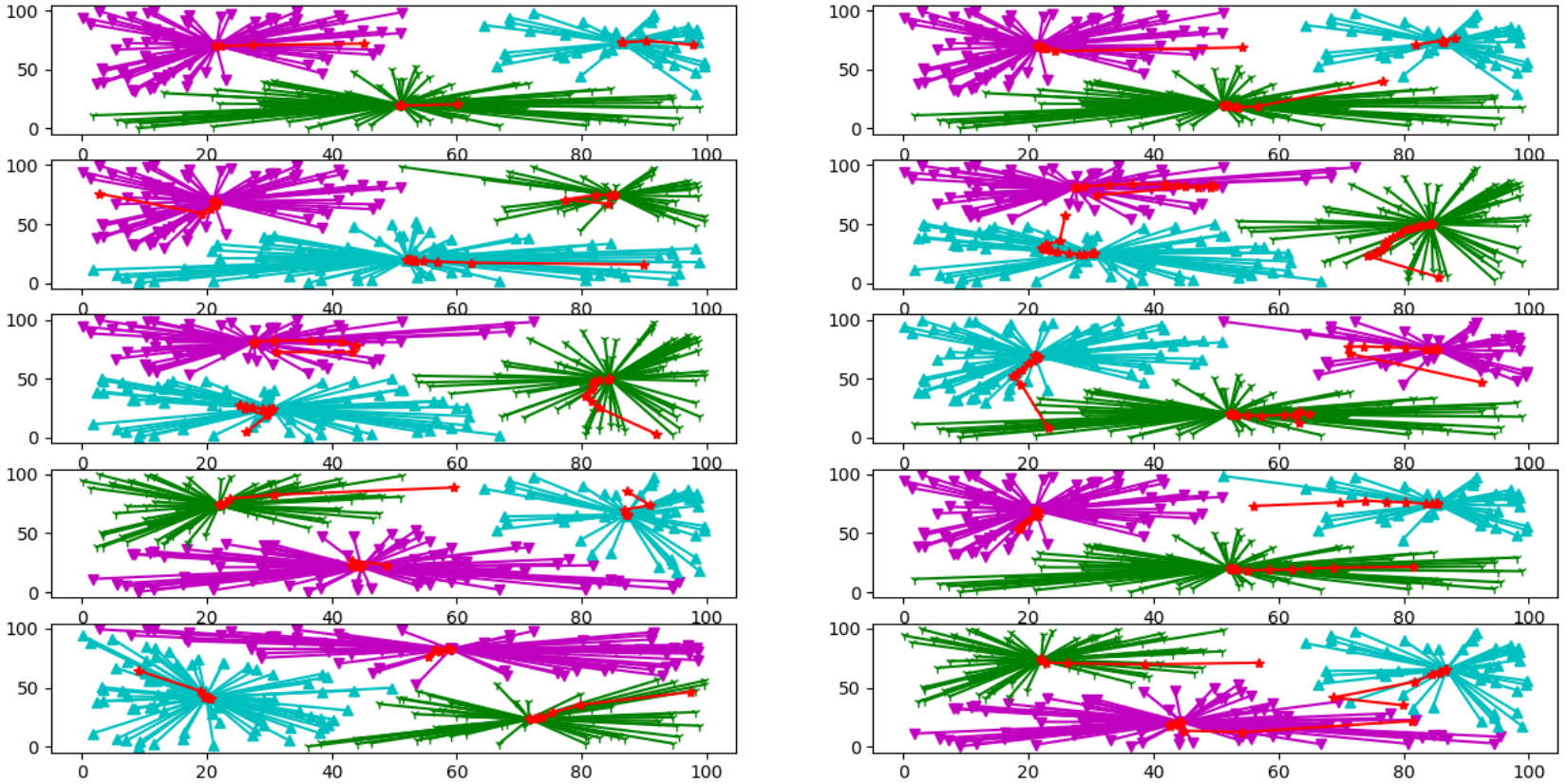

Figure 1: random initial centroid

Algorithm 1: Jejune K-means: Centroid-based k-means

Data: k, Points, initial centroids
Result: Clusters: C
begin
  Initialization
  begin
    centroids ← initial centroids
    C, result = **Assginment**(*points*, *centroids*)
  end
  while *result not converges* **do**
    centroids = **Update Centroid**(C)
    C, result = **Assginment**(*points*, *centroids*)
    if *Assginment not change* **then**
      result converges
    end
  end
  return C
end

Update Centroid(Clusters)
begin
  for C_i *in Clusters* **do**
    for p_j *in* C_i **do**
      $centroid_i = \frac{1}{M} \sum p_j$
    end
  end
  return *centroids*
end

Assignment(points,centroids)
begin
  for p *in Points* **do**
    Calculate Distance between p and $centroids_i,\ i = 1, 2, 3, ..., k$
    Store Nearest centroid c^* of p
  end
  assign p_i to respective c_i^* in C **if** *assignment is stable* **then**
    return *C, converge*
  end
  return *C, not converge*
end