1 Basic Concepts about Clustering

Let d be a positive integer and $\mathbb{R}$ the field of real numbers. For a set S of n points $\vec{p_i} \in \mathbb{R}^d$, we denote by $|S|$ the number of points of S. We consider the problem that we will call "k-means globally optimum clustering".

Definition 1. *The "k-means globally optimum clustering" is to split $S \subset \mathbb{R}^d$ of n points $\vec{p_i}$, $i = 1, \ldots, n$ into k disjoint nonempty subsets $S_1, \ldots, S_k$ called clusters in such a way that the following expression is minimized:*

$$f_{S_1,\ldots,S_k}(S) = \sum_{j=1}^{k} \sum_{\vec{p} \in S_j} \|\vec{p} - \vec{q_j}\|^2, \quad \textit{where } \vec{q_j} = \frac{\sum_{\vec{p} \in S_j} \vec{p}}{|S_j|}.$$

$S_1, \ldots, S_k$ is called an optimal partition of S.

It is well known that, given S, there always exists $\vec{q_1}, \ldots, \vec{q_k}$ such that the partition defined as,

$$S_j = \bigcap_{l=1}^{k} \{\vec{p} \in S \ : \ \|\vec{p} - \vec{q_j}\|^2 \leq \|\vec{p} - \vec{q_l}\|^2\},$$

is an optimal partition.[1] Indeed, the common approach to attack this problem is to use *Lloyd's heuristic* [6], which was first used in [7] and, under minor modifications, performs quite well in practice, see [1, 10].

We will need the following concepts from topology:

- A set contained in $\mathbb{R}^d$ is *convex* if for any pair of points within the set, every point in the straight line segment that joins them is also within the object.
- Given a set of points $S \subset \mathbb{R}^d$, the convex hull of S is the smallest set of $\mathbb{R}^d$ which contains S.
- Given $\vec{a} \in \mathbb{R}^d - \{\vec{0}\}$ and $b \in \mathbb{R}$, the set $\mathcal{H} = \{\vec{x} \in \mathbb{R}^d : (\vec{a})^{\mathbf{T}}\vec{x} = b\}$ is called a hyperplane.
- A point $\vec{p} \in \mathbb{R}^d$ lays in the *left side* of hyperplane $\mathcal{H}$ if $(\vec{a})^{\mathbf{T}}\vec{p} > b$. If $(\vec{a})^{\mathbf{T}}\vec{p} < b$, the point $\vec{p}$ lays in the *right side* of hyperplane $\mathcal{H}$.
- An hyperplane $\mathcal{H}$ *separates* two sets $S,\ S' \subset \mathbb{R}^d$ if all the points in S lays in the left side of $\mathcal{H}$ and all the points in S' lays in the right side of $\mathcal{H}$.
- An hyperplane $\mathcal{H}$ *splits* a set S if any point $\vec{p} \in S$ lays in the right or the left side of $\mathcal{H}$.

We cite here the separating hyperplane theorem, see [3, page 46] for a proof.

[1] Using this definition it could be that one point belong to more than one clusters. Fortunately, it is always possible to solve the ties in a reasonable manner

Lemma 1. *For any two convex sets $S,\ S' \subset \mathbb{R}^d$ such that $S \cap S' = \emptyset$, there exists an hyperplane $\mathcal{H}$ that separates S and S'.*

As it was stated before, it is known that one optimal partition is defined using k centroids. Partitions defined by centroid have a very interesting property.

Lemma 2. *Given a set of point $S \subset \mathbb{R}^d$ and centroids $\vec{q_1}, \ldots, \vec{q_k} \in \mathbb{R}^d$, the partition $S_1, \ldots, S_k$ defined as*

$$S_j = \bigcap_{l=1}^{k} \{\vec{p} \in S \ :\ \|\vec{p} - \vec{q_j}\|^2 \leq \|\vec{p} - \vec{q_l}\|^2\},$$

for $j = 1, \ldots, k$ satisfies:

- *the intersection of the convex hull of any two different clusters S_i, S_j is empty,*
- *for each pair S_i, S_j exists an hyperplane $\mathcal{H}$ that separates S_i and S_j.*

Proof. The first assertion of the lemma is proved by induction. For $k = 2$, it is trivial. The general case is done noting that the intersection of two convex sets is a convex set. So, the convex hull of

$$S_j = \bigcap_{l=1}^{k} \{\vec{p} \in S \ :\ \|\vec{p} - \vec{q_j}\|^2 \leq \|\vec{p} - \vec{q_l}\|^2\},$$

is just the intersection of the convex hulls of

$$\{\vec{p} \in S \ :\ \|\vec{p} - \vec{q_j}\|^2 \leq \|\vec{p} - \vec{q_l}\|^2\},$$

for $l \neq j$, which are disjoint by induction.

The second assertion is a direct application of Lemma 1 and that S_i, S_j are convex sets. $\square$

2 Reverse Enumeration

Reverse Enumeration is a method for enumerating element in a set. It was introduced in [2] which solves the following problem,

Problem 1. *Suppose that $G = (V, E)$ be an undirected graph, where V is a set of vertex and E is the edge set. Enumerate all the elements in V.*

The difficulty of this particular problem lays in the fact that V is not given explicitly, however given a node $v \in V$ it is possible to calculate its neighbors.

The problem of graph traversal is well-known, and there are well-known algorithms like breadth-first search and depth-first search, see [4, Page 597].

Unfortunately, these algorithms needs to mantain a data structure with all the nodes that have been visited in order to avoid looping endlessly because of cycles in the graph. This implies a big drawback.

For introducing a more efficient way of solving Problem 1 we need to introduce the definition of *local search*.

Definition 2. *A local search* (G, R, f) *is a triple satisfying* $R \subset V$*,* f *is a mapping* $f : V \mapsto V$ *with the following properties:*

- $(v, f(v))$ *is an edge of the graph for* $v \in V - R$*,*
- *for all* $v \in V - R$*, there exists a positive integer* t *such that* $f^t(v) \in R$*.*

The function f *is said to be the local search function and* G *the underlaying graph structure.*

Informally a local search algorithm is a way of explore a graph in a non systematic way, starting in any candidate and heading for the set of solutions R, see [8, Page 110] for a more detailed exposition.

A local search define a subgraph of G called the *trace* $T = (V, E(f))$ where,

$$E(f) = \{(v, f(v) \ : \ v \in V - R\}.$$

An important fact that appear in [2, Property 2.1] is that the trace of any local search contains all the nodes of G and each component contains only one element of the set R and no cycles.

So, most efficient way to output all the component is to traverse all the components of T, starting at every element of R. However, for a local search function f is normally difficult to find the preimages of a node v. In most of the cases, there is an *adjacency oracle* which gives the neighbours of a node. The original definition can be found in [2], but here we simplify it for our purposes.

Definition 3. *A graph* G *is given by an adjacency oracle if there exists a function* Adj *that takes a node* v *an return an ordered list of neighbors of* v*.*

So, given a graph G given by an adjacency oracle Adj and a local search (G, R, f), Algorithm 1 will output all the nodes. The complexity of the algorithm is given in [2, Theorem 2.2], which we enunciate here.

Theorem 1. *Let* (G, R, f) *be a local search where* G *is given by a adjacency oracle* Adj*. Suppose that* $Adj(v)$ *does not contain more than* δ *nodes for any* $v \in V$*. Then, if* $t(f)$ *and* $t(Adj)$ *are the times complexity for* f *and* Adj*, respectively, the time complexity of reverse search is of order of magnitude,*

$$\delta t(Adj)|V| + t(f)|E|.$$

2.1 Reverse search for two clustering problem

For the two clustering problem, we are going to adapt the scheme presented in the previous section. For that, we need to define the following objects:

- the graph $G = (V, E)$,
- a local search (G, R, f),
- an adjacency oracle Adj.

Procedure 1 General Reverse Search

Input: An adjacency oracle Adj, a local search (G, R, f)
Output: all nodes in V without any repetition

```
for all r ∈ R do
  yield r
  i, do, v = 1, True, r
  while do do
    while there are at least i elements in Adj(v) do
      let Next be the ith element of Adj(v)
      if f(Next) == v then
        v, i = Next, 0
      else
        i = i + 1
      end if
    end while
    if v == r then
      do = False
    else
      u, v, i = v, f(v), 1
      while ith element of Adj(v) is not u do
        i = i + 1
      end while
    end if
  end while
end for
```

V is a complete description of all possible ways to split S by hyperplanes. The set V is going to be a set of hyperplanes satisfying two properties:

- each of the hyperplanes splits the set S in a different way.
- If S_1 and S_2 is a partition of S and S_1 and S_2 are convex, then there exists an hyperplane $\mathcal{H} \in V$ which separates S_1 and S_2.

In other words, V is a set of representatives of all possible two-clusters.

Now, we define the set E implicitly by defining when two nodes are neighbors. Two hyperplanes $\mathcal{H}, \mathcal{H}' \in V$ are neighbours if and only if the points of S that lay in the left side of $\mathcal{H}$ are exactly the points that lay in the left side of $\mathcal{H}$, except for one point in S. Informally, two nodes are neighbors if the partitions they defined are the same except from one point.

2.1.1 Local search

The local search is an important part in the efficiency of the algorithm. For the rest of the paper, we suppose that points in S are ordered, so $S = \{\vec{p}_1, \ldots, \vec{p}_n\}$. One possible local search function is given in Algorithm 2. The set $R = \{\vec{r}\}$

Procedure 2 naive local search function f

Input: an hyperplane $\mathcal{H}$

```
Let S_1 and S_2 the partition defined by H
for all p in S_2 do
  if There exists an hyperplane H' that separate S_1 ∪ {p} and S_2 − {p} then
    return H'
  end if
end for
return H
```

contains only an hyperplane $\vec{r}$ with the property that all the points lay in the left side of this hyperplane.

The problem with this proposal is that it is necessary to find an hyperplanes that separates two sets of points and this can be quite expensive, see [5]. We adapt the proposal for the local search (G, R, f) in [9] because it is more efficient and only requires to find a separating hyperplane. Unlike Algorithm 2, this local search function depends on $\vec{r}$. Although in a cryptic way, this function takes the point in the left side of $\mathcal{H}$ that is the closest to hyperplane $\mathcal{H}$ and then search for an hyperplane that separates the points and move it to the

References

[1] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

Procedure 3 local search function f

Input: an hyperplane $\mathcal{H}$ and $\vec{r}$

$\vec{p}_{min} = \vec{p}_1$
for all $\vec{p_k}$ in S **do**
  if $\vec{p}_k{=}{=}MinDistance(\vec{p}_{min}, \vec{p}_k, \mathcal{H}, \vec{r})$ **then**
    $\vec{p}_{min} = \vec{p}_1$
  end if
end for
Let S_1 be the points laying at the left side of $\mathcal{H}$
Let S_2 be the points laying at the right side of $\mathcal{H}$
return An hyperplane separating $S_1 \cup \{\vec{p}_{min}\}$ and $S_2 - \{\vec{p}_{min}\}$

Procedure 4 Algorithm MinDistance

Input: Two hyperplanes h_1, h_2 and two different points $\vec{p}, \vec{r}$
Output: the hyperplane which intersects the segment closest to $\vec{p}$

$$c_1 = \frac{-h_1 \cdot \vec{p}}{h_1 \cdot (\vec{r} - \vec{p})}, \quad c_2 = \frac{-h_2 \cdot \vec{p}}{h_2 \cdot (\vec{r} - \vec{p})}$$

if $c_1 > c_2$ **then**
  return h_2
else if $c_1 < c_2$ **then**
  return h_1
else
  returns the minimum of $-h_1/h_1 \cdot \vec{p}$ and $-h_2/h_2 \cdot \vec{p}$ by the lexicographic order.
end if

[2] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1993.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[5] David A. Elizondo. The linear separability problem: some testing methods. *IEEE Transactions on Neural Networks*, 17(2):330–344, 2006.

[6] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[7] James B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[8] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[9] Nora H. Sleumer. Hyperplane arrangements: Construction, visualization and applications. Master's thesis, Swiss Federal Institute of Technology, 2000.

[10] Chen Zhang and Shixiong Xia. K-means clustering algorithm with improved initial center. In *Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on*, pages 790 –792, jan. 2009.