Global Optimality in k-means Clustering and Discretization *

Domingo Gómez, Cristina Tîrnăucă, José L Balcázar, José L Montaña, and Emilio Castillo Villar

Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria
Santander, Spain
{gomezd,cristina.tirnauca,montanjl,emilio.castillo}@unican.es
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Spain
jose.luis.balcazar@upc.edu

Abstract. The most popular clustering algorithm in practice is, most likely, Lloyd's heuristic approximation algorithm to the k-means optimum centroids. Instead, we study here the problem of finding the globally optimum set of centroids, a problem known to be NP-hard. Existing literature contains an algorithm that is stated to obtain this optimum in the "fixed-parameter tractability" setting; the published validations are incomplete, and we have not found any reference to this algorithm having been actually implemented. We provide validations, and refine the analysis to show better bounds; we identify alternative algorithms that turn out to be better for relevant particular cases; and we study variants in terms of parallelization and randomization. Whereas these algorithms may often be too slow to be of widespread application in data mining practice, they will allow us to identify absolutely optimum solutions for benchmark problems, whereby alternative heuristic proposals can evaluate the goodness of their solutions and the precise price paid for their faster running times.

1 Introduction

Assume we are given a dataset S containing n observations, for a fixed dimensionality d. Assume also that an integer value k is given. We consider the problem that we will call "k-means globally optimum clustering": find k centroids $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ that minimize the squared error committed by replacing each point $\boldsymbol{p}_i$ by its closest centroid.

* This work has been partially supported by projects FORMALISM (TIN2007-66523) and BASMATI (TIN2011-27479-C04) of Programa Nacional de Investigación, Ministerio de Ciencia e Innovación (MICINN), Spain, and by the Pascal-2 Network of the European Union.

An equivalent way of stating the problem is the following: S is to be partitioned into k disjoint subsets S_j called *clusters*, in such a way that the following expression is minimized:

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

The algorithm usually known as *k-means*, which was first used in [14] and has became the standard name in practice (for instance, in many Data Mining software tools), is also called *Lloyd's heuristic* [13]. Lloyd's ubiquitous heuristic consists of selecting k initial centroids $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ according to some criterion (for example, randomly), constructing S_j as the set of points $\boldsymbol{p}$ that are closer to $\boldsymbol{q}_j$ than to any other centroid (ties can be broken arbitrarily), recomputing the centroids as mass centers of the current S_j, and iterating until stability of the clusters. Interesting studies of the goodness of its solution, mostly in terms of the initialization criterion, are [2, 19].

This is a very popular algorithm: often, we have far more observations than a human user can look at and make sense of. "Abstracting" them out into a handful of "representative" centroids, maybe with an indication of the cardinalities of the clusterings, is a way of understanding a bit of the inherent structure of the dataset. Note that, for this process to actually make sense, we wish a smallish k; however, little is known about how to select it (see [16] for one successful approach).

Most of the practical implementations of k-means offer no indication of the quality of the locally optimal solution found. How difficult is it to come up with the actual global solution which minimizes to optimality the expression above? A simple approach working on partial clusterings of part of the data points, where single points are added sequentially, allows for application of A* schemes but leads to algorithms that run in exponential time on the number of points and require exponential space. It is known that the problem is NP-hard, see [1, 8]. Even restricting $k = 2$ (see [6]) or $d = 2$ (see [15]) are NP-hard problems.

However, an interesting alternative was proposed in [12]: plainly enumerating *all*[1] possible clusterings, evaluating them, and keeping the best seen so far is possible in time $\mathcal{O}(n^{dk+1})$, that is, the problem falls in the area known as "fixed-parameter tractability" [7], whereby exponentially difficult problems are reduced to polynomial upon fixing specific parameters (in our case, dimension and number of clusters). Admittedly, the polynomial time algorithms might need, for cases of interest, high degree.

A related problem is that of discretization. There, we consider each attribute (or: coordinate) of the points in the dataset separately, by projection on each axis. Then, we wish to divide the projected points into a small number of intervals. Here, the usage is, often, as preprocessing towards the usage of another modeling tool which works on categorical attributes instead of float-valued ones. For

[1] the algorithm does not actually enumerate ALL of them, but it does enumerate many of them, and most importantly, the optimal clustering is proven to be there

instance, even though Gaussian-based variants of the Naïve Bayes predictor work on floats, it has been argued [18] that better results are obtained if discretization is applied instead and the predictor is used on the intervals, considered as categorical attributes. The usual discretization methods include fixed-width bins, balanced cardinality bins, entropy minimization [10], fixed k-intervals, or proportional k-intervals [18]. However, it is natural to consider k-means-based intervals as a candidate method to choose the bins, except that the dependence of the heuristic on the initial choice leads to different discretizations for different runs, which may seem odd. The obvious, but admittedly expensive, solution we study here is to choose the uniquely defined globally optimum k-means discretization. The interest of this problem is that, since, in this case, $d = 1$, if we accept a limited number of discretization intervals (which is useful anyhow in practice, for interpretability's sake) we may end up having polynomial time algorithms of low degree that could find globally optimum k-means solutions in affordable running times.

2 Preliminaries

Let us first present the approach in [12]. Keep in mind that our objective is to find *all* possible partitions $S_1, \ldots, S_k$ of S, and select the one that minimizes the squared error $f_{S_1,\ldots,S_k}(S)$.

But instead of finding all partitions of S into k clusters, we will be searching for sets of k centroids, $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k\}$ in $\mathbb{R}^d$, and we will define a unique way of associating a partition to a given set of centroids. More precisely, given $S = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, we say that $S_1, \ldots, S_k$ is *the Voronoi partition defined by* $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k) \in \mathbb{R}^{dk}$ if :

$S_1 = \{\boldsymbol{p} \in S \mid \|\boldsymbol{p} - \boldsymbol{q}_1\|^2 \leq \|\boldsymbol{p} - \boldsymbol{q}_j\|^2, \forall j \in \{2, \ldots, k\}\}$

$S_2 = \{\boldsymbol{p} \in S \mid \|\boldsymbol{p} - \boldsymbol{q}_2\|^2 \leq \|\boldsymbol{p} - \boldsymbol{q}_j\|^2, \forall j \in \{3, \ldots, k\}, \|\boldsymbol{p} - \boldsymbol{q}_2\|^2 < \|\boldsymbol{p} - \boldsymbol{q}_1\|^2\}$

$\vdots$

$S_i = \{\boldsymbol{p} \in S \mid \|\boldsymbol{p} - \boldsymbol{q}_i\|^2 \leq \|\boldsymbol{p} - \boldsymbol{q}_j\|^2, \forall j \in \{i+1, \ldots, k\}, \|\boldsymbol{p} - \boldsymbol{q}_i\|^2 < \|\boldsymbol{p} - \boldsymbol{q}_j\|^2, \forall j \in \{1, \ldots, i-1\}\}$

$\vdots$

$S_k = \{\boldsymbol{p} \in S \mid \|\boldsymbol{p} - \boldsymbol{q}_k\|^2 < \|\boldsymbol{p} - \boldsymbol{q}_j\|^2, \forall j \in \{1, \ldots, k-1\}\}$.

Note that there may be cases in which a point $\boldsymbol{p}$ has at least two centroids $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ at equally minimum distance. Including it in either S_i or S_j (but not in both!) would make no difference (the sum of squared errors is the same for both cases), but, for algorithmic purposes, we need to have a fixed way of placing it (we choose to include it in the partition set of smaller index).

It is easy to check that $\|\boldsymbol{p} - \boldsymbol{q}_i\|^2 \leq \|\boldsymbol{p} - \boldsymbol{q}_j\|^2$ is equivalent to $\|\boldsymbol{q}_i\|^2 - \|\boldsymbol{q}_j\|^2 - 2\boldsymbol{p} \cdot (\boldsymbol{q}_i - \boldsymbol{q}_j) \leq 0$, where $\cdot$ denotes the standard dot product. Therefore, if $\mathbf{P} = (P_{\boldsymbol{p}(ij)})_{\boldsymbol{p} \in S, 1 \leq i < j \leq k}$ is a family of polynomials in $\mathbb{R}[X_1, \ldots, X_{dk}]$ defined

by

$$P_{\boldsymbol{p}(ij)} = \sum_{r=1}^{d} \left(X_{r+(i-1)d}^2 - X_{r+(j-1)d}^2 \right) - 2 \sum_{r=1}^{d} p_r (X_{r+(i-1)d} - X_{r+(j-1)d}), \quad (1)$$

where $\boldsymbol{p} = (p_1, \ldots, p_d)$, then $\boldsymbol{p} \in S_i$ if and only if it holds that for all $j > i$, $P_{\boldsymbol{p}(ij)}(\boldsymbol{q}) \leq 0$ and for all $j < i$, $P_{\boldsymbol{p}(ji)}(\boldsymbol{q}) > 0$.

Therefore, given a point $\boldsymbol{q}$ in $\mathbb{R}^{dk}$, in order to see which is the Voronoi partition it defines, it is enough to check the sign of a family of l polynomials, where $l = \frac{nk(k-1)}{2}$. Of course, going from *enumerating all possible partitions* (admittedly many, yet countable) to *enumerating all possible points in* $\mathbb{R}^{dk}$ (not countable) does not seem much of an improvement. Moreover, it is not clear that this way we obtain *all* partitions. Actually, we do not obtain all of them, but as we shall later see, we do obtain the optimal one.

On the other hand, we do not need to enumerate all points in $\mathbb{R}^{dk}$. Indeed, given $\mathbf{P}$, we can define an equivalence relation $\equiv_{\mathbf{P}}$ on $\mathbb{R}^{dk}$ as follows: $\boldsymbol{q} \equiv_{\mathbf{P}} \boldsymbol{q}'$ if and only if $P(\boldsymbol{q})$ and $P(\boldsymbol{q}')$ have the same sign for all P in $\mathbf{P}$. Then our problem reduces to *enumerating all possible equivalence classes* (again, a big yet countable number). And for that we can use existing standard techniques, which we describe above.

2.1 Cell Arrangements

For a fixed polynomial P in $\mathbb{R}[X_1, \ldots, X_m]$, the subset $H = \{\boldsymbol{x} \in \mathbb{R}^m \mid P(\boldsymbol{x}) = 0\}$ of $\mathbb{R}^m$ is called a *real algebraic variety*. Each point $\boldsymbol{p} \in \mathbb{R}^m$ belongs to either H, $H^+ = \{\boldsymbol{x} \in \mathbb{R}^m \mid P(\boldsymbol{x}) > 0\}$ or $H^- = \{\boldsymbol{x} \in \mathbb{R}^m \mid P(\boldsymbol{x}) < 0\}$.

Note that H divides $\mathbb{R}^m$ into two components, H^+ and H^-. Each of these sets can further be subdivided by a different algebraic variety, say $H' = \{x \in \mathbb{R}^m \mid P'(x) = 0\}$, for some P' in $\mathbb{R}[X_1, \ldots, X_m]$.

More generally, a set of algebraic varieties $\mathcal{H} = \{H_1, \ldots, H_l\}$ in $\mathbb{R}^m$ partitions the space into separate pieces called *cells*, each of which is a connected region in $\mathbb{R}^m$. This partition is called a *cell arrangement*. It is easy to see that there is a one-to-one correspondence between these cells and the equivalence classes previously defined.

Furthermore, we define a function $sv : \mathbb{R}^m \to \{-1, 0, +1\}^l$ such that for any point $\boldsymbol{p} \in \mathbb{R}^m$, $sv(\boldsymbol{p})$ is an l-dimensional vector, called the *sign vector*, with the property that its i-th component is given by

$$sv(\boldsymbol{p})_i = \begin{cases} +1 & \text{if } \boldsymbol{p} \in H_i^+, \\ -1 & \text{if } \boldsymbol{p} \in H_i^-, \\ 0 & \text{if } \boldsymbol{p} \in H_i. \end{cases}$$

Obviously, all points in a cell arrangement (or equivalently, all points in the same equivalence class) have the same sign vector. Going back to our problem, given a cell arrangement $\mathcal{H} = \{H_1, \ldots, H_l\}$, we are interested in enumerating all

possible sign vectors. We say that an l-dimensional vector $\boldsymbol{e}$ in $\{-1, 0, +1\}^l$ is *feasible* with respect to $\mathcal{H}$ if $\boldsymbol{e} \in sv(\mathbb{R}^m)$. Moreover, if we denote by $V_{CELL}(\mathcal{H})$ the set of all feasible vectors, then a trivial upper bound for the cardinality of $V_{CELL}(\mathcal{H})$ is 3^l. If $l > m$ and all the polynomials in $\mathcal{P}$ have degree at most deg, the cardinality of $V_{CELL}(\mathcal{H})$ is $l^m\mathcal{O}(deg/m)^m$ (see Corollary 2 on page 12 of [3]). Moreover, if $deg = 1$, $|V_{CELL}(\mathcal{H})|$ is at most $\sum_{i=0}^{m} \binom{l}{m-i}$ (see [9, p. 8] for details).

2.2 Cell Enumeration

Let $P_1, \ldots, P_l$ be multivariate polynomials and let

$$\mathcal{H} = \{H_1, \ldots, H_l\}, \quad H_i = \{\boldsymbol{x} \in \mathbb{R}^m \mid P_i(\boldsymbol{x}) = 0\}.$$

We are going to focus on the case in which $l > m$, since if $l \leq m$, one may have, in the worst case, 3^l feasible sign vectors (consider $P_1, \ldots, P_l$ in $\mathbb{R}[X_1, \ldots, X_m]$ defined by $P_i = X_i$), so outputting all vectors in $\{-1, 0, 1\}^l$ might be necessary.

The following result appears in a slightly modified form in [3]. Nevertheless, we choose to include an alternative proof for this particular formulation.

Proposition 1 ([3]) *Let $(e_1, \ldots, e_l)$ be a vector in $\{-1, -0, 1\}$ and $\mathcal{P} = P_1, \ldots, P_l$ a family of polynomials in $\mathbb{R}[X_1, \ldots, X_m]$. Consider the following system of equations:*

$$\begin{cases} sign(P_1) = e_1, \\ sign(P_2) = e_2, \\ \ldots \\ sign(P_l) = e_l, \end{cases} \tag{2}$$

Let V be a non-empty connected component of solutions of (2). Then, for sufficiently small $\varepsilon > 0$, there exists $I \subseteq \{1, \ldots, l\}$ such that (3) has solutions and at least one connected component of solutions of (3) is included in V.

$$P_i = e_i\varepsilon, \forall i \in I \tag{3}$$

Proof. Let V_0 be a non-empty connected component of solutions of (2). We show that there exists a non-empty connected component of solutions of (3) included in V_0 for some $I \subseteq \{1, \ldots, l\}$.

For that, assume without loss of generality that $e_1 = e_2 = \ldots = e_s = 0$ and $e_{s+1} = \ldots = e_l = 1$ for some $s \in \{0, \ldots l\}$. System (2) can therefore be written

equivalently:

$$\begin{cases} P_1 = 0, \\ \cdots \\ P_s = 0, \\ P_{s+1} > 0, \\ \cdots \\ P_l > 0, \end{cases} \tag{4}$$

Consider now the following system of equations

$$\begin{cases} P_1 = 0, \\ \cdots \\ P_s = 0, \end{cases} \tag{5}$$

Clearly, there exists W_0 a non-empty connected component of solutions of (5) such that $W_0 \supseteq V_0$. If $W_0 = V_0$ then we are done, since we can take $I = \{1, \ldots, s\}$.

Otherwise, let $x_1 \in W_0 \backslash V_0$ and take $x_0 \in V_0$ arbitrary. Moreover, consider a path function $f : [0,1] \to \mathbb{R}^m$ in W_0 such that $f(0) = x_0$ and $f(1) = x_1$. Also, let $t \in [0,1]$ be minimal such that $f(t) \notin V_0$. We show that there exists $i_1 \in \{s+1, \ldots, l\}$ such that $P_{i_1}(f(t)) = 0$.

Assume by contradiction that for all $i \in \{s+1, \ldots, l\}$ we have $P_i(f(t)) \neq 0$. Obviously, it cannot be the case that $P_i(f(t)) > 0$ for all $i \in \{s+1, \ldots, l\}$ since we chose t with $f(t) \in W_0 \backslash V_0$. So, let $J \subseteq \{s+1, \ldots, l\}$ ($|J| > 0$) such that $P_i(f(t)) < 0$ for all $i \in J$ and $P_i(f(t)) > 0$ for all $i \in \{s+1, \ldots, l\} \backslash J$. Because path functions and polynomials are continuous functions, there exists $\Delta > 0$ such that $P_i(f(t-\Delta)) < 0$ for all $i \in J$ and $P_i(f(t-\Delta)) > 0$ for all $i \in \{s+1, \ldots, l\} \backslash J$. Therefore, $f(t-\Delta) \in W_0 \backslash V_0$, which is a contradiction with the minimality of t.

Thus, there exists $i_1 \in \{s+1, \ldots, l\}$ such that $P_{i_1}(f(t)) = 0$. Since both P_{i_1} and f are continuous functions and $P_{i_1}(f(t')) > 0$ for all $t' \in [0, t)$, it must exist $t_0 \in [0, t)$ such that $P_{i_1}(f(t_0)) = \varepsilon$ for sufficiently small ε. Note that $f(t_0) \in V_0$ because of the minimality of t.

Now, consider the following two system of inequalities, and let V_1 be a connected component of solutions of (6) that contains $f(t_0)$ and W_1 a connected component of solutions of (7) that contains $f(t_0)$.

$$\begin{cases} P_1 = 0, \\ \cdots \\ P_s = 0, \\ P_{i_1} = \varepsilon, \\ P_i > 0, \forall i \in \{s+1, \ldots, l\} \backslash \{i_1\}, \end{cases} \tag{6}$$

$$\begin{cases} P_1 = 0, \\ \cdots \\ P_s = 0, \\ P_{i_1} = \varepsilon, \end{cases} \tag{7}$$

It is easy to see that $V_0 \supseteq V_1$ and $W_0 \supseteq W_1 \supseteq V_1$. We can repeat the same type of reasoning until we find $W_i = V_i$ for some $i \in \{0, \ldots, l-s\}$. □

Therefore, in order to enumerate all feasible sign vectors, it is enough to find a point in each connected component of all systems of at most l equations of the form $P_i = e_i\varepsilon$ for $e_i \in \{-1, 0, 1\}$. Then, substitute these points in $P_1, \ldots, P_l$ and obtain the corresponding sign vectors. This is a huge number of systems (more precisely, $\sum_{i=1}^{l} 3^i \binom{l}{i}$). Nevertheless, as we shall see, if the family of polynomials satisfies certain properties, there is no need to explore all of them. For example, if the polynomials are in general position, then one needs to check only systems of at most m equations (see [3] for details).

3 Global Optimum Clustering

Recall that our objective is to find $S_1, \ldots, S_k$ that minimizes $f_{S_1,\ldots,S_k}$. The main idea is to generate all possible partitions and to keep the one with the lowest squared error. But instead of plainly enumerating all partitions, we enumerate one point for each equivalence class of the quotient set $\mathbb{R}^{dk}/_{\equiv_{\mathbf{P}}}$ (where $\mathbf{P}$ is a family of $nk(k-1)/2$ polynomials defined as in (1)), which in turn define Voronoi partitions.

We have seen that enumerating the elements of $\mathbb{R}^{dk}/_{\equiv_{\mathbf{P}}}$ is equivalent to enumerating all cells in the cell arrangement defined by the following set of algebraic varieties:

$$\mathcal{H} = \bigcup_{\boldsymbol{p}\in S} \bigcup_{1\le i<j\le k} H_{\boldsymbol{p}(ij)}, \text{ where } H_{\boldsymbol{p}(ij)} = \{\boldsymbol{x} \in \mathbb{R}^{dk} \mid P_{\boldsymbol{p}(ij)}(\boldsymbol{x}) = 0\} \qquad (8)$$

Constructing the Voronoi partition defined by a point $\boldsymbol{q}$ in $\mathbb{R}^{dk}$ can be easily done once we know its sign vector: an observation $\boldsymbol{p}$ belongs to the cluster S_i if and only if $\boldsymbol{q} \in H^-_{\boldsymbol{p}(ij)} \cup H_{\boldsymbol{p}(ij)}$ for all $j > i$ and $\boldsymbol{q} \in H^+_{\boldsymbol{p}(ji)}$ for all $j < i$.

Thus, if two different points $\boldsymbol{q}$ and $\boldsymbol{q}'$ in $\mathbb{R}^{dk}$ are in the same equivalence class with respect to $\mathbf{P}$, then they have the same sign vector, and hence, they define the same clustering. On the other hand, the converse is not true, that is, the same clustering may correspond to non-equivalent points. Indeed, since the order in which we list the sets of each partition does not matter, we may end up having different sign vectors defining the same clustering simply because $\{S_1, S_2\}$ and $\{S_2, S_1\}$ represent the same clustering. But, it may also happen that two different vector signs define the same partition, in the same order (see example below).

Example 1. Let $S = \{-11, 0, 10\}$ be a set of three points in $\mathbb{R}$, and assume we want to separate them into three clusters. As we shall see, there are many vectors in $\mathbb{R}^3$ that lead to the same clustering although they have different sign vectors. First, we need to define 9 polynomials (3 for each point):

$P_{-11(1,2)} = X_1^2 - X_2^2 - 2 * (-11) * (X_1 - X_2)$
$P_{-11(1,3)} = X_1^2 - X_3^2 - 2 * (-11) * (X_1 - X_3)$
$P_{-11(2,3)} = X_2^2 - X_3^2 - 2 * (-11) * (X_2 - X_3)$
$P_{0(1,2)} = X_1^2 - X_2^2 - 2 * 0 * (X_1 - X_2)$

$P_{0(1,3)} = X_1^2 - X_3^2 - 2 * 0 * (X_1 - X_3)$
$P_{0(2,3)} = X_2^2 - X_3^2 - 2 * 0 * (X_2 - X_3)$
$P_{10(1,2)} = X_1^2 - X_2^2 - 2 * 10 * (X_1 - X_2)$
$P_{10(1,3)} = X_1^2 - X_3^2 - 2 * 10 * (X_1 - X_3)$
$P_{10(2,3)} = X_2^2 - X_3^2 - 2 * 10 * (X_2 - X_3)$

Now take $\boldsymbol{q} = (-10, 0, 11)$ and $\boldsymbol{q}' = (-11, 0, 10)$. It can be checked that
$sv(\boldsymbol{q}) = (+1, -1, -1, +1, +1, +1, -1, -1, -1)$,
$sv(\boldsymbol{q}') = (+1, +1, -1, +1, +1, +1, -1, -1, -1)$.
Nevertheless, the partition defined by both sign vectors is $\{S_1, S_2, S_3\}$, where $S_1 = \{-11\}$, $S_2 = \{0\}$ and $S_3 = \{10\}$.

The important thing is that going through all the cells of this hyperplane arrangement, although we might not get all possible clusterings, we do get the optimal one.

Theorem 1. *Let $\mathcal{H}$ be as in* (8)*. There exists $\boldsymbol{q}$ in $\mathbb{R}^{dk}$ such that the Voronoi partition defined by $\boldsymbol{q}$ is optimal*[2]*.*

Proof. Let $S = \{S_1, \ldots, S_k\}$ be an optimal partition. It is easy to check (by making the partial derivatives equal zero, and solving the equations) that among all possible sets of centroids, the one that minimizes the mean squared error is defined by:

$$\boldsymbol{q}_i = \frac{1}{|S_i|} \sum_{\boldsymbol{p} \in S_i} \boldsymbol{p}$$

Now let us show that the Voronoi partition $\{S'_1, \ldots, S'_k\}$ defined by $sv(\boldsymbol{q})$ for $\boldsymbol{q} = (q_1, \ldots, q_k)$, although it may be different from $\{S_1, \ldots, S_k\}$, it has the same mean squared error (thus also being an optimal partition).

Assume by contrary that it does not. Take $i \in \{1, \ldots, k\}$ to be the smallest index such that $S_i \neq S'_i$. Let $\boldsymbol{p}$ be a point in their symmetric difference $(S_i \backslash S'_i) \cup (S'_i \backslash S_i)$. We distinguish two cases:

- $\boldsymbol{p} \in S_i \backslash S'_i$. Thus, $\exists j > i$ such that $\boldsymbol{p} \in S'_j$, and therefore, $\|\boldsymbol{p} - \boldsymbol{q_j}\|^2 < \|\boldsymbol{p} - \boldsymbol{q_i}\|^2$. Consider now the partition constructed from $\{S_1, \ldots, S_k\}$ by moving $\boldsymbol{p}$ from S_i to S_j. It would have a mean squared error with respect to $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)$ strictly smaller than the optimal one (because the contribution to the sum of $\boldsymbol{p}$ is smaller), a contradiction.
- $\boldsymbol{p} \in S'_i \backslash S_i$. Thus, $\exists j > i$ such that $\boldsymbol{p} \in S_j$, and $\|\boldsymbol{p} - \boldsymbol{q_i}\|^2 \leq \|\boldsymbol{p} - \boldsymbol{q_j}\|^2$. Now, if $\|\boldsymbol{p} - \boldsymbol{q_i}\|^2 < \|\boldsymbol{p} - \boldsymbol{q_j}\|^2$ we construct a new partition from $\{S_1, \ldots, S_k\}$ by moving $\boldsymbol{p}$ from S_j to S_i. This new partition would have a strictly smaller mean squared error with respect to $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)$ than the optimal one, a contradiction. We are left with the case in which $\|\boldsymbol{p} - \boldsymbol{q_i}\|^2 = \|\boldsymbol{p} - \boldsymbol{q_j}\|^2$. If $\boldsymbol{p}$ was the only point that distinguished the two partitions, then both

[2] this is already stated (without proof) in [11] and [12] as being a known fact, but the references given there ([4, 17]) do not include such a result.

partitions would have the same minimal mean squared error with respect to $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)$, contradicting our assumption. So there must be another point, say $\boldsymbol{p}'$, such that either:

- $\boldsymbol{p}' \in S_i' \backslash S_i$ and $\|\boldsymbol{p} - \boldsymbol{q_i}\|^2 < \|\boldsymbol{p} - \boldsymbol{q_j}\|^2$ for some $j > i$, or
- $\boldsymbol{p}' \in (S_j \backslash S_j') \cup (S_j' \backslash S_j)$ for some $j > i$.

Following the same type of reasoning, we can check that both cases eventually lead to a contradiction.

The idea that cell arrangements can be used to enumerate all Voronoi partitions appeared already in [11], where the authors claim (without proof) that "all the Voronoi partitions can be enumerated in $\mathcal{O}(n^{dk(k+1)/2})$ time by using the hyperplane arrangement". One year later, the same authors (three of them) argue that "the number of Voronoi partitions is bounded by the combinatorial complexity of $nk(k-1)/2$ constant-degree algebraic surfaces", and therefore, "all the Voronoi partitions can be enumerated in $\mathcal{O}(n^{dk+1})$ time" (see [12, pp. 335]), with no further reference about which is the cell enumeration algorithm they had in mind. Clearly, it cannot be the one we presented in Section 2.2 simply because this technique was discovered five years later. We believe that the authors referred to the following result.

Theorem 2. *[5] Given $\mathcal{H}$ an arrangement of l algebraic varieties in $\mathbb{R}^m$, then there exist an algorithm for the cell enumeration problem with time complexity $\mathcal{O}\left(l^{m+1}\right)$. where the implied constants only depends on m and the maximum of the degrees of the varieties.*

However, the "implied constants" are as big as 2^{m^2}, which makes the algorithm in [3] in practice much faster than the one introduced in [5]. We next proceed to describe the algorithm that searches for the optimum clustering.

First, we show how Algorithm **??** can be improved for $k = 2$ and then we use this result for general k.

3.1 Binary Splitting ($k = 2$)

Here we consider the binary splitting problem: finding the globally optimum way of separating the data points into two clusters, for arbitrary dimension.

A first improvement of the algorithm outlined in Section 3.5 is an algorithm that explores $\mathcal{O}(f(n,d))$ cells in order to return the optimal clustering. This improves the $\mathcal{O}(n^{2d+1})$ bound given in [12].

The basic idea is that Algorithm **??** checks tuples of ℓ' polynomials with ℓ' in $\{1, \ldots, l\}$. We show that for the particular case of $k = 2$, it is enough to look only at tuples of dimension at most $d + 1$.

Lemma 1 *Given the following system of $d' > d + 2$ equations,*

$$\begin{cases} P_{\boldsymbol{p}_1(12)} = e_1\varepsilon, \\ P_{\boldsymbol{p}_2(12)} = e_2\varepsilon, \\ \ldots = \ldots, \\ P_{\boldsymbol{p}_{d'}(12)} = e_{d'}\varepsilon, \end{cases}$$

where

- ε *is a infinitesimal,*
- $e_1, \ldots, e_{d'} \in \{-1, 0, +1\}$*,*

all its solutions can be obtained using at most $d+1$ *equations.*

Proof. First, let us note that an equivalent system of equations is the following one:

$$\begin{cases} P_{\boldsymbol{p}_1(12)} = e_1\varepsilon, \\ P_{\boldsymbol{p}_2(12)} - P_{\boldsymbol{p}_1(12)} = (e_2 - e_1)\varepsilon, \\ \ldots = \ldots, \\ P_{\boldsymbol{p}_i(12)} - P_{\boldsymbol{p}_1(12)} = (e_i - e_1)\varepsilon, \\ \ldots = \ldots, \\ P_{\boldsymbol{p}_{d'}(12)} - P_{\boldsymbol{p}_1(12)} = (e_{d'} - e_1)\varepsilon, \end{cases}$$

All these equations, with the exception of the first one, are linear, since

$$P_{\boldsymbol{p}_i(12)} - P_{\boldsymbol{p}_1(12)} = -2\sum_{r=1}^{d}(p_r^{(i)} - p_r^{(1)})(X_r - X_{r+d}).$$

where $\boldsymbol{p}_i = (p_1^{(i)}, \ldots, p_d^{(i)})$ for all i in $\{1, \ldots, d+2\}$.

We perform the following change of variables:
$Y_r = X_r - X_{r+d}$, for all $r \in \{1, \ldots, d\}$
$Z_r = X_r + X_{r+d}$, for all $r \in \{1, \ldots, d\}$.
Our system of equations becomes:

$$\begin{cases} \displaystyle\sum_{r=1}^{d} Y_r(Z_r - 2p_r^{(1)}) = e_1\varepsilon, \\ \displaystyle -2\sum_{r=1}^{d}(p_r^{(2)} - p_r^{(1)})Y_r = (e_2 - e_1)\varepsilon, \\ \ldots = \ldots, \\ \displaystyle -2\sum_{r=1}^{d}(p_r^{(i)} - p_r^{(1)})Y_r = (e_i - e_1)\varepsilon, \\ \ldots = \ldots, \\ \displaystyle -2\sum_{r=1}^{d}(p_r^{(d')} - p_r^{(1)})Y_r = (e_{d'} - e_1)\varepsilon, \end{cases}$$

In matrix notation,

$$\begin{cases} \displaystyle\sum_{r=1}^{d} Y_r(Z_r - 2p_r^{(1)}) = e_1\varepsilon, \\ A \cdot Y = b \end{cases} \tag{9}$$

where $Y = (Y_1, \ldots, Y_d)^T$, $b = \varepsilon(e_2 - e_1, \ldots, e_{d'} - e_1)^T$ and $A = (a_{ij})_{i,j}$ with $a_{ij} = 2(p_j^{(1)} - p_j^{(i+1)})$ for all $i \in \{1, \ldots, d'-1\}$ and $j \in \{1, \ldots, d\}$. Since the rank of A is at most d, the system of equations $A \cdot Y = b$ either has no solutions, being inconsistent, or, if it does have one or more solutions, it must be the case that one of the equations can be written as a linear combination of other equations, and thus, it can be eliminated while maintaining an equivalent system. This concludes our proof. □

This is the first refinement, the idea is instead of taking up to l observations and assign them in clusters, it is only necessary to take at most $d+1$ points for defining the two clusters. In order to present the actual algorithm we need some further notations. Given a vector sign $\boldsymbol{e} = (e_1, \ldots, e_l)$ in $\{-1, 0, +1\}^l$ and $I = \{i_1, \ldots, i_{l'}\} \subseteq \{1, \ldots, l\}$, we define the following polynomial

$$G'_{\boldsymbol{e},I}(X_1, \ldots, X_m) = \sum_{j \in I} (P_j(X_1, \ldots, X_m) - e_j\varepsilon)^2, \tag{10}$$

Moreover, let $h_I : \mathbb{R}^{l'} \to \mathbb{R}^l$ be a function such that if $\boldsymbol{e} = (e_1, \ldots, e_{l'})$ then

$$h_I(\boldsymbol{e})_i = \begin{cases} 0, & i \notin I \\ e_j, & i = i_j \in I \end{cases}$$

(I am sure there is a name for this kind of function... inverse projection?)

The algorithm is presented below.

Algorithm 1 A First Improvement of Algorithm **??** for the case $n > 2d$

```
Input: {P_1, ..., P_l} polynomials in 2d variables defined as in (1)
for all l' in {1, ..., d, d+1} do
    for all I ⊆ {1, ..., n} with |I| = l' do
        for all e ∈ {−1, 0, +1}^{l'} do
            e' := h_I(e)
            if ∃x ∈ R^m such that G'_{e',I}(x) = 0 then
                output (x, sv(x))
            end if
        end for
    end for
end for
```

Theorem 3. *Given $S \subset \mathbb{R}^d$ a set of cardinality n. Then the number of Voronoi partitions into two clusters is, at most,*

$$f(n, d+1) = \sum_{i=1}^{d+1} 3^i \binom{n}{i},$$

and the number of operations necessary to generate each of the partitions is of order $d^2 n$.

Thus, the global optimum can be obtained performing $nf(n, d+1)$ operations. We present a second refinement, which can be applied whenever l is much bigger than m.

Lemma 2 *Let $B = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d\}$ be a base in $\mathbb{R}^d$ and $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d'}\}$ a subset of S, with $1 \leq d' < d$, and consider the following system of d' equations*

$$\begin{cases} P_{\boldsymbol{p}_1(12)} = e_1\varepsilon, \\ P_{\boldsymbol{p}_2(12)} = e_2\varepsilon, \\ \ldots = \ldots, \\ P_{\boldsymbol{p}_{d'}(12)} = e_{d'}\varepsilon, \end{cases}$$

where

- *ε is a infinitesimal,*
- *$e_1, \ldots, e_{d'} \in \{-1, 0, +1\}$,*

If the system is consistent, then there exists a point $\boldsymbol{p}$ in $S \cup B \backslash \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d'}\}$ such that by adding the equation $P_{\boldsymbol{p}(12)} = e_1\varepsilon$, the system will continue to be consistent but will have strictly higher rank. Moreover, any solution of the later system will be solution to the initial one.

Proof. Performing the same change of variables as we did in the proof of Lemma 1, our system of equations becomes

$$\begin{cases} \sum_{r=1}^{d} Y_r(Z_r - 2p_r^{(1)}) = e_1\varepsilon, \\ A \cdot Y = b \end{cases}$$

where $Y = (Y_1, \ldots, Y_d)^T$, $b = \varepsilon(e_2 - e_1, \ldots, e_{d'} - e_1)^T$ and $A = (a_{ij})_{i,j}$ with $a_{ij} = 2(p_j^{(1)} - p_j^{(i+1)})$ for all $i \in \{1, \ldots, d'-1\}$ and $j \in \{1, \ldots, d\}$. It is clear that this system of equations is consistent if and only if $A \cdot Y = b$ has solutions. Note that matrix A is a $(d'-1) \times d$ matrix, thus having at most rank $d'-1$.

Now, if $A \cdot Y = b$ is consistent, let $\boldsymbol{p} = (p_1, \ldots, p_d)$ in B be such that $\boldsymbol{p}$ cannot be written as a linear combination of $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d'}\}$ (such a point must exist since otherwise $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d'}\}$ would be a base for $\mathbb{R}^d$, which is impossible given that $d' < d$). The system obtained by adding the equation $P_{\boldsymbol{p}(12)} = e_1\varepsilon$ is consistent and has strictly higher rank since $\boldsymbol{p} - \boldsymbol{p}_1$ cannot be written as linear combination of $\{\boldsymbol{p}_2 - \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d'} - \boldsymbol{p}_1\}$:

$$\begin{cases} \sum_{r=1}^{d} Y_r(Z_r - 2p_r^{(1)}) = e_1\varepsilon, \\ A \cdot Y = b \\ \sum_{r=1}^{d} 2(p_r^{(1)} - p_r)Y_r = 0, \end{cases}$$

Moreover, it is easy to see that any solution of the later system will be solution to the initial one.

□

An immediate consequence of Lemma 2 is that if we have a consistent system of at most $d-1$ equations as in (9), we can complete it with more points in B until the rank of matrix A is $d-1$ such that every solution of the new system is a solution of the initial one. Therefore, it is enough to check sets with at least d polynomials. The following theorem then trivially follows from the results above.

Theorem 4. *Given $S \subset \mathbb{R}^d$ a set of cardinality n. Then the number of Voronoi partitions into two clusters is, at most,*

$$g(n,d) = 3^d \binom{n+d}{d} + 3^{d+1} \binom{n}{d+1},$$

and the number of operations necessary to generate each of the partitions is of order $d^2 n$.

The algorithm becomes:

Algorithm 2 A Second Improvement of Algorithm **??** for the case $n > 2d$

```
Input: {P_1, ..., P_l} polynomials in 2d variables defined as in (1)
for all I ⊆ {1, ..., n} with |I| = d + 1 do
    for all e ∈ {−1, 0, +1}^{d+1} do
        e' := h_I(e)
        if ∃x ∈ R^m such that G'_{e',I}(x) = 0 then
            output (x, sv(x))
        end if
    end for
end for
let B be a base in R^d
for all I ⊆ {1, ..., n} ∪ B with |I| = d do
    for all e ∈ {−1, 0, +1}^d do
        e' := h_I(e)
        if ∃x ∈ R^m such that G'_{e',I}(x) = 0 then
            output (x, sv(x))
        end if
    end for
end for
```

The global optimum can therefore be obtained by performing the following number of operations: $n \min\left(f(n,d+1), g(n,d)\right)$.

3.2 Reducing k-clustering to 2-clustering

In order to explain how the k-clustering problem can be reduced to the 2-clustering problem we need to introduce some notation. As before, let $S =$

$\{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$ and let us fix $k > 2$. We define a family of projection functions $pr^{ij} : \mathbb{R}^{dk} \to \mathbb{R}^{2d}$ by

$$pr^{ij}(x_1, \ldots, x_{dk}) = (x_{(i-1)d+1}, \ldots, x_{id}, x_{(j-1)d+1}, \ldots, x_{jd}).$$

Let $\mathbf{P} = (P_{\boldsymbol{p}(ij)})_{\boldsymbol{p} \in S, 1 \leq i < j \leq k}$ be a family of $nk(k-1)/2$ polynomials in $\mathbb{R}[X_1, \ldots, X_{dk}]$ defined as in (1) and $\mathbf{P'} = (P'_{\boldsymbol{p}(ij)})_{\boldsymbol{p} \in S, 1 \leq i < j \leq k}$ the family of n polynomials in $\mathbb{R}[X_1, \ldots, X_{2d}]$ defined as in (1) for the 2-clustering problem.

For a better readability we choose to rename the polynomials in $\mathbf{P}$ as follows.

$$\begin{array}{l} P_1 = P_{\boldsymbol{p}_1(12)}\ P_{n+1} = P_{\boldsymbol{p}_1(23)}\ P_{2n+1} = P_{\boldsymbol{p}_1(34)}\ \cdots\ P_{n(k^2-k-2)/2+1} = P_{\boldsymbol{p}_1(1k)} \\ P_2 = P_{\boldsymbol{p}_2(12)}\ P_{n+2} = P_{\boldsymbol{p}_1(23)}\ P_{2n+2} = P_{\boldsymbol{p}_2(34)}\ \cdots\ P_{n(k^2-k-2)/2+2} = P_{\boldsymbol{p}_2(1k)} \\ \quad \vdots \\ P_n = P_{\boldsymbol{p}_n(12)}\ P_{n+n} = P_{\boldsymbol{p}_1(23)}\ P_{2n+n} = P_{\boldsymbol{p}_n(34)}\ \cdots\ P_{n(k^2-k-2)/2+n} = P_{\boldsymbol{p}_n(1k)} \end{array}$$

That is, $P_s = P_{\boldsymbol{p}_t(ij)}$ where $s = t + n(i - 1 + (j - i - 1)(2k - j + 1)/2)$. The same order applies to polynomials in $\mathbf{P'}$.

An important observation is that given $\boldsymbol{p}$ in S, $P_{\boldsymbol{p}(ij)}$ only uses $2d$ of its variables, being transformable to $P'_{\boldsymbol{p}(12)}$ for all $1 \leq i < j \leq k$. More formally, $P_{\boldsymbol{p}(ij)}(\boldsymbol{x}) = P'_{\boldsymbol{p}(12)}(pr^{ij}(\boldsymbol{x}))$

With this notation, we have the following lemma.

Lemma 1. *If $(e_1, \ldots, e_l)$ is a feasible sign vector with respect to $\mathbf{P}$ ($l = nk(k-1)/2$), then $(e_1, \ldots, e_n), (e_{n+1}, \ldots, e_{2n}), \ldots, (e_{n(k^2-k-2)/2+1}, \ldots, e_{nk(k-1)/2})$ are all feasible sign vectors with respect to $\mathbf{P'}$.*

Proof. Let $\boldsymbol{e} = (e_1, \ldots, e_l)$ be a feasible sign vector with respect to $\mathbf{P}$. Then there exists $\boldsymbol{x} = (x_1, \ldots, x_{kd})$ in $\mathbb{R}^{dk}$ such that $sv(\boldsymbol{x}) = \boldsymbol{e}$. Therefore, for each $1 \leq i < j \leq k$ and $s = n(i - 1 + (j - i - 1)(2k - j + 1)/2)$ we have:

$$\begin{aligned} (e_{s+1}, \ldots, e_{s+n}) &= (sign(P_{s+1}(\boldsymbol{x})), \ldots, sign(P_{s+n}(\boldsymbol{x}))) \\ &= (sign(P_{\boldsymbol{p}_1(ij)}(\boldsymbol{x})), \ldots, sign(P_{\boldsymbol{p}_n(ij)}(\boldsymbol{x}))) \\ &= (sign(P'_{\boldsymbol{p}_1(12)}(pr^{ij}(\boldsymbol{x}))), \ldots, sign(P'_{\boldsymbol{p}_n(12)})(pr^{ij}(\boldsymbol{x}))) \\ &= (sign(P'_{\boldsymbol{p}_1(12)}(\boldsymbol{y})), \ldots, sign(P'_{\boldsymbol{p}_n(12)})(\boldsymbol{y})) \\ &= sv(\boldsymbol{y}) \text{ with respect to } \mathbf{P'}. \end{aligned}$$

where $\boldsymbol{y} = pr^{ij}(\boldsymbol{x})$. Hence, each $(e_{s+1}, \ldots, e_{s+n})$ for s in $\{1, \ldots, k(k-1)/2\}$[3] is a feasible vector with respect to $\mathbf{P'}$, which concludes our proof.

This lemma ensures that the following algorithm is correctly outputting at least all feasible sign vectors with respect to $\mathbf{P}$ (it may also output non feasible sign vectors, but those will not affect the solution to the k-clustering problem).

The complexity of this algorithm is $\mathcal{O}(nh(n,d) + h(n,d)^k)$ where $h(n,d) = \min(f(n, d+1), g(n,d))$, since by Theorem 3 and Theorem 4, E can have at most $h(n,d)$ elements.

[3] this is due to the fact that $1 \leq i < j \leq k$ covers all polynomials in $\mathbf{P}$

Algorithm 3 A First Improvement of Algorithm **??** for the case $n > 2d$

1: Input: the family $\mathbf{P}$ of polynomials in $\mathbb{R}[X_1, \ldots, X_{dk}]$
2: E = the set of sign vectors output by Algorithm 1 or 2 on $\mathbf{P}'$
3: **for all** $\boldsymbol{e} \in E^k$ **do**
4: output $\boldsymbol{e}$
5: **end for**

3.3 Reduced Number of Clusters ($k \leq d$)

For the case $k > 2$, we can enunciate another bound for the number of Voronoi partitions of S into k clusters.

Theorem 5. *Given $S \subset \mathbb{R}^d$ a set of cardinality n. Then the number of partitions into k clusters is, at most,*

$$\sum_{j=1}^{(d+1)(k-1)} \binom{n}{j},$$

and the number of operations necessary to generate each of the partitions is $\mathcal{O}(n)$, where the implied constant depends only on d and k.

Proof. Notice that for each $\boldsymbol{p}$ in S and $1 < i < j \leq k$, $P_{\boldsymbol{p}(ij)} = P_{\boldsymbol{p}(1j)} - P_{\boldsymbol{p}(1i)}$, and each $P_{\boldsymbol{p}(1i)} = \sum_{r=1}^{d} X_r^2 - X_{r+(i-1)d}^2 - \sum_{r=1}^{d} 2p_r(X_r - X_{r+(i-1)d})$ can be written as $P_{\boldsymbol{p}(1i)} = Z^{(i-1)} - \sum_{r=1}^{d} 2p_r Y_r^{(i-1)}$, where $Z^{(i-1)} = \sum_{r=1}^{d} X_r^2 - X_{r+(i-1)d}^2$ and $Y_r^{(i-1)} = X_r - X_{r+(i-1)d}$ for all i in $\{2, \ldots, k\}$.

With this change of variables, $P_{\boldsymbol{p}(ij)}$ is linear, so we can use linear algebra techniques to generate all solutions in time $\mathcal{O}(f(n, (d+1)(k-1))$.

3.4 Discretization of Numeric Attributes ($d = 1$)

For $d = 1$, the japos theorem gives $\mathcal{O}(n^{k+1})$ running time to traverse all the clusterings; our theorem is worse as $k > d$: it gives $\mathcal{O}(n^{2k-1})$. We show that, actually, it is possible to perform the task in $\mathcal{O}(n^{k-1})$ operations. Thus, finding the globally optimum k-means discretization for not too large datasets and a smallish number of bins may be feasible in practice. We report on actual running times of sequential and parallel implementations below.

Solving $k-$clustering problem can be done efficiently when $d = 1$. The result follows from the following lemma.

Lemma 2. *The number of Voronoi partitions of a set $S \subset \mathbb{R}$ of n points generated by k points is $O(n^{(k-1)})$, and all the Voronoi partitions can be enumerated in $O(n^{(k-1)})$ where the constant depends on k. Therefore, the optimum $k-$clustering of S can be found in $O(n^{(k-1)})$ operations.*

Proof. By the definition of Voronoi partition, it is easy that each $S_1, \ldots, S_k$ is defined by its maximum and minimum element. The reason is Voronoi regions

are convex regions, so if two points belong to a Voronoi region, then the line that connects the points belongs to the Voronoi region.

Notice also, that the set S has a minimum value, so this value has to be in one of the partitions. Without losing any generality, suppose that it is in the first one, since the order of the sets $S_1, \ldots, S_k$ is irrelevant for the partition.

Fixing the minimum of each $S_2, \ldots, S_k$, fix the partition. The reason is that the the maximum of S_1 is given by the following formula:

$$\max S_1 = \max\{s \in S \mid s \leq \min_{j=2,\ldots,k} S_j\}.$$

To recover $S_2, \ldots, S_k$, it is only necessary to apply the same reasoning to $S' = S_2 \cup \ldots \cup S_k$. The number of choices for the minimum is exactly the number of sets of k elements of S (that is, $\binom{n-1}{k-1}$), which is $\mathcal{O}(n^{k-1})$ and enumerating the sets is trivial. This remark finishes the proof.

3.5 A Running Example of Algorithm ??

Let $S = \{(1,0), (-1,0), (0,1), (0,-1), (0,2)\}$ and assume we are interested in partitioning S into two different clusters (thus, $d = 2$, $k = 2$, $m = 4$ and $l = 5$).

First, for each point $\boldsymbol{p}$ in S and for all pairs (i,j) con $1 \leq i < j \leq k$ we need to define a polynomial $P_{\boldsymbol{p}(ij)}$ in $\mathbb{R}^{dk}$. Since $k = 2$ and (i,j) can only take the value $(1,2)$, we drop the subscript (ij) from our notation. Moreover, instead of $P_{\boldsymbol{p_i}}$ we simply write P_i, where $\boldsymbol{p}_i$ represents the i-th point in S.

$$\begin{aligned}
P_1 &= (X_1^2 - X_3^2) + (X_2^2 - X_4^2) - 2(X_1 - X_3) - 0(X_2 - X_4),\\
P_2 &= (X_1^2 - X_3^2) + (X_2^2 - X_4^2) + 2(X_1 - X_3) - 0(X_2 - X_4),\\
P_3 &= (X_1^2 - X_3^2) + (X_2^2 - X_4^2) - 0(X_1 - X_3) - 2(X_2 - X_4),\\
P_4 &= (X_1^2 - X_3^2) + (X_2^2 - X_4^2) - 0(X_1 - X_3) + 2(X_2 - X_4),\\
P_5 &= (X_1^2 - X_3^2) + (X_2^2 - X_4^2) - 0(X_1 - X_3) - 4(X_2 - X_4),
\end{aligned}$$

Thus, the four algebraic surfaces that define our arrangement are

$$\begin{aligned}
H_1 &= \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 \mid (x_1-1)^2 + (x_2)^2 - (x_3-1)^2 - (x_4)^2 = 0\},\\
H_2 &= \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 \mid (x_1+1)^2 + (x_2)^2 - (x_3+1)^2 - (x_4)^2 = 0\},\\
H_3 &= \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 \mid (x_1)^2 + (x_2-1)^2 - (x_3)^2 - (x_4-1)^2 = 0\},\\
H_4 &= \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 \mid (x_1)^2 + (x_2+1)^2 - (x_3)^2 - (x_4+1)^2 = 0\},\\
H_5 &= \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 \mid (x_1)^2 + (x_2-2)^2 - (x_3)^2 - (x_4-2)^2 = 0\},
\end{aligned}$$

Algorithm **??** starts with $l' = 1$. For each i in $\{1,2,3,4,5\}$, it tries all e in $\{-1,0,+1\}$, and searches for a solution to the equation

$$(P_i - e\varepsilon)^2 = 0.$$

We will only detail here one of these, for example, for $i = 1$ and $e = 1$. The equation $P_1 = \varepsilon$ has many solutions. Consider for example $\boldsymbol{x} = (1, \sqrt{\varepsilon}, 1, 0)$. We compute $P_j(\boldsymbol{x})$ for all $j > 1$ and obtain that $sv(\boldsymbol{x}) = (+1, +1, -1, +1, -1)$ (keep in mind that ε is an infinitesimal). In order to compute the partition defined by $sv(\boldsymbol{x})$, recall that $\boldsymbol{p} \in S_i$ if and only if $\boldsymbol{x} \in H^-_{\boldsymbol{p}(ij)} \cup H_{\boldsymbol{p}(ij)}$ for all $j > i$ and $\boldsymbol{x} \in H^+_{\boldsymbol{p}(ji)}$ for all $j < i$. Dropping unnecessary indexes, $S_1 = \{\boldsymbol{p}_i \in S \mid \boldsymbol{x} \in H_i^- \cup H_i\}$ and $S_2 = \{\boldsymbol{p}_i \in S \mid \boldsymbol{x} \in H_i^+\}$. That is, $S_1 = \{\boldsymbol{p}_3, \boldsymbol{p}_5\}$ and $S_2 = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_4\}$.

Other sign vectors obtained for $l' = 1$ are $(-1, -1, +1, -1, +1)$, $(0, 0, 0, 0, 0)$, $(+1, -1, -1, -1, -1)$ and $(-1, +1, +1, +1, +1)$, which only lead to one more valid 2-clustering of the initial set of points: $\{\boldsymbol{p}_1\}$ and $\{\boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5\}$.

The algorithm then checks 2-tuples and 3-tuples.

4 Parallel Implementation on a Computing Cluster

Explanation of the computing cluster

Same as before on it

Podemos clusterizar en 2 (binary splitting) todo diabetes en paralelo?

Podemos hacerlo para 100 tuplas? 200? 300? ...

One example for each subsection.

5 Quality of the clustering?

Compare to k-means heuristico - esto hay que pensar un poco como hacerlo!

6 Conclusion

We have used the idea in [] and have improved on running times.

Implementation available.

Allows for creating a "repository" with the value of global minimum for datasets so people can contrast their clustering algorithms against the optimum.

References

1. Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75:245–248, 2009. 10.1007/s10994-009-5103-0.
2. 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.
3. Saugata Basu, Richard Pollack, and Marie-Françoise Roy. A new algorithm to find a point in every cell defined by a family of polynomials. In *Quantifier elimination and cylindrical algebraic decomposition (Linz, 1993)*, Texts Monogr. Symbol. Comput., pages 341–350. Springer, Vienna, 1998.

4. Endre Boros and Peter L. Hammer. On clustering problems with connected optima in euclidean spaces. *Discrete Mathematics*, 75(1-3):81–88, 1989.
5. John Canny. Improved algorithms for sign determination and existential quantifier elimination. *Comput. J.*, 36(5):409–418, 1993.
6. S. Dasgupta. The hardness of k-means clustering. (Technical Report CS2008-0916). University of California., 2008.
7. Rod G. Downey and M.R. Fellows. *Parameterized complexity*. Monographs in computer science. Springer, 1999.
8. P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33, 2004. 10.1023/B:MACH.0000033113.59016.96.
9. Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.
10. Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.
11. Susumu Hasegawa, Hiroshi Imai, Mary Inaba, and Naoki Katoh. Efficient algorithms for variance-based k-clustering. In *Proceedings of the First pacific Conference on Computer Graphics and Applications*, pages 75–89, 1993.
12. Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based *k*-clustering (extended abstract). In *Symposium on Computational Geometry*, pages 332–339, 1994.
13. Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.
14. J. 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.
15. Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation*, WALCOM '09, pages 274–285, Berlin, Heidelberg, 2009. Springer-Verlag.
16. Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
17. S. J. Wan, S. K. M. Wong, and P. Prusinkiewicz. An algorithm for multidimensional data clustering. *ACM Trans. Math. Softw.*, 14(2):153–162, June 1988.
18. Ying Yang and Geoffrey I. Webb. Proportional k-interval discretization for naive-bayes classifiers. In Luc De Raedt and Peter A. Flach, editors, *ECML*, volume 2167 of *Lecture Notes in Computer Science*, pages 564–575. Springer, 2001.
19. 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.