16th World Congress of the International Fuzzy Systems Association (IFSA)
9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)

A K-means-like algorithm for informetric data clustering

Anna Cena[1] **Marek Gagolewski**[1,2]

[1] Systems Research Institute, Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland
[2] Faculty of Mathematics and Information Science, Warsaw University of Technology,
ul. Koszykowa 75, 00-662 Warsaw, Poland

Abstract

The K-means algorithm is one of the most often used clustering techniques. However, when it comes to discovering clusters in informetric data sets that consist of non-increasingly ordered vectors of not necessarily conforming lengths, such a method cannot be applied directly. Hence, in this paper, we propose a K-means-like algorithm to determine groups of producers that are similar not only with respect to the quality of information resources they output, but also their quantity.

Keywords: k-means clustering, informetrics, aggregation, impact functions

1. Introduction

In the practice of data analysis and mining, the K-means algorithm is very often applied in order to automatically discover a partition of a given, unlabeled data set, so that objects within each cluster are similar as much as possible and objects in distinct groups differ (with respect to some criteria) as much as possible from each other (see e.g. [1]). K-means procedure can be easily applied to data sets consisting of n-dimensional real vectors. In case of informetric data sets, however, we are faced with sets of vectors of nonconforming lengths that are sorted non-increasingly. For example, one of the informetric tasks is the problem of evaluating the quality of information resources and their producers, the so-called Producers Assessment Problem (PAP, cf. [5]). Let $P = \{p_1, \ldots, p_l\}$ be a set of l producers and assume that each of them outputs n_i, $i = 1, \ldots, l$, products. Additionally, each product is given some kind of rating concerning its overall quality. The state of a producer p_i may be represented by a non-increasingly ordered sequence. Additionally, the lengths of vectors in PAP may vary from producer to producer.

Scientometrics, in which a scientist is considered as a information resources' producer and his/her scholarly papers are conceived as products, makes a perfect example here. Moreover, the papers' quality is often described by a function of the number of citations they received (see e.g. [6]). Similarly, a Facebook or Twitter user is also a kind of producer. In such a case, his/her posts are products, and the numbers of their "re-tweets" or "likes" can be considered as their quality assessment. PAP has many other instances, even in manufacturing and quality management [7].

Thus, the aim of this paper is to derive a K-means-like procedure for clustering informetric data. The structure of this contribution is as follows. Description of K-means algorithm is given in Sec. 2. In Sec. 3 a dissimilarity measure for vectors of nonconforming lengths is defined. Then, in Sec. 4 a procedure to determine the d_M^2-centroid of a given set of vectors is derived. Next, in Sec. 5, a fast algorithm to deal with this task is constructed. Sec. 6 presents results of an application of the proposed approach on a real-world data set. Finally, Sec. 7 concludes the paper and shows future research directions.

2. K-means algorithm

Let us begin with a short introduction of clustering task. Given a set of observations $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}\}$, where each $\mathbf{x}^{(i)} \in \mathbb{R}^n$, we aim at partitioning the l observations into k nonempty pairwise disjoint sets $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, $\bigcup_{i=1}^k C_i = \mathcal{X}$, so that:

$$\mathcal{C} = \underset{\text{partition } \mathcal{C} \text{ of } \mathcal{X}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d_{L_2}^2 \left(\mathbf{x}, \boldsymbol{\mu}^{(i)} \right), \qquad (1)$$

where $\boldsymbol{\mu}^{(i)}$ is the centroid of all the vectors in C_i, $\mu_j^{(i)} = \sum_{\mathbf{x} \in C_i} x_j / |C_i|$, and $d_{L_2}^2(\mathbf{x}, \boldsymbol{\mu}) = \sum_{j=1}^n (x_j - \mu_j)^2$ is the squared Euclidean distance. Of course, the squared Euclidean distance is actually not a metric in the mathematical sense, since in general it does not satisfy the triangle inequality. However, it is a so-called dissimilarity measure, i.e. a function $d(\mathbf{x}, \mathbf{y})$ such that $(\forall \mathbf{x}, \mathbf{y})$ (a) $d(\mathbf{x}, \mathbf{y}) \geq 0$, (b) $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$ and (c) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$. Nevertheless, the usage of d_{L_2} instead of $d_{L_2}^2$ is much more difficult to tackle both mathematically and computationally. In particular, there exists no analytic solution the the Euclidean 1-center problem, see e.g. [2].

As the problem stated in Eq. (1) is known to be NP-complete [3], the following heuristic is used in the K-means algorithm, see [4]. For the initial set

of cluster centers, do what follows until convergence occurs:

1. Assign each point in $\mathcal{X}$ to the cluster with the nearest center,
2. Recalculate cluster centers by computing the means $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(k)}$ of all the points assigned to particular clusters.

3. A dissimilarity measure

Let $\mathcal{S} := \{(x_1, \ldots, x_n) \in \bigcup_{n\geq 1} \mathbb{R}^n : x_1 \geq x_2 \geq \cdots \geq x_n\}$ be the space of non-increasingly sorted vectors of arbitrary lengths. Moreover, for any $n \in \mathbb{N}$, let $\mathcal{S}_n := \{\mathbf{x} \in \mathcal{S} : |\mathbf{x}| = n\}$ denote the set of all vectors in $\mathcal{S}$ of length exactly n. For example, in a scientometric context, x_i usually represents the number of citations or the impact factor of the i-th most cited paper of a scientist. On the other hand, if we consider the output of the users of the Stack Exchange portal[1], x_i may denote a post (question or answer) rating given by community members. Note that posts' quality is not only quantified by counting the so-called UpVotes ($+1$), but also DownVotes (-1). Thus, the assessment of some answer may be negative.

In [8] some classes of metrics on the space $\mathcal{S}$ were introduced. Here we will focus on a squared version of an Euclidean-like metric from [8] that allows to capture the dissimilarity between two vectors of not necessarily equal lengths. Let $d_M^2 : \mathcal{S} \times \mathcal{S} \to [0, \infty)$ be a function defined for $\mathbf{x} \in \mathcal{S}_{n_x}$, $\mathbf{y} \in \mathcal{S}_{n_y}$ as:

$$\begin{aligned} d_M^2(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{\min\{n_x, n_y\}} (x_i - y_i)^2 + \sum_{i=n_x+1}^{n_y} y_i^2 + \\ &+ \sum_{i=n_y+1}^{n_x} x_i^2 + |n_x - n_y|, \end{aligned}$$

with convention $\sum_{i=u}^{v} \cdots = 0$ for $u > v$. It is easily seen that d_M^2 is a dissimilarity measure.

Note that in Sec. 7 we are going to generalize the above dissimilarity measure. However, all the results derived below will still be valid.

4. Determining the d_M^2-centroid

In order to derive a K-means-like procedure for clustering informetric data, first we have to provide a method for computing the d_M^2-centroid of a set of vectors representing producers. Thus, let $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}\} \subseteq \mathcal{S}$. Our aim is to find

$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu} \in \mathcal{S}} \sum_{i=1}^{l} d_M^2(\mathbf{x}^{(i)}, \boldsymbol{\mu}).$$

[1] See http://stackexchange.com/; Stack Exchange is a network of over one hundred communities created and run by enthusiasts and experts on specific topics like computer science, physics, wine tasting, psychology, etc.

For brevity of notation, let $F(\boldsymbol{\mu}) = \sum_{i=1}^{l} d_M^2(\mathbf{x}^{(i)}, \boldsymbol{\mu})$. Also, let $m = \max\{|\mathbf{x}| : \mathbf{x} \in \mathcal{X}\}$ be the maximal length of a vector in $\mathcal{X}$.

Firstly, note that the length of $\boldsymbol{\mu}$ which is a minimizer of F cannot be greater than m.

Lemma 1. $\left|\arg\min_{\boldsymbol{\mu} \in \mathcal{S}} F(\boldsymbol{\mu})\right| \leq m.$

Proof. Let $\boldsymbol{\mu}' \in \mathcal{S}_n$, $n > m$. Then it holds that:

$$\begin{aligned} &F(\boldsymbol{\mu}') = \\ &= \sum_{j=1}^{l} \Big(\sum_{i=1}^{n_x^{(j)}} (x_i^{(j)} - \mu_i')^2 + \sum_{i=n_x^{(j)}+1}^{n} \mu_i'^2 + |n_x^{(j)} - n| \Big) = \\ &= \sum_{j=1}^{l} \Big(\sum_{i=1}^{n_x^{(j)}} (x_i^{(j)} - \mu_i')^2 + \sum_{i=n_x^{(j)}+1}^{m} \mu_i'^2 \Big) + \\ &+ nl + l \sum_{i=m+1}^{n} \mu_i'^2 - \sum_{j=1}^{l} n_x^{(j)}. \end{aligned}$$

However,

$$\begin{aligned} & F(\mu_1', \ldots, \mu_n') - F(\mu_1', \ldots, \mu_m') \\ = \;& l \sum_{i=m+1}^{n} \mu_i'^2 + nl - ml \\ = \;& l \sum_{i=m+1}^{n} \mu_i'^2 + l(n-m) > 0. \end{aligned}$$

Thus, $F(\mu_1', \ldots, \mu_n') > F(\mu_1', \ldots, \mu_m')$ and $\boldsymbol{\mu}'$ is not a minimizer of F. □

Taking the above into account, our problem may be decomposed as follows. For $n = 1, \ldots, m$ determine:

$$\boldsymbol{\mu}^{(n)} = \arg\min_{\boldsymbol{\mu} \in \mathcal{S}_n} F(\boldsymbol{\mu}) \tag{2}$$

and then compute:

$$\boldsymbol{\mu} = \arg\min_{n=1,\ldots,m} F(\boldsymbol{\mu}^{(n)}). \tag{3}$$

Therefore, we should focus on solving (2), assuming that n is fixed. As F is a sum of convex functions, we of course have that F is a convex function. However, here we deal with a constrained optimization problem, as our search space consists of vectors that are sorted non-increasingly: we have that $\mu_1^{(n)} \geq \mu_2^{(n)} \geq \cdots \geq \mu_n^{(n)}$.

From now on let $[n] := \{1, 2, \ldots, n\}$. Let us introduce the notion of a contiguous partition of an index set $[n]$, that is a set of nonempty, disjoint sets of consecutive elements in $[n]$. Formally, $\mathcal{P} \subseteq 2^{[n]}$ is a contiguous partition of $[n]$, if $\bigcup_{P \in \mathcal{P}} = [n]$, $P \cap P' = \emptyset$, $|P| > 0$, $\{i, j\} \in P$ with $i \leq j$ implies that $i+1, i+2, \ldots, j-1 \in P$ for all $P \neq P'$. The whole class of such contiguous partitions will from

now on be denoted as $\mathcal{CP}([n])$. It might be shown that $|\mathcal{CP}([n])| = 2^{n-1}$. For example, we have:

$$\mathcal{CP}([3]) = \left\{ \begin{array}{c} \{\{1\},\{2\},\{3\}\}, \\ \{\{1,2\},\{3\}\}, \\ \{\{1\},\{2,3\}\}, \\ \{\{1,2,3\}\} \end{array} \right\}.$$

Given $\mathcal{P} \in \mathcal{CP}([n])$ and $i \in [n]$, let $P_{\{i\}}$ stand for an element in $\mathcal{P}$ such that $i \in P_{\{i\}}$. Moreover, let $P^{(i)}$ be the i-th ordered element in $\mathcal{P}$, i.e. such that for $1 \leq i < j \leq |\mathcal{P}|$ it holds $\max P^{(i)} < \min P^{(j)}$. Assuming that $\tilde{x}_i = \sum_{\mathbf{x}:|\mathbf{x}|\geq i} x_i$ we have what follows.

Theorem 1. *Fix $n \in [m]$ and let $\mathcal{P} \in \mathcal{CP}([n])$. Define $\mathbf{y} \in \mathbb{R}^n$ as:*

$$y_i = \frac{1}{l|P_{\{i\}}|} \sum_{j \in P_i} \tilde{x}_j \quad \text{for } i = 1, \ldots, n.$$

If $y_1 \geq y_2 \geq \cdots \geq y_n$ and for all $i \in [n]$ with $i \in (P_{\{i\}} \setminus \{\max P_{\{i\}}\})$ we have

$$\frac{i - \min P_{\{i\}} + 1}{|P_{\{i\}}|} \sum_{j \in P_{\{i\}}} \tilde{x}_j - \sum_{j \in P_{\{i\}}, j \leq i} \tilde{x}_j > 0,$$

then $\mathbf{y}$ is a solution to Eq. (2).

Proof. Our aim is to find

$$\min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{y})$$

with respect to $n-1$ constraints of the form:

$$g_i(\mathbf{y}) \quad = \quad y_{i+1} - y_i \leq 0 \quad \text{for } i = 1, \ldots, n-1.$$

By means of the Karush-Kuhn-Tucker (KKT) theorem (cf. [9]), we need to find $\mathbf{y}$ and $\lambda_1, \ldots, \lambda_{n-1}$ such that

$$\nabla F(\mathbf{y}) + \sum_{i=1}^{n-1} \lambda_i \nabla g_i(\mathbf{y}) = 0,$$

with $\lambda_i g_i(\mathbf{y}) = 0$ and $\lambda_i \geq 0$ for $i \in [n-1]$. Note that for $h \in [n]$ we have:

$$\frac{\partial F}{\partial y_h}(\mathbf{y}) = 2ly_h - 2\tilde{x}_h.$$

For brevity of notation, let us assume that $\lambda_0 := 0$ and $\lambda_n := 0$. Thus, our task reduces to solving the following system of linear equations:

$$\left\{ \begin{array}{rcllll} 0 & = & 2(ly_1 - \tilde{x}_1) & +\lambda_0 & -\lambda_1 \\ 0 & = & 2(ly_2 - \tilde{x}_2) & +\lambda_1 & -\lambda_2 \\ & \vdots & & & \\ 0 & = & 2(ly_{n-1} - \tilde{x}_{n-1}) & +\lambda_{n-2} & -\lambda_{n-1} \\ 0 & = & 2(ly_n - \tilde{x}_n) & +\lambda_{n-1} & -\lambda_n \\ 0 & = & \lambda_1(y_2 - y_1) & & \\ & \vdots & & & \\ 0 & = & \lambda_{n-1}(y_n - y_{n-1}) & & \end{array} \right.$$

under constraints $\lambda_1 \geq 0, \ldots, \lambda_{n-1} \geq 0$ and $y_1 \geq y_2 \geq \cdots \geq y_n$.

Thus, let us consider a solution (not necessarily feasible) that fulfills $\boldsymbol{\lambda} \geq 0$. First of all, let us take u such that $\lambda_{u-1} = \lambda_u = 0$. It immediately implies that:

$$y_u = \frac{1}{l}\tilde{x}_u.$$

On the other hand, for each u and $p \geq 2$ such that $\lambda_{u-1} = 0$, $\lambda_u > 0$, $\lambda_{u+1} > 0$, $\ldots$,$\lambda_{u+p-2} > 0$, $\lambda_{u+p-1} = 0$ we get that $y_u = \cdots = y_{u+p-1}$. More specifically, we have:

$$y_i = \frac{1}{lp} \sum_{j=u}^{u+p-1} \tilde{x}_j \quad \text{for } i = u, \ldots, u+p-1.$$

In such a case, we have that for $i = u, \ldots, u+p-2$:

$$\lambda_i = 2\frac{i-u+1}{p} \sum_{j=u}^{u+p-1} \tilde{x}_j - 2\sum_{j=u}^{i} \tilde{x}_j > 0, \qquad (4)$$

Moreover, since F is convex and $(\forall i)\, g_i$ is an affine function, then by [10] we get that if $\mathbf{y}$ calculated as above fulfills $y_1 \geq y_2 \geq \cdots \geq y_n$, then it is the optimal solution to the task of our interest. □

Remark 1. *Please note that if $\mathbb{I} = [0, \infty]$, i.e. all vectors consist of non-negative numbers, it is easily seen that we have $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_m$. Therefore, in Theorem 1 we have $|P_i| = 1$ for all $i = 1, \ldots, n$, hence $y_i = \frac{1}{l}\tilde{x}_i$ gives the optimal solution to (2).*

Example 1. *Let*

$$\mathcal{X} = \left\{ \begin{array}{lllll} (& 42, & 21, & 12, & 10, & 8 &), \\ (& 1, & 0, & -10 & & &), \\ (& 0, & -1 & & & &), \\ (& -10, & -13 & & & &) \end{array} \right\}.$$

Then the optimal solution to (2) for $n = 5$ is $(8\frac{1}{4}, 4\frac{1}{4}, 1\frac{2}{3}, 1\frac{2}{3}, 1\frac{2}{3})$. We get that by considering $\mathcal{P} = \{\{1\},\{2\},\{3,4,5\}\}$.

Example 2. *Let*

$$\mathcal{X} = \left\{ \begin{array}{lllll} (& -10, & -12, & -14, & -16, & -17 &), \\ (& 1, & 0, & -10 & & &), \\ (& -10, & -15, & -16 & & &), \\ (& -20 & & & & &) \end{array} \right\}.$$

Then the optimal solution to (2) for $n = 5$ is $(-6.95, -6.95, -6.95, -6.95, -6.95)$. This is generated with and $\mathcal{P} = \{\{1,2,3,4,5\}\}$.

5. A fast algorithm to compute the d_M^2-centroid

Theorem 1 induces a simple algorithm to determine $\mu^{(n)} \in \mathcal{S}_n$. One may consider every possible contiguous partition of $[n]$ and then verify if the conditions listed in the theorem are met. Such a routine, being of course mathematically correct, is unfortunately practically unusable.

Therefore, to solve (2), we propose Algorithm 1. The proof of its correctness is presented below.

Data: A set of l vectors $\mathcal{X} \subset \mathcal{S}$ and $n \in \mathbb{N}$.
Result: $\boldsymbol{\mu}^{(n)} = \arg\min_{\boldsymbol{\mu} \in \mathcal{S}_n} F(\boldsymbol{\mu})$.
Let $\tilde{\mathbf{x}}$ be such that $\tilde{x}_i = \sum_{\mathbf{x}:|\mathbf{x}| \geq i} x_i$, $i \in [n]$;
Let $\mathcal{P} = \emptyset$;
Let $\mathbf{y} \in \mathbb{R}^n$;
for $k = 1, 2, \ldots, n$ **do**
  $y_k = \tilde{x}_k / l$;
  Let $\mathcal{P} := \mathcal{P} \cup \big\{\{k\}\big\}$; *(we have $\mathcal{P} \in \mathcal{CP}([k])$)*
  while $|\mathcal{P}| > 1$ *and* $y_{\min P^{(|\mathcal{P}|)}} > y_{\max P^{(|\mathcal{P}|-1)}}$ **do**
    $\mathcal{P} := \Big(\big(\mathcal{P} \setminus \{P^{(|\mathcal{P}|)}\}\big) \setminus \{P^{(|\mathcal{P}|-1)}\}\Big) \cup \{P^{(|\mathcal{P}|-1)} \cup P^{(|\mathcal{P}|)}\}$; *(merge $P^{(|\mathcal{P}|-1)}, P^{(|\mathcal{P}|)}$)*
    for $i \in P^{(|\mathcal{P}|)}$ **do**
      Set $y_i := \frac{1}{l|P^{(|\mathcal{P}|)}|} \sum_{j \in P^{(|\mathcal{P}|)}} \tilde{x}_j$;
    end
  end
end
return $\mathbf{y}$;

Algorithm 1: An algorithm to solve (2).

Theorem 2. *Fix n and let $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}\}$. If $\mathbf{y}$ is the result of applying Algorithm 1, then $\mathbf{y} = \arg\min_{\mathbf{y} \in \mathcal{S}_n(\mathbb{I})} F(\mathbf{y})$.*

Proof. We shall show that for each $k = 1, 2, \ldots, n$ the $(y_1, \ldots, y_k)$ vector at the end of the **for** loop in Algorithm 1 is such that $(y_1, \ldots, y_k) = \arg\min_{\boldsymbol{\mu} \in \mathcal{S}_k} F(\boldsymbol{\mu})$.

First of all, for $k = 1$ we trivially have that $(y_1) = (\tilde{x}_1 / l)$ fulfills the conditions listed in Theorem 1.

Now let us assume that after the $(k-1)$-th iteration, $k \geq 2$, $(y_1, \ldots, y_{k-1}) = \arg\min_{\boldsymbol{\mu} \in \mathcal{S}_{k-1}} F(\boldsymbol{\mu})$. By Theorem 1 this implies that for all $a < b \leq |\mathcal{P}|$:

$$\frac{\sum_{j \in P^{(a)}} \tilde{x}_j}{|P^{(a)}|} \quad \geq \quad \frac{\sum_{j \in P^{(b)}} \tilde{x}_j}{|P^{(b)}|}.$$

What is more, for all $c \in [|\mathcal{P}|]$ and $i = \min P^{(c)}, \min P^{(c)} + 1, \ldots, \max P^{(c)} - 1$:

$$\frac{\sum_{j \in P^{(c)}} \tilde{x}_j}{|P^{(c)}|} \quad > \quad \frac{\sum_{j=\min P^{(c)}}^{i} \tilde{x}_j}{i - \min P^{(c)} + 1}. \tag{5}$$

These conditions imply that for all $a < |\mathcal{P}|$ and $i = \max P^{(a)} + 1, P^{(a)} + 2, \ldots, k-1$ we have:

$$\frac{\sum_{j \in P^{(a)}} \tilde{x}_j}{|P^{(a)}|} \quad \geq \quad \frac{\sum_{j=P^{(a)}+1}^{i} \tilde{x}_j}{i - \max P^{(a)}}. \tag{6}$$

Now let us proceed to the k-th iteration of the algorithm.

We set $\mathcal{P} = \mathcal{P} \cup \{\{k\}\}$. If $\sum_{j \in P^{(|\mathcal{P}|-1)}} \tilde{x}_j \geq (|\mathcal{P}| - 1)\tilde{x}_k$, then for the current $\mathcal{P}$ and $(y_1, \ldots, y_k)$ all the conditions in Theorem 1 are of course met. Thus, assume that we need to merge $\{u, u+1, \ldots, u+p-1\} := P^{(|\mathcal{P}|-1)}$ and $\{u+p, u+p+1, \ldots, u+p+p'-1\} := P^{(|\mathcal{P}|)}$, with $u \geq 1$, $p, p' \geq 1$. Note that surely $u + p + p' - 1 = k$. We shall show that after each such merge for $i = u, u+1, \ldots, u+p+p'-2$ it holds that

$$\frac{i-u+1}{p+p'} \sum_{j=u}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j > 0.$$

By induction, this is sufficient for proving this very theorem, as the algorithm guarantees that $(y_1, \ldots, y_k)$ is non-increasingly sorted at the end of each **for** loop.

First of all, as a merge in $\mathcal{P}$ occurs, we have that

$$\frac{1}{p} \sum_{j=u}^{u+p-1} \tilde{x}_j < \frac{1}{p'} \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j. \tag{7}$$

(a) Let $i = u + p - 1$. Then,

$$\begin{aligned}
&\frac{p}{p+p'} \sum_{j=u}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u}^{u+p-1} \tilde{x}_j \\
=&p \left(\sum_{j=u}^{u+p-1} \tilde{x}_j + \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j \right) - (p+p') \sum_{j=u}^{u+p-1} \tilde{x}_j \\
=&p \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j - p' \sum_{j=u}^{u+p-1} \tilde{x}_j > 0. \quad \textit{(by (7))}
\end{aligned}$$

(b) Let $i = u+1, \ldots, u+p-2$. Then,

$$\begin{aligned}
&\frac{i-u+1}{p+p'} \sum_{j=u}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \\
=&\frac{i-u+1}{p+p'} \Big(\sum_{j=u}^{u+p-1} \tilde{x}_j + \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j \Big) - \sum_{j=u}^{i} \tilde{x}_j \\
=&p \Big(\frac{(i-u+1)}{p} \sum_{j=u}^{u+p-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \Big) \\
&+p' \Big(\frac{(i-u+1)}{p'} \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \Big) \quad \textit{(by (7))} \\
>&p[c_1 (> 0 \ \textit{by (5)})] + p' \Big(\frac{(i-u+1)}{p} \sum_{j=u}^{u+p-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \Big) \\
>&p[c_1 (> 0 \ \textit{by (5)})] + p' [c_2 (> 0 \ \textit{by (5)})] > 0.
\end{aligned}$$

(c) Let $i = u+p, \ldots, u+p+p'-2$. Then,

$$\begin{aligned}
&\frac{i-u+1}{p+p'} \sum_{j=u}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \\
=&\frac{p'}{p+p'} \frac{i-u+1+p-p}{p'} \left(\sum_{j=u}^{u+p-1} \tilde{x}_j + \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j \right) \\
&- \frac{p+p'}{p+p'} \left(\sum_{j=u}^{u+p-1} \tilde{x}_j + \sum_{j=u+p}^{i} \tilde{x}_j \right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{p'}{p+p'} \left(\frac{i-(u+p)+1}{p'} \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j - \sum_{j=u+p}^{i} \tilde{x}_j \right) \\
&+ \frac{1}{p+p'} \left(p \sum_{j=u+p}^{u+p+p'-1} \tilde{x}_j - p' \sum_{j=u}^{u+p-1} \tilde{x}_j \right) \\
&+ \frac{p}{p+p'} \left(\frac{i-u+1}{p} \sum_{j=u}^{u+p-1} \tilde{x}_j - \sum_{j=u}^{i} \tilde{x}_j \right) \\
&= \frac{p'}{p+p'} [c_3 (> 0 \ by \ (5))] + \frac{1}{p+p'} [c_4 (> 0 \ by \ (7))] \\
&+ \frac{p}{p+p'} [c_5 (\geq 0 \ by \ (6))] > 0.
\end{aligned}
$$

Hence, the proof is complete. □

Note that Algorithm 1 may very simply be extended so that it solves (3) directly. An exemplary C++ implementation using Rcpp [11] classes (for use within the R [12] environment for statistical computing) is provided in Fig. 3. We see that at most $O(\min\{m^3, m^2 l\})$-time is required to compute the d_M^2-centroid.

Example 3. *Let us again focus on a data set discussed in Example 1. Here is the output of Algorithm 1 for each n. $n = 5$ gives the optimal solution to* (2).

```
  xtilde= | 8.25 4.25 0.50 2.50 2.00 0.00
- ------- | ------------------------------
n    dist | y1   y2   y3   y4   y5   y6
1 3139.75 | 8.25
2 3063.50 | 8.25 4.25
3 3062.50 | 8.25 4.25 0.50
4 3047.50 | 8.25 4.25 1.50 1.50
5*3034.17*| 8.25 4.25 1.67 1.67 1.67
6 3037.17 | 8.25 4.25 1.67 1.67 1.67 0.00
```

Example 4. *Let us go back to input data from Example 2. Here is the output of Algorithm 1 for each n. Again, $n = 5$ gives the optimal solution to* (2).

```
  xtilde= | -9.750 -6.750 -10.000 -4.000 -4.250
- ------- | ----------------------------------
n    dist |  y1     y2     y3      y4     y5
1 1694.75 | -9.750
2 1528.50 | -8.250 -8.250
3 1126.50 | -8.250 -8.250 -10.000
4 1142.75 | -7.625 -7.625  -7.625 -7.625
5*1108.95*| -6.950 -6.950  -6.950 -6.950 -6.950
```

6. Empirical analysis

Let us now investigate the proposed approach on an informetric data set that has been gathered from the Cross Validated portal[2]. It is one of the sites on the Stack Exchange network. Data consisting of users' posts and ratings (sums of UpVotes and DownVotes) were gathered on 2014-09-15[3]. Basic sample statistics for vectors' lengths (n), maximal values (max), and total sums of ratings (sum) of the 5063 registered users are presented in Table 1.

Table 1: Basic sample statistics for the Cross Validated data set.

	Min.	Median	Mean	Max.
n	1.00	1.00	8.08	1348.00
max	-11.00	2.00	4.00	155.00
sum	-11.00	2.00	7.00	7111.00

Investigation carried out in this section is based on a K-means-like procedure using the d_M^2 dissimilarity measure and clusters centroids computed with Algorithm 1. The procedure was set up to determine six clusters and it starts with random assignments of observations to each cluster. Next, by applying Algorithm 1 iteratively over each cluster, the centroid was recalculated, and clusters' assignment was updated. Basic summary statistics of clusters' centroids are given in Table 2 and those connected to the whole cluster are presented in Table 4.

Table 2: Basic statistics of clusters' centroids: length of $\boldsymbol{\mu}$ (n), minimal (Min), maximal (Max) value, quartiles, and mean element of $\boldsymbol{\mu}$.

C_i	1	2	3	4	5	6
Min	1.59	0.50	0.67	0.48	0.51	0.74
1st Q	1.59	0.76	1.29	0.86	0.97	1.26
Med	1.59	1.27	2.33	1.53	1.92	2.37
Mean	1.59	2.19	6.53	2.83	3.35	4.24
3rd Q	1.59	2.40	5.00	3.16	4.15	4.89
Max	1.59	8.29	68.39	19.55	31.80	79.89
n	1	10	23	35	157	454

Table 3: Cardinalities of obtained clusters.

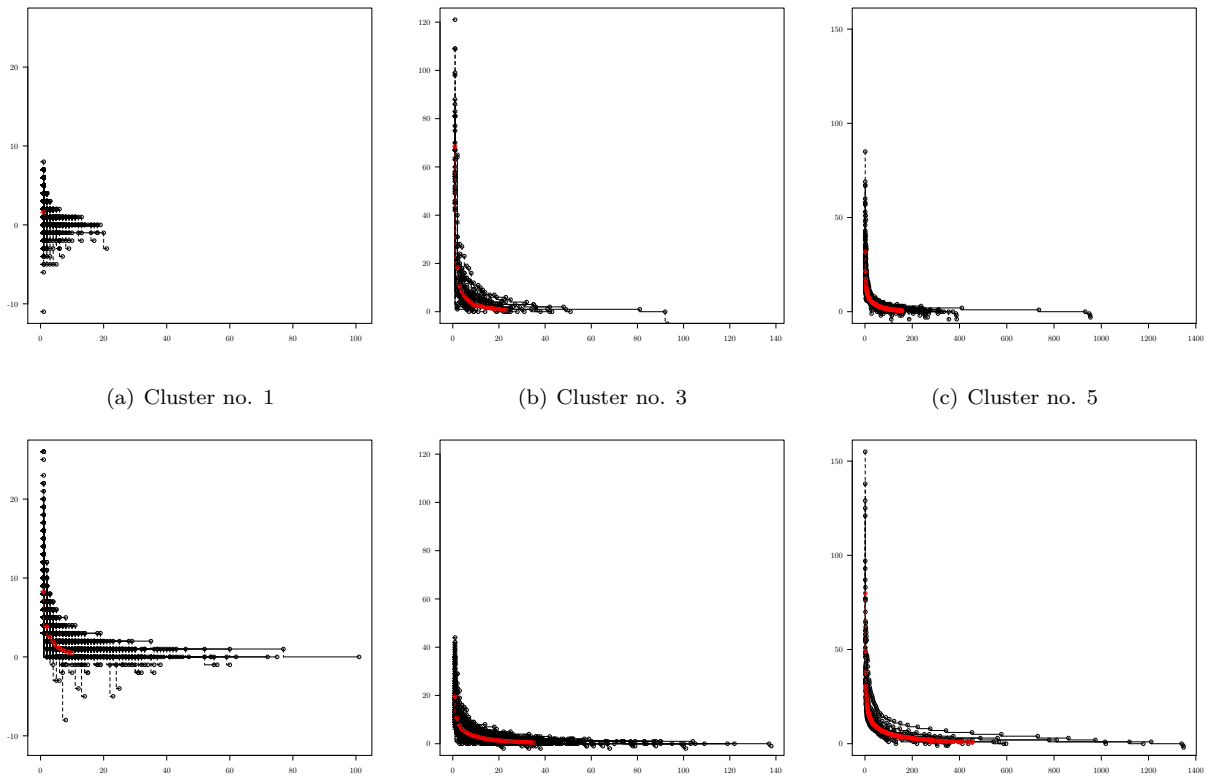
Cluster no.	1	2	3	4	5	6
Size	4060	695	36	188	65	19

Fig. 1 depicts the step functions corresponding to the vectors in each cluster. The centroid is highlighted. On the other hand, Fig. 2 depicts the box plots for the producers' productivity (as measured by n) and quality (as measured by total sums of elements) per each cluster. Please note the logarithmic scale on the Y-axis. Box plots are sorted increasingly with respect to the mean length of vectors in each cluster. Note that in the first cluster the users with low number of posts and ratings were captured (median productivity, n, equals to 1 and median sum of ratings, sum, equals to 1).

We see that by using d_M^2 we may quite easily linguistically describe the aggregated characteristics of vectors in each cluster according to both productivity and quality. Clusters no. 2, 3, and 4 consist of users with low-moderate, moderate, and high-moderate productivity (median n equals 7, 12 and 28.5), respectively, and they may be distinguished by taking into account the sums of ratings of their

[2] See http://stats.stackexchange.com/.

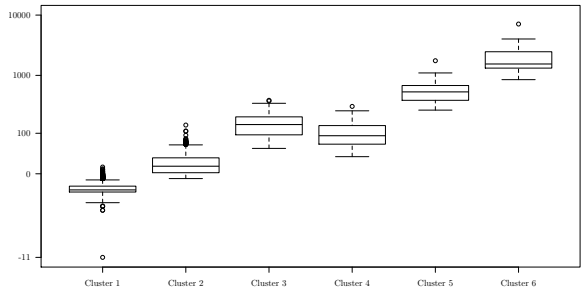
[3] See https://archive.org/details/stackexchange.



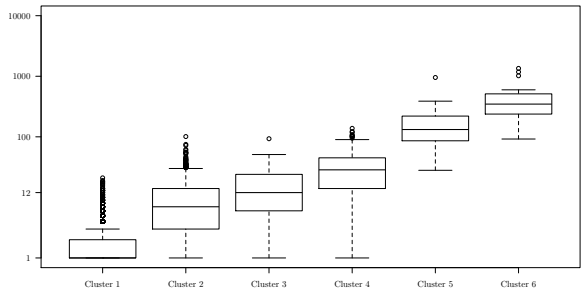
(a) Cluster no. 1 (b) Cluster no. 3 (c) Cluster no. 5

(d) Cluster no. 2 (e) Cluster no. 4 (f) Cluster no. 6

Figure 1: Step functions depicting vectors in each cluster (Cross Validated) and their centroids.



(a) Total sums of elements in each vector.



(b) Lengths of vectors.

Figure 2: Basic sample statistics for vectors in each cluster (Cross Validated).

Table 4: Basic sample statistics for vectors' lengths (n), maximal values (max), and total sums of elements per each cluster.

	Min.	Median	Mean	Max.
Cluster no. 1				
n	1.00	1.00	1.74	21.00
max	-11.00	1.00	1.59	8.00
sum	-11.00	1.00	2.08	19.00
Cluster no. 2				
n	1.00	7.00	10.97	101.00
max	3.00	8.00	8.29	26.00
sum	8.00	20.00	25.12	141.00
Cluster no. 3				
n	1.00	12.00	18.11	93.00
max	42.00	63.50	68.39	121.00
sum	51.00	144.00	157.10	380.00
Cluster no. 4				
n	1.00	28.50	33.33	138.00
max	6.00	17.50	19.55	44.00
sum	34.00	90.00	106.40	299.00
Cluster no. 5				
n	28.00	132.00	165.10	954.00
max	11.00	28.00	31.80	85.00
sum	258.00	528.00	568.90	1754.00
Cluster no. 6				
n	92.00	347.00	450.30	1348.00
max	32.00	60.00	79.89	155.00
sum	848.00	1548.00	2154.00	7111.00

posts (median sum: 20, 144 and 90). In Cluster no. 5 we can find users with quite high productivity (median 132) and quite high quality ($\sim$ 528). Finally, last cluster contains users of very high productivity (median 347) and very high sum of ratings ($\sim$ 1548).

7. Conclusions

This contribution addresses the question of how to determine groups of producers that are similar not only with respect to the quality of information resources they output, but also their quantity. In such a model, the agents are usually represented by real vectors of nonconforming lengths. One of the possible solutions to this issue is the K-means-like algorithm proposed in this paper. The procedure bases on a dissimilarity measure which takes into account the difference of vectors' lengths.

Moreover, please note that the proposed approach is not only restricted to the dissimilarity measure d_M^2 introduced in this paper. Note that the "penalty" for difference of vectors lengths can be easily change. By replacing $|n_x - n_y|$ with ν such that $\nu(|\mathbf{x}|, |\mathbf{y}|) = 0$ iff $|\mathbf{x}| = |\mathbf{y}|$ and $\nu(|\mathbf{x}|, |\mathbf{y}|) > 0$ otherwise (e.g. $\nu(|\mathbf{x}|, |\mathbf{y}|) = p|n_x^r - n_y^r|$, $p > 0$, $r \neq 0$, at it was proposed in [8]), we can control the impact of the difference in the vectors' lengths. In such a way, the procedure may be better calibrated to suit the nature of an input data set we analyze.

There are still many interesting directions worth of deeper investigation. First of all, the proposed approach may be generalized in order to mimic the fuzzy k-means algorithm, which in some applications may be more valid. Secondly, an axiomatic analysis of the d_M^2-centroid (perhaps as an aggregation operator on a special kind of a lattice) can give us a better insight into the derived theoretical results.

On the other hand, please note that in some applicative problems in order to apply clustering techniques on vectors of nonconforming lengths one may try to reduce the data dimension by considering a fixed number of attributes or indicators. This approach discussed in [8] may lead to substantial information loss. However, extensive analysis of such an approach in comparision to proposed in this paper K-means-like procedure may give a better insight into the essence of informatric data.

Acknowledgments

This study was partially supported by the National Science Center, Poland, research project 2014/13/D/HS4/01700.

Anna Cena would like to acknowledge the support by the European Union from resources of the European Social Fund, Project PO KL "Information technologies: Research and their interdisciplinary applications", agreement UDA-POKL.04.01.01-00-051/10-00 via the Interdisciplinary PhD Studies Program.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2013.

[2] B. Gärtner and S. Schönherr. An efficient, exact, and generic quadratic programming solver for geometric optimization. In *Proc. 16th ACM Symposium on Computational Geometry*, pages 110–118, 2000.

[3] H.S. Witsenhausen M.R. Garey, D.S. Johnson. The complexity of the generalized Lloyd-Max problem. *IEEE Transactions on Information Theory*, IT-28(2):255–256, 1982.

[4] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[5] A. Cena and M. Gagolewski. OM3: Ordered maxitive, minitive, and modular aggregation operators – Axiomatic and probabilistic properties in an arity-monotonic setting. *Fuzzy Sets and Systems*, 264:138–159, 2015.

[6] M. Gagolewski and R. Mesiar. Aggregating different paper quality measures with a generalized h-index. *Journal of Informetrics*, 6(4):566–579, 2012.

[7] F. Franceschini and D.A. Maisano. The Hirsch index in manufacturing and quality engineering. *Quality and Reliability Engineering International*, 25:987–995, 2009.

[8] A. Cena, M. Gagolewski, and R. Mesiar. Problems and challenges of information resources producers' clustering, 2015. submitted paper.

[9] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 2006.

[10] D.H. Martin The essence of invexity. *Journal of Optimization Theory and Applications*, (1):65–76, 1985.

[11] D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013.

[12] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. http://www.R-project.org.

```
#include <deque>
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::plugins("cpp11")]]

double dM2_dist(List X, NumericVector y,
                int ny) {
   int l = X.size();
   double dist = 0.0;
   for (int i=0; i<l; ++i) {
      NumericVector x(X[i]);
      int nx=x.size();
      int min_nx_ny=std::min(nx, ny);
      for (int j=0; j<min_nx_ny;  ++j)
         dist+=(x[j]-y[j])*(x[j]-y[j]);
      for (int j=min_nx_ny; j<nx; ++j)
         dist+=x[j]*x[j];
      for (int j=min_nx_ny; j<ny; ++j)
         dist+=y[j]*y[j];
      dist+=abs(nx-ny);
   }
   return dist;
}

// [[Rcpp::export]]
NumericVector dM2_centroid(List X) {
   int l=X.size();
   int m=calc_max_vector_length(X);
   NumericVector xtilde=calc_xtilde(X, m);
                // a linked list (a stack):
   std::deque< std::pair<int, int> > part;
   NumericVector y(m);
   NumericVector best_y=NumericVector(0);
   double best_dist=INFINITY;

   for (int n=1; n<=m; ++n) {
      // C++ arrays use 0-based indices
      part.push_front(
         std::pair<int, int>(n-1, n-1) );
      y[n-1]=xtilde[n-1]/l;

      auto it=part.begin();
      while (it+1!=part.end() &&
               y[(*it).first]
             > y[(*(it+1)).second]) {
         // merge:
         int p1=(*it).second-(*it).first+1;
         int p2=(*(it+1)).second-
                  (*(it+1)).first+1;
         y[(*it).second]=
            (y[(*it).second]*p1+
            y[(*(it+1)).second]*p2)/(p1+p2);
         for (int j=(*it).second-1;
               j>=(*(it+1)).first; --j)
            y[j]=y[(*it).second];
         (*(it+1)).second=(*it).second;
         // erases current it
         // and move forward (pop stack)
         it=part.erase(it);
      }

      double cur_dist=dM2_dist(X, y, n);
      if (cur_dist<best_dist) {
         best_dist=cur_dist;
         best_y=NumericVector(y.begin(),
                              y.begin()+n);
      }
   }
   return best_y;
}
```

Figure 3: A C++ (using Rcpp [11] classes) implementation of Algorithm 1.