CLUSTERING FOR DIFFERENT SCALES OF MEASUREMENT - THE GAP RATIO WEIGHTED K-MEANS ALGORITHM

Joris Guerin, Olivier Gibaru, Stephane Thiery and Eric Nyiri

Laboratoire des Sciences de l'Information et des Systemes (CNRS UMR 7296)
Arts et Metiers ParisTech, Lille, France

ABSTRACT

This paper describes a method for clustering data that are spread out over large regions and which dimensions are on different scales of measurement. Such an algorithm was developed to implement a robotics application consisting in sorting and storing objects in an unsupervised way. The toy dataset used to validate such application consists of Lego bricks of different shapes and colors. The uncontrolled lighting conditions together with the use of RGB color features, respectively involve data with a large spread and different levels of measurement between data dimensions. To overcome the combination of these two characteristics in the data, we use a weighted K-means algorithm which consists in weighting each dimension of the feature space before running K-means. The novelty of this paper lies in the introduction of new weights, relevant for the combination of large spread and different scales. The weight associated with a feature is proportional to the ratio of the biggest gap between two consecutive data points, and the average of all the other gaps. We call this algorithm gap-ratio K-means. This method is compared with two other variants of K-means on the Lego bricks clustering problem as well as two other common classification datasets.

KEYWORDS

Unsupervised Learning, Weighted K-means, Scales of measurement, Robotics application

1. INTRODUCTION

1.1 MOTIVATIONS

In a relatively close future, we are likely to see industrial robots performing tasks on their own. In this perspective, we have developed a smart table cleaning application in which a robot sorts and store objects judiciously among the storage areas available. This clustering application can have different uses: workspaces can be organized before the workday, unsorted goods can be sorted before packaging, ... Even in domestic robotics, such an application, dealing with real objects, can be useful to perform household chores.

As shown in Figure 1, a Kuka LBR iiwa collaborative robot equipped with a camera is presented a table cluttered with unsorted objects. Color and shape features of each object are extracted and

David C. Wyld et al. (Eds) : CCSEIT, AIAP, DMDB, ICBB, CNSA - 2017
pp. 35– 52, 2017. © CS & IT-CSCP 2017 DOI : 10.5121/csit.2017.70605

the algorithm clusters the data in as many classes as there are storage bins. Once objects have been labelled with bin numbers, the robot physically cleans up the table. A better description of the experiment is given in the body of the article. This application was tested with Lego bricks of different colors and shapes (see section 4.2). A link to a demonstration video is given in the caption of Figure 1.

Because such application is meant for ordinary environments, the clustering algorithm needs to be robust to unmastered light conditions, which is synonymous with widely spread datasets. Moreover, the features chosen are on different levels of measurement [1]: RGB-color features are interval type variables whereas lengths are on a ratio scale. Both these specificities of the clustering problem motivated the development of a new weighted K-means algorithm, which is the purpose of this paper.

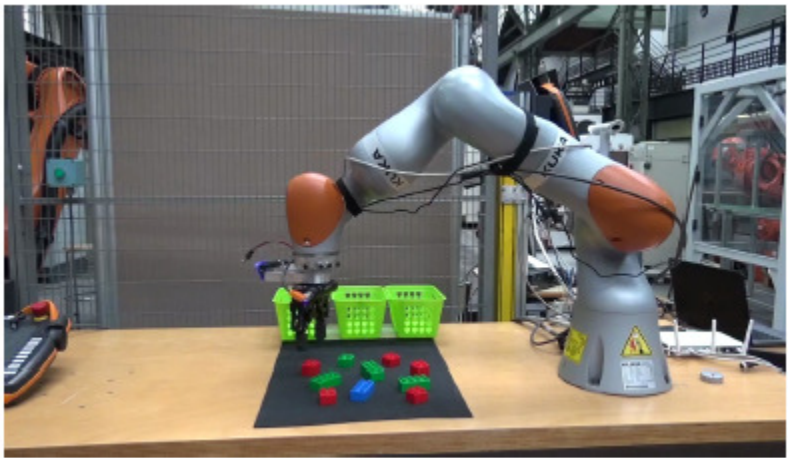

Figure 1: KUKA LBR iiwa performing the Lego bricks sorting application.
Video at : https://www.youtube.com/watch?v=korkcYs1EHM

1.2 INTRODUCTION TO THE CLUSTERING PROBLEM

The table cleaning problem described above boils down to a clustering problem [2, 3]. Clustering is a subfield of Machine Learning also called unsupervised classification. Given an unlabelled dataset, the goal of clustering is to define groups among the entities. Members in one cluster should be as similar as possible to each other and as different as possible to other clusters members. In this paper, we are only concerned with parametric clustering, where the number of clusters is a fixed parameter passed to the algorithm. However, we note that recently, non-parametric Bayesian methods for clustering have been widely studied [4{6].

There are many possible definitions of similarity between data points. The choice of such definition, together with the choice of the metric to optimize, differentiates between the different clustering algorithms. The two surveys [7] and [8], give two slightly different classifications of the various clustering algorithms.

After trying several clustering algorithms on simulated Lego bricks datasets using scikit-learn [9], K-means clustering [10], a partitioning method, appeared efficient for our problem. Therefore, among all clustering methods, this paper focuses on K-means.

1.3. MAIN CONTRIBUTION

For certain unsupervised classification problems on data with large spread, a conventional implementation of K-means algorithm can be inappropriate. Indeed, under certain lighting conditions (not uniform throughout the scene), we observe absurd Lego bricks sorting. In order to increase the clustering robustness, we try to emphasize the influence of certain features using weighted K-means [11]. Weights based on coefficient of variation [12] (cv K-means) are tried first. However, coefficient of variation only makes sense for data measured on a ratio scale, which is not the case for our problem.

In this paper, we propose a new way to define weights in the framework of weighted K-means, which works for both interval and ratio scaled data, we call gap-ratio K-means (gr K-means) the resulting algorithm. In Section 4.2, we show that gr K-means is more robust than conventional K-means and cv K-means for the Lego bricks sorting problem. The approach of exponentiating the weights is also suggested. Introduction of gr K-means algorithm, together with an experimental comparison with K-means and cv K-means on different datasets are the main contributions of this paper.

The article is organized as follows, in Section 2, we briefly describe and derive both the K-means and the cv K-means clustering methods as a baseline to understand the gr K-means algorithm, explained in Sections 3. Between algorithms descriptions, intuitive explanations about why they do not fit our problem are provided, data normalization before clustering is also discussed. In Section 4, we propose an experimental validation of the efficiency of the gr K-means algorithm, first on datasets constituted of Lego bricks, then on two other famous classification datasets. Finally, we conclude this paper by suggesting our recommendations on how to handle a clustering problem using K-means and give pointers to future work directions.

2. PRELIMINARIES

2.1 K-MEANS CLUSTERING

2.1.1 NOTATIONS

All along this paper, we try to respect the following notations. The use of letters i represents indexing on data objects whereas letter j designates the features. Thus,

- $X = \{ x_1, ..., x_i, ..., x_M \}$ represents the dataset to cluster, composed of M data points.
- $F = \{ f_1, ..., f_j, ... , f_N \}$ is the set of N features which characterize each data object.
- x_{ij} stands for the j^{th} feature of object x_i

A data object is represented by a vector in the feature space.

Likewise, the use of letter k represents the different clusters and

- $C = \{ C_1, ..., C_k, ..., C_K \}$ is a set of K clusters.

K-means clustering is based on the concept of centroids. Each cluster C_k, is represented by a cluster center, or centroid, denoted c_k, which is simply a point in the feature space.

We also introduce d, the function used to measure dissimilarity between a data object and a centroid. For K-means, such dissimilarity is quantified with Euclidean distance:

$$d(x_i, c_k) = \sqrt{\sum_{j=1}^{N} (x_{ij} - c_{kj})^2}. \quad (1)$$

2.1.2 DERIVATION

Given a set of cluster centers $c = \{ c_1,\ldots,c_k,\ldots,c_K \}$, cluster membership is defined by

$$x_i \in C_l \iff d(x_i, c_l) \leq d(x_i, c_k),\ \forall k \in \{1, \ldots, K\}. \quad (2)$$

The goal of K-means is to find the set of cluster centers c* which minimizes the sum of dissimilarities between each data object and its closest cluster center.

Introducing the binary variable a_{ik}, which is 1 if x_i belongs to C_k and 0 else, and the membership matrix $A = (a_{ik})_{\substack{i \in \{1, \ldots M\} \\ k \in \{1, \ldots K\}}}$. K-means can be written as an optimization problem:

$$\begin{aligned} \underset{A,\ c}{\text{Minimize}} \quad & \sum_{i=1}^{M} \sum_{k=1}^{K} a_{ik} \times d(x_i, c_k), \\ \text{subject to} \quad & \sum_{k=1}^{K} a_{ik} = 1,\ \forall i \in \{1, \ldots, M\}, \\ & a_{ik} \in \{0, 1\},\ \forall i, \forall k. \end{aligned} \quad (3)$$

In practice, (3) is optimized by solving iteratively two subproblems, one where the set c is fixed and one where A is fixed. The most widely used algorithm to implement K-means clustering is the Lloyd's algorithm [13]. It is based on the method of Alternating Optimization [14], also used in the Expectation-Maximization algorithm [15]. The K-means optimization is composed of two main steps:

- The Expectation step (or E-step) :
  - Initial situation : centroids are fixed (i.e., c is fixed)
  - Action : Each data point in X is associated with a cluster following (2) (i.e., A is computed).

- The Maximization step (or M-step) :
  - Initial situation : Each data object is associated with a given cluster (i.e., A is fixed).
  - Action : For each cluster, the centroid that minimizes the total dissimilarity within the cluster is computed (i.e., c is computed).

When the norm used for dissimilarity is the L^2 norm, which is the case for K-means, it can be shown [10] that the M-step optimization is equivalent to computing the cluster mean:

$$c_k = \frac{1}{\sum_{i=1}^{M} a_{ik}} \sum_{i=1}^{M} a_{ik} \times x_i. \quad (4)$$

2.1.3 CENTROID INITIALIZATION

In order to start iterating between the expectation and maximization steps, initial centroids need to be defined. The choice of such initial cluster centers is crucial and motivates many research, as shown in the survey paper [16]. The idea is to choose the initial centroids among the data points. In our implementation, we use K-means++ algorithm [17] for clusters initialization (see Section 4.1).

2.1.4 DATA NORMALIZATION

In most cases, running K-means algorithm on raw data does not work well. Indeed, features with largest scales are given more importance during dissimilarity calculation and clustering results are biased. To deal with this issue, a common practice is to normalize the data before running the clustering algorithm:

$$x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\sigma_j}, \ \forall i, \forall j \quad (5)$$

where μ_j and σ_j represent respectively the empirical mean and variance of feature f_j.

The made-up, two dimensional toy dataset in Figure 2 illustrates the interest of using data normalization as a preprocessing to K-means. The two natural clusters in Figure 2 present similar mean and variance, but expressed in different units, which makes K-means results completely wrong without normalization.

However, reducing each feature distribution to a Gaussian of variance 1 can involve a loss of valuable information for clustering. Weighted K-means [11] methods can solve such issue. The underlying idea is to capture with weights relevant information about important features. This information is reinjected in the data by multiplying each dimension with the corresponding weight after normalization. In this way, the most relevant features for clustering are enlarged and the others curtailed. In Sections 2.2.4 and 3, we propose two different weighted K-means methods: cv K-means [12] and a new method that we call gap-ratio K-means (gr K-means). These methods differ by the definition of the weights. We compare regular K-means, cv K-means and gr K-means experimentally in Section 4.

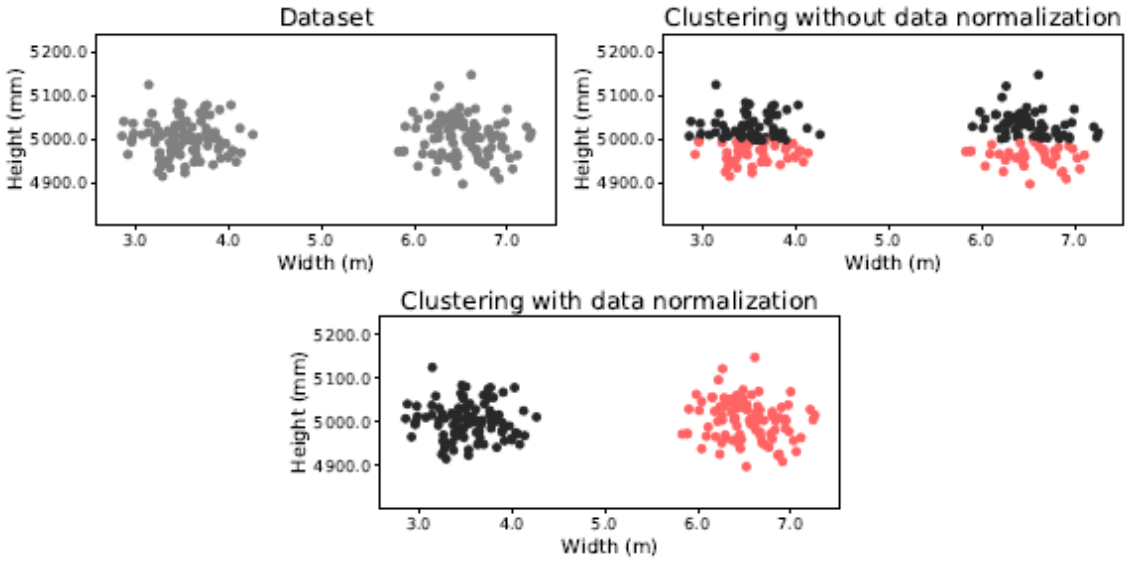

Figure 2: Toy data set to illustrate the need for data normalization before K-means.

2.2 WEIGHTED K-MEANS

2.2.1 ISSUES WITH DATA NORMALIZATION

As explained above, data normalization is often necessary to obtain satisfactory clustering results, but involves a loss of information that can affect the quality of the clusters found. Weighted K-means is based on the idea that information about the data can be captured before normalization and reinjected in the normalized dataset.

2.2.2 WEIGHTED K-MEANS

In a weighted K-means algorithm, weights are attributed to each feature, giving them different importance. Let us call w_j the weight associated with feature f_j. Then, the norm used in the E-step of Weighted K-means is:

$$d(x_i, c_k) = \sqrt{\sum_{j=1}^{N} w_j (x_{ij} - c_{kj})^2} \tag{6}$$

The difference between weighted K-means algorithms lies in the choice of the weights.

2.2.3 EXPONENTIAL WEIGHTED K-MEANS

In this paper, we also propose an extension of weighted K-means that consists in raising the weights to the power of an integer p in the norm formula:

$$d(x_i, c_k) = \sqrt{\sum_{j=1}^{N} w_j^p (x_{ij} - c_{kj})^2} \tag{7}$$

By doing so, we emphasize even more the importance of features with large weights, which makes sense if the information captured by the weights is relevant. In practice, as the weights are between 0 and 1, p should not be too large to avoid considering only one feature. Influence of p in the clustering results is studied in Section 4.

2.2.4 A PARTICULAR EXAMPLE : THE CV K-MEANS ALGORITHM

Weighted K-means based on coefficient of variation (cv K-means) [12] relies on the idea that the variance of a feature is an good indicator of its importance for clustering. Such approach makes sense, indeed, if two objects are of different nature because of a certain feature, the values of this feature come from different distributions, which increases the internal variance of the feature. In this way, the weights used for cv K-means are such that variance information is stored, so that it can be reinjected in the data after normalization.

Hence, the cv weights are derived based on coefficient of variation, also called relative standard deviation. For a one dimensional dataset, it is defined by

$$cv = \frac{\sigma}{\mu} \tag{8}$$

where μ and σ are respectively the mean and standard deviation of the dataset (computed empirically in practice).

Then, coefficients of variation are normalized and weights are computed such that emphasis is placed on features with highest coefficient of variation:

$$w_j = \frac{cv_j}{\sum_{j'=1}^{N} cv_{j'}} \tag{9}$$

cv K-means algorithm follows the same principle as regular K-means, but using norm (6) with weights (9) instead of norm (1).

cv K-means assumes that a feature with high relative variance is more likely to involve objects being of different nature. Such approach works on several datasets but a highly noisy feature might have high variance and thwart cv K-means. However, on the original paper [12], authors test their algorithm on three well-known classification datasets (Iris, Wine and Balance scale) from UCI repository [18] and obtain better results than using regular K-means.

3. GAP RATIO K-MEANS

3.1 INTERVAL SCALE ISSUES

To reason why cv weights do not fit the Lego bricks classification problem lies in the concept of levels of measurement [1]. More specifically, it comes from the difference between ratio scale and interval scale.

Indeed, the notion of coefficient of variation only makes sense for data on a ratio scale and does not have any meaning for data on an interval scale. On an interval scale, it is not relevant to use coefficient of variation because when the mean decreases, the variance does not change

accordingly. Therefore, at equal variance, features closer to zero have higher coefficients of variation for no reason, which biases the clustering process.

In the table cleaning application defined above, the features chosen are colors (RGB) and lengths. RGB-colors are given by three variables, representing the amount of red, green and blue, distributed between 0 and 255. They are on an interval scale and thus should not be weighted using coefficient of variation. This duality in the features measurement scales motivated the development of gap-ratio weights, which is the purpose of this section.

3.2 THE GR-K-MEANS ALGORITHM

The idea behind gap-ratio K-means is fairly simple. When doing clustering, we want to distinguish if different feature values between two objects come from noise or from the fact that objects are of different nature. If we consider that the distribution of a certain feature differs between classes, this feature's values should be more different between objects of different classes than between objects within a class. Gap-ratio weights come from this observation, their goal is to capture this information about the features.

To formulate this concept mathematically, we sort the different values x_{ij} for each feature f_j . Hence, for every j, we create a new data indexing, where integers $i\{j\}$ are defined such that

$$\forall j,\, x_{i\{j\},j} \leq x_{i\{j\}',j} \Leftrightarrow i\{j\} \leq i\{j\}'. \tag{10}$$

Then we define the $i\{j\}^{\text{th}}$ gap of j^{th} feature by:

$$g_{i\{j\},j} = x_{i\{j\}+1,j} - x_{i\{j\},j} \tag{11}$$

Obviously, if there are M data objects, there are M - 1 gaps for each feature.

After computing all the gaps for feature f_j, we define the biggest gap G_j and the average gap μg_j as follows :

$$\begin{aligned} G_j &= \max_{i\{j\}\in\{1,\ldots,M-1\}} g_{i\{j\},j}, \\ \mu g_j &= \frac{1}{N} \sum_{\substack{i\{j\}=1 \\ i\{j\}\neq I\{j\}}}^{M-1} g_{i\{j\},j}, \end{aligned} \tag{12}$$

where $I\{j\}$ is the index corresponding to G_j.

Finally, we define the gap-ratio for feature f_j by :

$$gr_j = \frac{G_j}{\mu g_j}. \tag{13}$$

In other words, for a given feature, the gap ratio is the ratio between the highest gap and the mean of all other gaps. Then, as for cv K-means, gap-ratios are used to compute scaled weights:

$$w_j = \frac{gr_j}{\sum_{j'=1}^{N} gr_{j'}}. \quad (14)$$

The dissimilarity measure for gr K-means is obtained by using weights (14) in (6). Likewise exponential cv K-means, we call exponential gr K-means the algorithm using dissimilarity measure (7) with weights (14).

3.3 INTUITION BEHIND GR K-MEANS

Figure 3 shows a simple two dimensional toy example where using gr weights is more appropriate than cv weights.

In this example, the coefficient of variation along the x-axis is larger than for the y-axis. Indeed, mean values for both dimensions are approximately the same (around 10) whereas variance is higher for the x-axis. Thus, cv K-means focuses on the x-axis despite we can see it is not a good choice just by looking at the plots. The clusters found in the middle plot, together with the weights, confirm the wrong behavior of cv K-means.

However, weights and groups obtained with gr K-means (bottom plot) indicate that the right information is stored in gap-ratio weights for such problem. The biggest gap along the y-axis is a lot bigger than average gaps whereas these two quantities are similar along the x-axis.

4. EXPERIMENTAL VALIDATION

In the previous sections, we have introduced the K-means clustering method. We explained why data normalization is required and why it should not work on data with relatively high spread. Then, we presented cv K-means as a solution to capture important information about the data before normalization but showed that it is not compatible with interval scale data. Finally, we derived a new weighted K-means algorithm that should fit the kind of datasets we are interested in.

In this section, we intend to validate the intuitive reasoning above. To do so, we compare the different weighted K-means algorithms (including regular K-means, with weights $w_i = 1, \forall j,$ and exponentiated weights) on different datasets. First, in Section 4.2, the Lego bricks dataset, used to demonstrate the table cleaning application, is clustered with the different methods. Then, in Section 4.3, two other famous classification datasets are used to investigate further the algorithms behaviors.

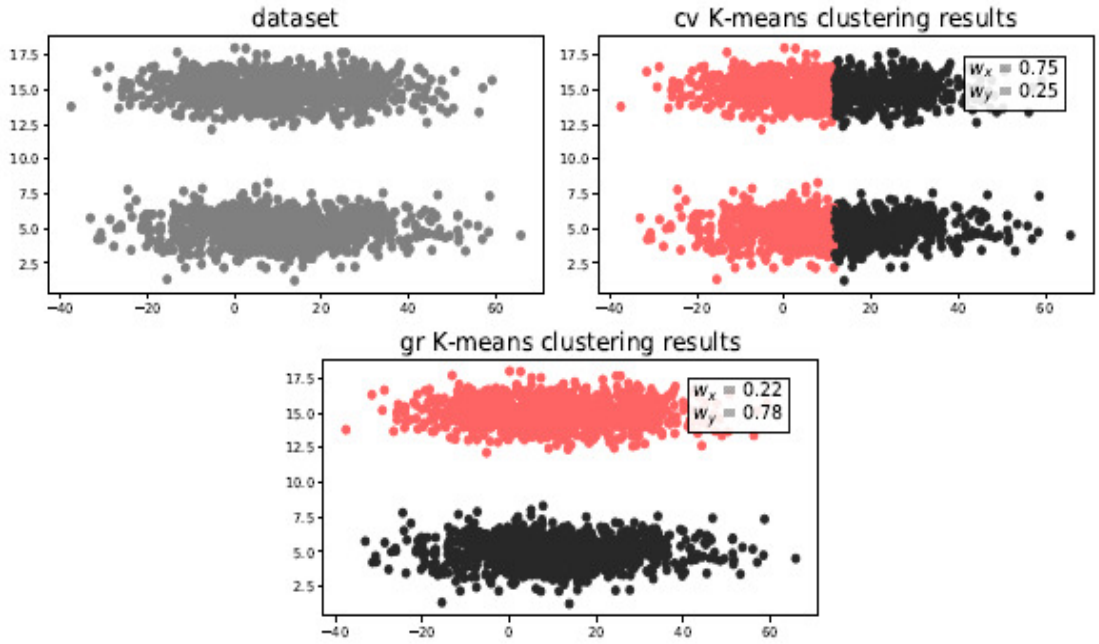

Figure 3: Comparison of cv K-means and gr K-means on a simple made up example. This is to illustrate cases where it seems more logical to deal with gaps rather than variances.

4.1 WEIGHTED K-MEANS IMPLEMENTATION

In this validation section, we used the K-means implementation of scikit-learn [9], an open-source library, as is. This way, our results can be checked and further improvements can be tested easily. To implement weighted K-means algorithms, we also use scikit-learn implementation but on a modified dataset. After normalization, our data are transform using the following feature map:

$$\Phi : x_{ij} \to \sqrt{\omega_j} x_{ij}. \tag{15}$$

As the dissimilarity computation appears only in the E-step, the dataset modifications are equivalent to changing the norm. Indeed, (6) is the same as

$$d(x_i, c_k) = \sqrt{\sum_{j=1}^{N} (\sqrt{w_j} x_{ij} - \sqrt{w_j} c_{kj})^2}. \tag{16}$$

By doing this, results obtained can be compared more reliably. Differences in the results is less likely to come from poor implementation as the K-means implementation used is always the same. Following these steps, implementation is straightforward (Figure 4).

Input:
A data set: $X = \{x_1, ..., x_M\}$,
The number of desired clusters: K.
Method:
1: Compute the weights using (9) or (14).
2: *Raise the weights to some power p:* $w_j \leftarrow w_j^p$.
3: *Data normalization: replace features by standard scores using (5).*
4: Multiply data by the squared root of the weights:
$x_{ij} \leftarrow \sqrt{w_j} x_{ij}\ \forall i, \forall j$.
5: Run K-means on modified data.
6: **return** List of classes for each object.

Figure 4: Implementation of weighted K-means. Steps in italic are optional.

4.2 RESULTS ON THE LEGO CLASSIFICATION PROBLEM

4.2.1. Experiment description

The first dataset used to compare different algorithms is one composed of nine Lego bricks of different sizes and colors, as shown on Figure 5. As explained in the introduction, the original goal was to develop an intelligent robot table cleaning application that can choose where to store objects using clustering. Such application is tested by sorting sets of Lego bricks because it is easy and not subjective to draw natural classes and thus validate the robot choices. Figure 5 shows the kind of data sets we are dealing with, three classes can easily be found within these Lego bricks. Naturally, such set of bricks needs to be sorted within three boxes; the clustering algorithm needs to place the big green, small green and small red bricks in different bins.

Furthermore, on Figure 5, we can see that among the four pictures, lighting varies a lot. Color features observed are really different between two runs of the application. The algorithm needs to be robust to poor lighting conditions and to be able to distinguish between red and green even when colors tend to be saturated (see bottom right image).

A video showing the robot application running can be found at https://www.youtube.com/watch?v=korkcYs1EHM. Three different cases are illustrated: the one of Figure 5, one with a different object (not a Lego brick), and one with four natural classes with only three boxes.

The experiment goes as follows: the robot sees all the objects to cluster and extract three color features (RGB) and two length features (length and width). Colors are extracted by averaging a square of pixels around the center of the brick. The dataset gathered is then passed to all variants of weighted K-means algorithms which return the classes assigned of each object. Finally, the robot physically sorts the objects (see video). For our experimental comparison of different algorithms, the experiment has been repeated 98 times with different arrangements of the Lego bricks presented on Figure 5, and with different lighting conditions. For each trial, if the algorithm misclassified one or more bricks, it is counted as a failure, else, it is a success.

As explained in Section 4.2.2, clustering is tried with different weighted K-means algorithms but also with and without scaling. Different values of p for exponentiated weighted K-means are also tried.

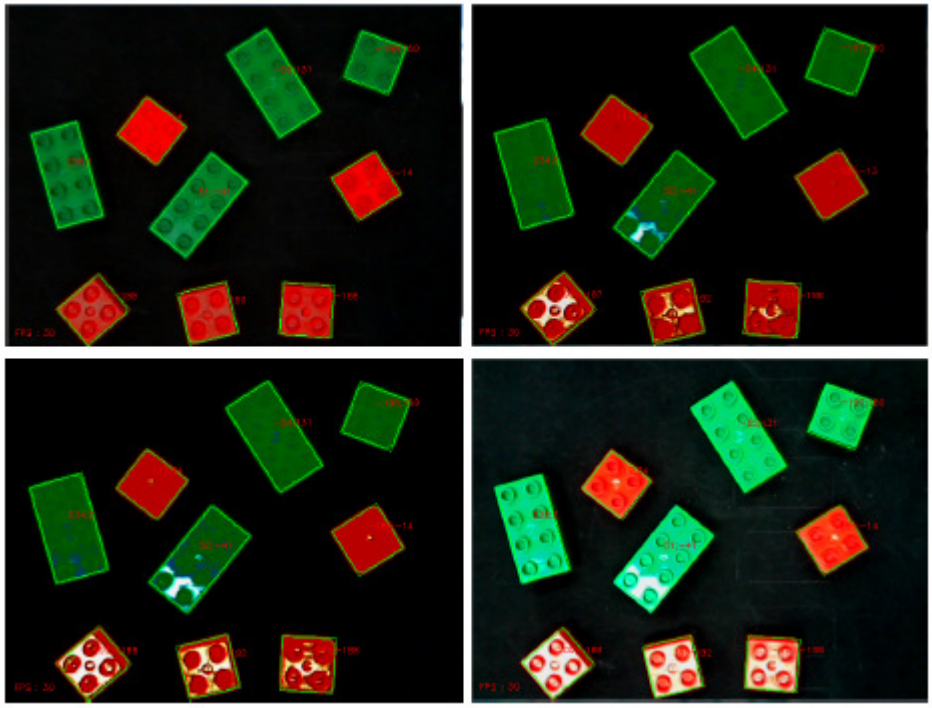

Figure 5: Data set to cluster under different light conditions.

Figure 6 presents results obtained on the 98 trialsLego bricks datasets. The two left charts represent results on the original datasets and the right ones are results on the same dataset with a slight color modification. We removed 50 to the blue component of the bricks, which corresponds to using bricks of slightly different colors, in order to test the robustness of the algorithms.

4.2.2. Results Interpretation

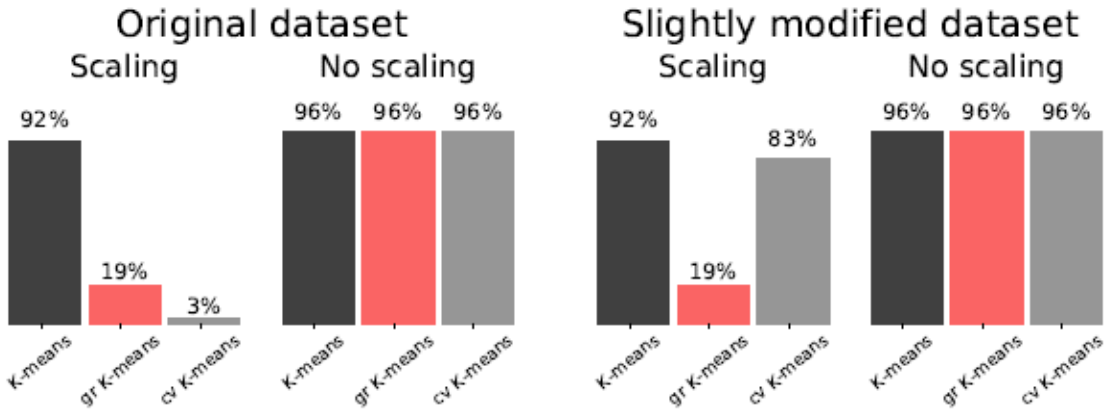

Figure 6: Percentage of experiments with at least one misclassification. The experiment was run 98 times under different lightning conditions and with different layouts of the bricks. Error rates presented are averaged among these 98 runs.

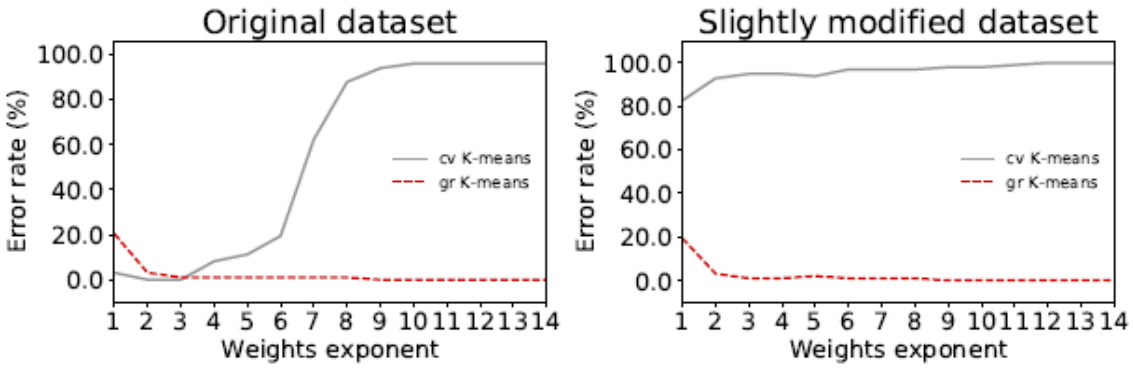

Figure 7: Exponent influence for Lego bricks clustering with exponentiated weighted K-means algorithms.

We start by analyzing the influence of scaling the dataset. As we can see on Figure 6, without scaling (right column), error rates are all very large, around 95%. For this precise problem, K-means cannot perform good without a proper scaling of the dataset before running the clustering algorithm. Such observation makes sense as lengths ($\approx$ 5cm) cannot be compared with colors ($\approx$ 150) because they are totally differently scaled. K-means always put emphasis on data with the largest values (i.e., colors), which means that noise on the color has much more influence than different values of the length. Hence, for this practical example, we can assert that data scaling is required to have a decent classification.

Then, let us compare the different algorithm. The first observation is that whatever preprocessing is used, regular K-means always results in very poor classification. One possible explanation for such bad behavior is the relatively high spread in the data (due to lighting conditions), which involve high variance in the features values and makes it difficult to differentiate between noise and true difference of nature. For this reason, emphasizing certain features is required and we can know compare cv and gr K-means.

Looking at the left subplots of Figure 6, we can see that cv K-means performs particularly well on the original dataset. However, after little modifications, it falls into very bad behavior. Such issue with cv K-means comes from the fact that coefficient of variation is not appropriate for interval scale data. In other words, cv K-means can succeed on the original dataset only because the bricks used do not present RGB components too close to zero. On the other hand, gr K-means performs reasonably well ($\approx$ 20% error rates), and is stable to dataset modifications.

Another way to determine the relevance of the information stored in the weights is to look at different values of the exponent for exponentiated weighted K-means. Figure 7 shows such curves for both cv weights and gr weights with the original and the modified datasets. For gr weights, curves for both datasets are superimposable, gr K-means algorithm is insensitive to average values of the interval scaled features. As for cv K-means, it performs good under certain conditions (left figure) but is not robust to decreasing the mean value of one feature (bottom figure).

Figure 7 left plot shows another interesting thing. For low value of p, cv K-means performs better than gr K-means (0% error rate for $p = 3$). Even if all we want is a certain exponent for which error is low, it is interesting to note that high exponents involve bad clustering results with cv weights. Such behavior shows that the weights are not so relevant because if they are given

too much importance, clustering gets worse. On the other hand, with exponentiated gr K-means error rate tends to decrease when the exponent increases. Information carried by gr weights is good for such clustering problem and should be given more importance. Error rate falls to zero at p = 9 and remains stable to exponent increases until relatively high values of p (> 20); the balance between important components is well respected within the weights. For this kind of datasets, characterized by large spread, mixed scales of measurement and relatively independent features, exponentiated gr K-means with relatively high exponent seems to be a good solution for clustering.

4.3 GENERALIZATION TO OTHER DATA SETS

Gap-ratio weighted K-means was developed with Lego bricks classification task in mind, so it is not surprising that it performs good on such datasets. Now, we test this algorithm on other classification datasets of different nature to see how well it generalizes. Different weighted K-means methods are compared on two famous supervised learning datasets, so that we have labels to evaluate the clustering output. The two datasets chosen are the Fisher Iris dataset [19] and the Wine dataset, both taken from the UCI Machine Learning Repository [18]. Table 1 gives some important characteristics of both datasets.

Table 1. Datasets descriptions

Dataset	Iris	Wine
Number of instances	150	178
Number of attributes	4	13
Number of classes	3	3
Is linearly separable?	No	Yes
Data type	Real	Real and Integers
Scale of measurement	Ratio	Ratio

Table 2 summarizes clustering results for both datasets, using all previously described implementations of different algorithms. For each configuration, we ran the algorithm 1000 times with different random initializations. The percentages reported in Table 2 corresponds to the average clustering accuracy over the different runs. For completeness, we also report the average NMI (Normalized Mutual Information) scores. We note that NMI scores do not appear for the Lego bricks evaluation because in the application, we are interested in having zero error.

We acknowledge that data normalization increases accuracy for the Wine dataset but not for the Fisher Iris dataset. We explain such results by the fact that the values of the four Iris attributes are of the same order of magnitude. Hence, normalizing involves a loss of information that is not compensated by scaling the different features.

Regarding the algorithms efficiency, For the Iris dataset, both gr and cv K-means implementations are better than regular K-means. Moreover, increasing the weights exponent improves the quality of the clustering. This means that both gap-ratio and coefficient of variation weights are able to capture the important information for clustering. However, for the Wine dataset, the best option is to stick to regular K-means.

Finally, we also underline that for both K-means and cv K-means, we do not find the same results than in the original paper of cv K-means ([12]). Overall we obtain higher accuracy. this might come from the K-means++ initialization, as in [12], all the points are initialized at random.

Table 2: Results on other data sets.

IRIS DATASET

	Scaling		No scaling	
	Accuracy	NMI	Accuracy	NMI
K-means	82.94 %	0.65	89.33 %	0.75
gr K-means	88.27 %	0.72	91.33 %	0.81
cv K-means	95.84 %	0.85	94.24 %	0.84
gr^2 K-means	95.76 %	0.86	94.00 %	0.82
cv^2 K-means	95.99 %	0.87	95.33 %	0.85

WINE DATASET

	Scaling		No scaling	
	Accuracy	NMI	Accuracy	NMI
K-means	96.63 %	0.88	70.22 %	0.43
gr K-means	94.99 %	0.84	70.22 %	0.43
cv K-means	92.99 %	0.79	70.22 %	0.43
gr^2 K-means	85.83 %	0.63	70.22 %	0.43
cv^2 K-means	87.13 %	0.67	70.22 %	0.43

Accuracy and NMI scores averaged over 1000 runs of the algorithms from different centroid initializations. gr^2 and cv^2 denote the exponential versions of the algorithms ($p = 2$).

In the conclusion, we propose a short recommendation section to help the reader selecting a weighted K-means algorithm given the properties of the dataset to cluster.

5. CONCLUSION

5.1 RECOMMENDATIONS

Preprocessing the data by normalizing the features seems to be a good idea as long as the initial dimensions present different scales. On the other hand, if features already have the same order of magnitude, it is better to leave them unchanged, unless important information is already captured, with properly chosen weights for example.

As for the choice of the algorithm, from what we have observed, we suggest to stick to regular K-means when your data appears to have high correlation and clusters do not come from only a few dimensions. This is more likely to happen with high dimensional data. In contrast, on relatively low dimensional data, it seems a smart idea to go for a weighted K-means algorithm. If patterns are to be found along isolated dimensions, gap-ratio seems to be a better indicator than coefficient of variation. However, for certain cases, such as Iris dataset, we acknowledged that cv K-means produces similar results. For data on different scales of measurement, cv K-means cannot be used and gap-ratio is the right choice; especially with wide spread data.

Finally, regarding weights exponentiation, we found out that for linearly separable datasets, when weighted K-means makes improvements, it is better to raise the weights to a relatively high power. Information gathered in the weights is good and should be emphasized. However, the

exponent should not be too large or the algorithm ends up considering a single feature. We should remain careful to avoid loosing the multidimensionality of the problem.

The algorithm developed is a new approach for clustering data that are mixed between interval and ratio measurement scales and should be considered whenever facing such case. However, as for all other clustering problems, it works only for a certain range of problems and should not be used blindly.

5.2 FUTURE WORK

gr K-means - Regarding the gr K-means algorithm, we have several possibility of improvement in mind. First, combining data orthogonalization methods (such as ICA [20]) and gap-ratio indicator seems a promising idea and it might be fruitful to search in this direction. Indeed, gr weights are computed along different dimensions of the feature space and if features have strong correlation, gaps might disappear and variance might be spread along several dimensions. For this reason, it seems appealing to try to decorrelate data using orthogonalization methods.

It could also be interesting to consider not only the largest gap along one dimension but also the next ones, according to the number of different classes desired. Indeed if within three classes the two separations come from the same features, even more importance should be given to this set of features. Some modifications of the equations in Section 3 should enable to try such approach.

Automatic feature extraction - Regarding the table cleaning application which motivated this research, developing gr K-means enabled us to get the robot sorting judiciously Lego bricks, as well as other objects (see video). However, clustering is based on carefully selected features which are only valid for a range of objects. As a future research direction, we consider trying to develop the same application with automatic feature extraction, using transfer learning from a deep convolutional network trained on a large set of images [21].

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Harsh Gazula and Pr. Hamed Sari-Sarraf for their constructive reviews of this paper**.**

REFERENCES

[1] S. S. Stevens, "On the theory of scales of measurement," 1946.

[2] S. Theodoridis and K. Koutroumbas, "Chapter 11-clustering: Basic concepts," Pattern Recognition, pp. 595-625, 2006.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, "Unsupervised learning and clustering," Pattern classification, pp. 519-598, 2001.

[4] Z. Ghahramani, "Bayesian non-parametrics and the probabilistic approach to modelling,"Phil. Trans. R. Soc. A, vol. 371, no. 1984, p. 20110553, 2013.

[5] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," Journal of Mathematical Psychology, vol. 56, no. 1, pp. 1-12, 2012.

[6] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," Journal of the ACM (JACM), vol. 57, no. 2, p. 7, 2010.

[7] R. Xu and D. Wunsch, "Survey of clustering algorithms," IEEE Transactions on neural networks, vol. 16, no. 3, pp. 645-678, 2005.

[8] P. Berkhin, "A survey of clustering data mining techniques," in Grouping multidimensional data. Springer, 2006, pp. 25-71.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R.Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[10] S. Theodoridis and K. Koutroumbas, "Chapter 14-clustering algorithms iii: Schemes based on function optimization," Pattern Recognition, pp. 701-763, 2006.

[11] X. Chen, W. Yin, P. Tu, and H. Zhang, "Weighted k-means algorithm based text clustering," in Information Engineering and Electronic Commerce, 2009. IEEC'09. International Symposium on. IEEE, 2009, pp. 51-55.

[12] S. Ren and A. Fan, "K-means clustering algorithm based on coefficient of variation," in Image and Signal Processing (CISP), 2011 4th International Congress on, vol. 4. IEEE, 2011, pp.2076-2079.

[13] S. Lloyd, "Least squares quantization in pcm," IEEE transactions on information theory, vol. 28, no. 2, pp. 129-137, 1982.

[14] J. C. Bezdek and R. J. Hathaway, Some Notes on Alternating Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 288-300. [Online]. Available: http://dx.doi.org/10.1007/3-540-45631-7 39

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," Journal of the royal statistical society. Series B (methodological), pp.1-38, 1977.

[16] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," Expert Systems with Applications, vol. 40, no. 1, pp. 200-210, 2013.

[17] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007, pp. 1027-1035.

[18] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[19] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annals of eugenics, vol. 7, no. 2, pp. 179-188, 1936.

[20] E. Oja and A. Hyvarinen, "A fast fixed-point algorithm for independent component analysis," Neural computation, vol. 9, no. 7, pp. 1483-1492, 1997.

[21] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition." in ICML, 2014, pp.647-655.

AUTHORS

Joris Guerin received the diplome d'ingenieur (equivalent to M.Sc. degree) from Arts et Metiers ParisTech and the M.Sc. in Industrial Engineering from Texas Tech University, both in 2015. He is currently a Ph.D student at Laboratoire des Sciences de l'Information et des Systemes (LSIS), at Arts et Metiers ParisTech, Lille, France. His current research focuses on Clustering and Reinforcement Learning for Robotics manipulation

Olivier Gibaru is currently full professor at the Department of Mathematics and Computer Science at ENSAM, Lille campus. He obtained his PhD in applied mathematics in 1997. His main research interests includes: applied mathematics, estimation for robotic applications, geometry, control engineering and high precision mechanical systems. He is the coordinator of the EU Horizon 2020 ColRobot project www.colrobot.eu. He is an active member of the SMAI-SIGMA group which is a national learned society dedicated to Applied Mathematics for the Industrial Applications.

Stephane Thiery received the Ph.D degree in Automatics from University of Nice-Sophia Antipolis, France, in 2008. He was a post-doctoral fellow in the NON-A team in INRIA-Lille, for eight months in 2009-2010, and joined Arts et Metiers ParisTech Engineering School, as assistant professor in Applied Mathematics and Automatics, in 2010. His current research includes machine learning, real-time parameters estimation, and control of mechanical systems.

Eric Nyiri received the Ph.D degree in Computer Science from University of Lille I, France, in 1994. He joined Arts etMetiers ParisTech Engineering School, as an assistant professor in Applied Mathematics and Computer Science, in 1995. His initial research domain was L1 interpolation and approximation. In 2010, he joined the LSIS Lab and his current research includes machine learning and path planning for robots. Since 2016, he is a member of the COLROBOT European project.