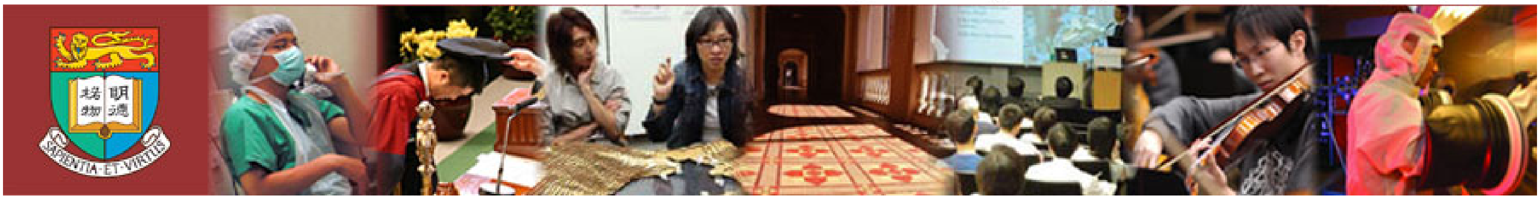

Reducing UK-means to K-means

S. D. Lee Ben Kao
Department of Computer Science
The University of Hong Kong
sdlee,kao@cs.hku.hk

Reynold Cheng
Department of Computing
Hong Kong Polytechnic University
csckcheng@comp.polyu.edu.hk

Abstract

This paper proposes an optimisation to the UK-means algorithm, which generalises the k-means algorithm to handle objects whose locations are uncertain. The location of each object is described by a probability density function (pdf). The UK-means algorithm needs to compute expected distances (EDs) between each object and the cluster representatives. The evaluation of ED from first principles is very costly operation, because the pdf's are different and arbitrary. But UK-means needs to evaluate a lot of EDs. This is a major performance burden of the algorithm. In this paper, we derive a formula for evaluating EDs efficiently. This tremendously reduces the execution time of UK-means, as demonstrated by our preliminary experiments. We also illustrate that this optimised formula effectively reduces the UK-means problem to the traditional clustering algorithm addressed by the k-means algorithm.

1. Introduction

Clustering of uncertain data has recently attracted interests from researchers. This is driven by the need of applying clustering techniques to data that are uncertain in nature, and a lack of clustering algorithms that can cope with the uncertainty. Uncertainty in data arises naturally due to random errors in physical measurements, data staling, as well as defects in the data collection models. For instance, when track locations with GPS devices, the reported location can have errors of a few metres. When attempting to cluster the location of objects tracked using GPS, the errors may affect the clustering result.

Traditional clustering approaches model objects as having accurately known positions. This model does not cope well with uncertain data. It does not take into account the uncertainty inherent in the data, and may lead to undesirable clustering results because information on the uncertainty is dropped.[1] Owing to this shortcoming as well as the practical need to deal with data with uncertainty, there has been growing interest in developing problem models and algorithms to handle uncertain data.[1, 5, 3, 4]

In this paper, we study the problem of clustering objects with multi-dimensional uncertainty. We are given a set of objects, for which we do not know the locations accurately. Rather than a single point in space, the location of each object is represented by a probability density function (pdf) over the space R^m being studied. Given a set of such objects, we want to divide them into k clusters, minimising the total expected distance (ED) from the objects to their cluster centres. We focus on the case where ED is defined using MSE (mean squared error). [1]

This problem was first proposed in [1], in which the UK-means algorithm was proposed to solve it. This is a generalisation of the traditional k-means algorithm to handle objects with uncertain locations. The major computational cost of the algorithm is the evaluation of the EDs, which involves numerical integration using a large amount of sample points for the pdf's. To improve the efficiency of UK-means, [5] introduced some pruning techniques to avoid many ED computations. The pruning techniques make use of the bounding boxes as well as the triangle inequality to establish lower- and upper-bounds of the ED. Using these bounds, some candidate assignments of clusters to objects are eliminated, and hence the corresponding ED calculations can be avoided. But these techniques do not reduce the cost of each ED computation. Further, the pruning effectiveness is not guaranteed, as it depends on the distribution of data.

In this paper, we aim at boosting the performance of UK-means not by pruning, but by deriving a simple formula for ED computations so that the cost of these computations can be tremendously reduced. The inspiration comes from the calculation of the *moment of inertia* of rigid bodies in the field of mechanics.[2] Calculating the moment of inertia of an object based on the definition is costly. However, there exist theorems to simplify such computations. We use similar ideas to simplify ED computations. Upon further analysis, we realize that applying this optimisation to UK-means effectively reduces it to k-means. This means that it is pos-

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.40

sible to perform clustering on uncertain data using the traditional k-means algorithm, producing the same clustering results as UK-means.

The rest of the paper is organised as follows. We will mention a few related works in Section 2. In Section 3, we formally define the problem. In Section 4, we derive an efficient formula for ED computations. Preliminary experiments supporting our claim are presented in Section 5. In Section 6, we discuss about some restrictions of this new technique and we conclude this paper in Section 7.

2. Related Works

The UK-means algorithm was first proposed in [1] for solving the problem that we are studying. This is an extension of the well known k-means algorithm to handle the uncertain data. Like k-means, UK-means is an iterative algorithm. Initially, the cluster centres are assigned arbitrary points in the vector space R^m. (These arbitrary points may be randomly generated, or heuristically determined using domain knowledge.) Then following two steps are repeated until the solution converges. In step 1, each object in question is assigned to the cluster whose centre is the closest to that object. In step 2, each cluster centre is recalculated as the centre of mass of the objects assigned to it.

The major differences between UK-means and k-means is in step 1. Traditionally, k-means determines the closest cluster to an object based on a distance function $d(\vec{x}, \vec{y})$ (where $\vec{x}, \vec{y} \in R^m$). UK-means uses instead the expected distance $\mathrm{ED}(o, \vec{y})$ defined between an uncertain object o and any point $\vec{y} \in R^m$. ED is defined as an integral involving $d(\vec{x}, \vec{y})$ and the probability density function, which can be arbitrary, for the object in question. Evaluating the integral is computationally costly, as it involves many computations of $d(\vec{x}, \vec{y})$ (for many sample points $\vec{x}$ for the object). Further, UK-means has to evaluate ED for every possible pair of objects and clusters in *each* iteration. The cost of ED computation thus creates a performance bottleneck for the UK-means algorithm.

Since the computation cost of UK-means is mainly contributed by the great amount of ED calculations, pruning techniques [5] called MinMax have been proposed to reduce the number of ED calculations. These work by using fast methods to find upper bounds and lower bounds of the ED between an object and a cluster. The idea is that if the lower bound of $\mathrm{ED}(o_i, \vec{c}_p)$ (i.e. expected distance between object o_i and cluster centre $\vec{c}_p$) is strictly greater than the upper bound of $\mathrm{ED}(o_i, \vec{c}_q)$, then we are sure that $\mathrm{ED}(o_i, \vec{c}_p) > \mathrm{ED}(o_i, \vec{c}_q)$. Since step 1 of UK-means selects the closest cluster to o_i, cluster p will not be selected. We can thus prune away cluster p from consideration for o_i. With the help of the bounds, there is no need to evaluate $\mathrm{ED}(o_i, \vec{c}_p)$, thus saving the cost of its computation. The more candidates that can be pruned this way, the more time savings can be achieved. [5] proposes various ways to find bounds on ED quickly. These include using the minimum bounding box of an object, and using the triangle inequality. In this paper, we are not proposing new pruning techniques for UK-means. Rather, we speed up step 1 by optimising the ED computation.

In mechanics, the *moment of inertia* of a rigid body about an axis of rotation is an interesting quantity for studying its rotational motion. This quantity appears in the formulae for angular moment, relation between torque and angular acceleration, as well as rotational kinetic energy.[2] In particular, for a planar object O, its moment of inertia $I_{\vec{y}}$ about an axis, which passes through point $\vec{y}$ and is perpendicular to the object, is defined as:

$$I_{\vec{y}} = \int_{\vec{x} \in O} \|\vec{x} - \vec{y}\|^2 \rho(\vec{x})\, d\vec{x} \qquad (1)$$

where $\rho(\vec{x})$ is the density of the object at the point $\vec{x}$, and $\|\cdot\|$ is the Euclidean norm. Computing the moment of inertia using this formula is tedious. So, physicists have derived, among others, the *parallel axis theorem*, which says that

$$I_{\vec{y}} = I_{\vec{k}} + M \left\|\vec{y} - \vec{k}\right\|^2 \qquad (2)$$

where $\vec{k}$ is the centre of gravity of O and M is its mass. So, once $I_{\vec{k}}$ has been pre-computed, we can determine $I_{\vec{y}}$ efficiently for any $\vec{y} \in R^2$. (As a further optimisation, tables containing formulae for $I_{\vec{k}}$ of standard shapes have been published. One can thus avoid performing integration completely when computing the moment of inertia about *any* axis when considering objects of standard shapes or a combination of them.) Since Equation (1) resembles Equation (6) below, similar techniques can be developed for computing ED efficient.

3. Definitions

Consider a set of objects $O = \{o_1, \ldots, o_n\}$ in m-dimensional space R^m with a distance function $d : R^m \times R^m \rightarrow R$ giving the distance $d(\vec{x}, \vec{y}) \geq 0$ between any points $\vec{x}, \vec{y} \in R^m$. Associated with each object is a pdf $f_i : R^m \rightarrow R$, which gives the probability density of o_i at each point $\vec{x} \in R^m$. By the definition of pdf, we have (for all $i = 1, \ldots, n$)

$$\begin{aligned} f_i(\vec{x}) &\geq 0 \quad \forall \vec{x} \in R^m \\ \int_{\vec{x} \in R^m} f_i(\vec{x})\, d\vec{x} &= 1 \end{aligned} \qquad (3)$$

We define the expected distance between object o_i and any point $\vec{y} \in R^m$ as:

$$\mathrm{ED}(o_i, \vec{y}) = \int_{\vec{x} \in R^m} d(\vec{x}, \vec{y}) f_i(\vec{x})\, d\vec{x} \qquad (4)$$

In this paper, we follow a suggestion in [1] and consider only the mean squared error (MSE) for ED. This means that we consider the case where $d(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\|^2 = (\vec{x}-\vec{y})\cdot(\vec{x}-\vec{y})$ (based on the square of Euclidean norm). This is different from [5], which uses the Euclidean norm, i.e. $d(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\|$.

Now, given an integer constant k, the problem of clustering uncertain data is to find a set of cluster representative points $C = \{\vec{c}_1,\ldots,\vec{c}_k\}$ and a mapping $h : \{1,\ldots,n\} \to \{1,\ldots,k\}$ so that the objective function—total expected distance (TED)—is minimised:

$$\mathrm{TED} = \sum_{i=1}^{n} \mathrm{ED}(o_i, \vec{c}_{h(i)}) \tag{5}$$

4. Computing the Expected Distance

Since we are considering Euclidean distance, we can rewrite Equation (4) as:

$$\mathrm{ED}(o_i,\vec{y}) \quad = \quad \int_{\vec{x}\in R^m} \|\vec{x}-\vec{y}\|^2 f_i(\vec{x})\,d\vec{x} \tag{6}$$

i.e. ED is actually the second moment. Note its resemblance to Equation (1) for the moment of inertia. This suggests that we may be able to find a simple formula for ED resembling Equation (2), similar to the parallel axis theorem. The derivations should be similar, too.

4.1. A simple formula for Expected Distance

First of all, for each uncertain object o_i, define the centroid $\vec{k}_i$ as:

$$\vec{k}_i \quad = \quad \int_{\vec{x}\in R^m} f_i(\vec{x})\,\vec{x}\,d\vec{x} \tag{7}$$

which is a vector in R^m. Now, we ask: What is the difference between $\mathrm{ED}(o_i,\vec{k}_i)$ and $\mathrm{ED}(o_i,\vec{y})$, for any arbitrary $\vec{y}\in R^m$?

$$\begin{aligned}
&\mathrm{ED}(o_i,\vec{y}) - \mathrm{ED}(o_i,\vec{k}_i)\\
=\;& \left(\int_{\vec{x}\in R^m} \|\vec{x}-\vec{y}\|^2 f_i(\vec{x})\,d\vec{x}\right)\\
&- \left(\int_{\vec{x}\in R^m} \left\|\vec{x}-\vec{k}_i\right\|^2 f_i(\vec{x})\,d\vec{x}\right)\\
=\;& \int_{\vec{x}\in R^m} \left(\|\vec{x}-\vec{y}\|^2 - \left\|\vec{x}-\vec{k}_i\right\|^2\right) f_i(\vec{x})\,d\vec{x}
\end{aligned} \tag{8}$$

Now, note that

$$\begin{aligned}
&\|\vec{x}-\vec{y}\|^2\\
=\;& \left\|(\vec{x}-\vec{k}_i)-(\vec{y}-\vec{k}_i)\right\|^2\\
=\;& \left[(\vec{x}-\vec{k}_i)-(\vec{y}-\vec{k}_i)\right]\cdot\left[(\vec{x}-\vec{k}_i)-(\vec{y}-\vec{k}_i)\right]\\
=\;& (\vec{x}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i) + (\vec{y}-\vec{k}_i)\cdot(\vec{y}-\vec{k}_i)\\
&-2(\vec{y}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i)\\
=\;& \left\|\vec{x}-\vec{k}_i\right\|^2 + \left\|\vec{y}-\vec{k}_i\right\|^2 - 2(\vec{y}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i)
\end{aligned} \tag{9}$$

Equation (9) is actually the cosine rule for triangles, expressed in vector form (and extended to R^m). Rewriting it as:

$$\|\vec{x}-\vec{y}\|^2 - \left\|\vec{x}-\vec{k}_i\right\|^2 = \left\|\vec{y}-\vec{k}_i\right\|^2 - 2(\vec{y}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i)$$

and substituting this into Equation (8), we get:

$$\begin{aligned}
&\mathrm{ED}(o_i,\vec{y}) - \mathrm{ED}(o_i,\vec{k}_i)\\
=\;& \int_{\vec{x}\in R^m} \left(\left\|\vec{y}-\vec{k}_i\right\|^2 - 2(\vec{y}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i)\right) f_i(\vec{x})\,d\vec{x}\\
=\;& \left\|\vec{y}-\vec{k}_i\right\|^2 \int_{\vec{x}\in R^m} f_i(\vec{x})\,d\vec{x}\\
&-2\int_{\vec{x}\in R^m} ((\vec{y}-\vec{k}_i)\cdot(\vec{x}-\vec{k}_i)) f_i(\vec{x})\,d\vec{x}\\
=\;& \left\|\vec{y}-\vec{k}_i\right\|^2 - 2(\vec{y}-\vec{k}_i)\cdot\left(\int_{\vec{x}\in R^m} f_i(\vec{x})(\vec{x}-\vec{k}_i)\,d\vec{x}\right)
\end{aligned} \tag{10}$$

by virtue of Equation (3). Let us concentrate on the integral:

$$\begin{aligned}
&\int_{\vec{x}\in R^m} f_i(\vec{x})(\vec{x}-\vec{k}_i)\,d\vec{x}\\
=\;& \int_{\vec{x}\in R^m} f_i(\vec{x})\vec{x}\,d\vec{x} - \vec{k}_i \int_{\vec{x}\in R^m} f_i(\vec{x})\,d\vec{x}\\
=\;& \vec{k}_i - \vec{k}_i\\
=\;& \vec{0}
\end{aligned} \tag{11}$$

because the first integral equals $\vec{k}_i$ by Equation (7) whereas the second integral is equal to unity because of Equation (3).

Now, substituting Equation (11) back into Equation (10), we get: $\mathrm{ED}(o_i,\vec{y}) - \mathrm{ED}(o_i,\vec{k}_i) = \left\|\vec{y}-\vec{k}_i\right\|^2$. Rearranging the terms, we get an analogue of the parallel axis theorem (Equation (2)):

Theorem 1 *For any uncertain object o_i and any point $\vec{y}\in R^m$,*

$$\mathit{ED}(o_i,\vec{y}) = \mathit{ED}(o_i,\vec{k}_i) + \left\|\vec{y}-\vec{k}_i\right\|^2 \tag{12}$$

Therefore, once we have pre-computed, $\vec{k}_i$ and ED$(o_i, \vec{k}_i)$, we can find any ED$(o_i, \vec{y})$ by simply adding $\left\|\vec{y} - \vec{k}_i\right\|^2 = d(\vec{y}, \vec{k}_i)$ to ED$(o_i, \vec{k}_i)$, without the need of evaluating the probability density function f_i again. Since $d(\vec{x}, \vec{y})$ is relatively cheap to compute (when compared to evaluating ED from first principles), this formula allows us to compute ED efficiently. This result holds for any object o_i.

Besides facilitating computation, Equation (12) also tells us that ED$(o_i, \vec{y})$ attains its minimum value at the centroid (i.e. when $\vec{y} = \vec{k}_i$), and is not bounded from above. Note that unlike the parallel-axis theorem (and similar theorems for the moment of inertia), which is applicable only in 3D space, Theorem 1 holds for any Euclidean space R^m where m is an integer.

Now, we can modify the UK-means algorithm to cluster the objects o_i easily. First of all, we calculate once-for-all the values of $\vec{k}_i$ and ED$(o_i, \vec{k}_i)$ for all $i = 1, \ldots, n$. Then, we proceed with UK-means, but calculate all ED$(o_i, \vec{y})$ using Equation (12) instead of Equation (4). In this way, we can calculate ED$(o_i, \vec{y})$ efficiently, without even the need to access the pdf f_i of each object. In practice, the pdf is usually represented as a set of sample points with probabilities stored in file system or database. Using Equation (12) thus saves the cost of accessing such information.

4.2. Reduction to k-means Clustering

Applying Theorem 1 to the definition (Equation (5)) of our objective function, we get:

$$\begin{aligned}
\text{TED} &= \sum_{i=1}^{n} \text{ED}(o_i, \vec{c}_{h(i)}) \\
&= \sum_{i=1}^{n} \left(\text{ED}(o_i, \vec{k}_i) + \left\| \vec{c}_{h(i)} - \vec{k}_i \right\|^2 \right) \\
&= \left(\sum_{i=1}^{n} \text{ED}(o_i, \vec{k}_i) \right) + \left(\sum_{i=1}^{n} \left\| \vec{c}_{h(i)} - \vec{k}_i \right\|^2 \right)
\end{aligned}$$

Note that the first term above is a constant, because our set of objects o_i and their pdf's f_i are fixed. Therefore, the solution of the clustering problem would be the same if we use the following adjusted objective function F instead of TED:

$$F = \text{TED} - \sum_{i=1}^{n} \text{ED}(o_i, \vec{k}_i) = \sum_{i=1}^{n} \left\| \vec{c}_{h(i)} - \vec{k}_i \right\|^2$$

The advantage of using F instead of TED as the objective function is that we no longer need to consider ED$(o_i, \vec{k}_i)$ at all. All we need as input are the locations of the object centroids $\vec{k}_i$.

What does this mean? If we use F as the objective function, then we are minimising the total squared distance from the cluster centres to the object centroids. This is equivalent to running k-means to cluster n points in R^m whose locations coincide with the object centroids $\vec{k}_i$. Therefore, we have reduced the problem of clustering uncertain objects to the classical problem of clustering point objects (with accurately known locations). We can thus reuse the classical k-means algorithm, by using the centroids $\vec{k}_i$ of the original objects o_i as the input points. All the nice properties of the k-means algorithm apply, e. g. convergence and all sorts of optimisations that have been studied in the literature.

So, we have a new method of solving the problem of clustering uncertain objects. First, we pre-process the data to compute the centroids $\vec{k}_i$ of each object o_i. Then, we feed these $\vec{k}_i$ to the classical k-means algorithm and get back the clustering result. We call this the CK-means algorithm. From the arguments above, we should expect CK-means to give identical clustering results (i.e. the assignment function h and the cluster centres c_j) as UK-means when given the same input objects o_i and the same number of clusters k. Furthermore, if both algorithms are given the same set of initial cluster centres c_j, they will give the same intermediate results h and c_j at the end of each iteration.

5. Experiments

Both the UK-means and CK-means algorithms have been implemented in C++. The programs are compiled using GCC 4.1.2 with optimisations turned on. They are run on a Pentium-4 2.26GHz machine with 512MB of main memory running Linux kernel 2.6.20.7.

Following [1], we generated data sets with n uncertain objects in two dimensional space. For each uncertain object, a point is generated randomly in $[0, 100] \times [0, 100]$. A bounding box of side length $d = 10.0$ centred at this point is then considered. A total of $s = 1000$ sample points are randomly generated inside the bounding box, each associated with a probability value, a uniformly distributed random value between 0 and 1. These s probability values are then normalised to make the sum equal to 1. So, each object is associated with s sample points representing a pdf. Next, k random points are generated in $[0, 100] \times [0, 100]$. These are used as the seeding clustering representatives for both UK-means and CK-means.

Many sets of data are generated by varying n and k. Each set of data generated in this way is then given to the algorithms UK-means and CK-means to do clustering. The CPU time taken by each algorithm is measured. The I/O time taken is roughly the same for both algorithms, and are very small in value. So, it is not shown in the following pages.

An important observation, which verifies our theoretical results above, is that for any data set, both algorithms finishes in the same number of iterations. In addition, all intermediate results (assignment of clusters to objects as well as the cluster centres computed) are identical between both algorithms. This confirms the validity of Theorem 1 and the correctness of our implementation.

5.1. Effect of number of objects

The first set of experiments is performed with a data sets with varying number of objects (n) and a fixed number of clusters $k = 20$. The results are shown in Figure 1. The curves for UK-means and CK-means show the CPU-time taken by the two algorithms. Note that these are plotted in logarithmic scale, and the values should be read using the scale on the left edge of the figure. The time for CK-means already includes the time taken for pre-computing the centroids. However, for reference, we have also plotted the time taken for pre-computing the centroids separately in the figure. In addition, we also plotted the number of iterations taken. But this curve is plotted using linear scale, and its values should be read off from the right edge of the figure.

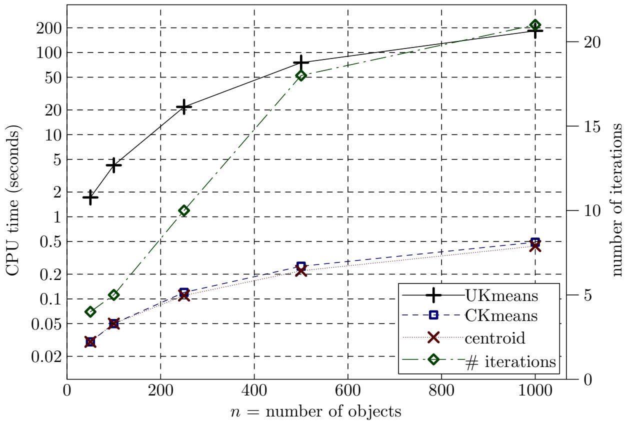

Figure 1. Effects of n

The results show that CK-means is 57–375 times faster than UK-means. The higher the value of n, the bigger the speed up is. The speedup would be even bigger, had we assumed that the centroids were computed offline and hence deducted the time spent on the computation of the centroids from the time taken by CK-means. The cost savings of using CK-means is obvious, because ED computations are entirely eliminated. CK-means actually spends most of its time on the pre-computation of the centroids, which involves evaluating the integral in Equation (7) using the sample points of each object . Once the centroids are computed, CK-means takes little extra time to perform the clustering. Note that the amount of time spent on centroid computations is relatively small, when compared to the time taken by UK-means. It is 1.7% for $n = 50$. This percentages decrease as n increases, because the centroids are computed only once, and then reused in all iterations. As n increases, this percentage decreases and reaches as low as 0.24% at $n = 1000$ in our experiments. As the number of iterations in general increases with n, the cost of centroid computations per object per iteration decreases with n.

5.2. Effect of number of clusters

In the next set of experiments, we fixed the number of objects $n = 1000$ and varied k, the number of clusters to find. The results are shown in Figure 2. CK-means is 19.9–796 times faster than UK-means. The time taken for centroid computations is roughly 0.45s for all these cases. This is because the number of objects is fixed. So, the number of centroid calculations does not vary. The general trend observed is that as k increases, the number of iterations needed also increases, and the savings of CK-means increases, because per-iteration cost of centroid computations decreases.

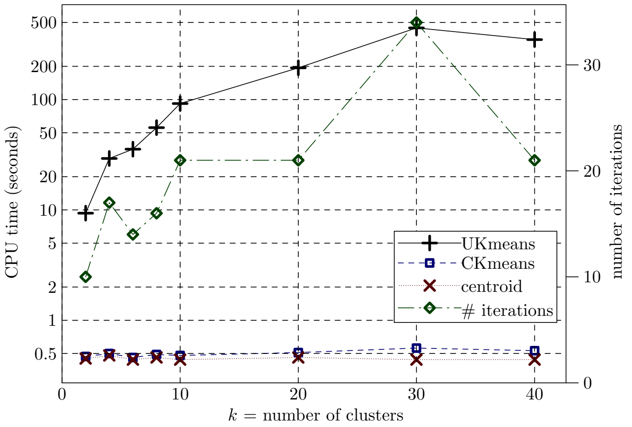

Figure 2. Effects of k

5.3. Effect of number of dimensions

We have repeated the above experiments using 5 dimensions instead of 2. Again, we first fixed k and varied n from 50 to 1000. Then, we fixed n and varied k from 2 to 40. The results are similar to the experiments performed in 2D space: CK-means consistently outperforms UK-means significantly, saving 96.6%–99.8% of CPU time. This means CK-means is 29.8–605 times faster than UK-means. The time taken for centroid computations in CK-means is always around 0.5ms per object. The number of dimensions has little effect on the percentage improvement of CK-means over UK-means.

5.4. Effect of number of samples

Finally, we studied the effects of the number of samples by varying s from 1000 to 5000 and keeping $n = 1000$, $k = 20$. We used 2-dimensional data in this set of experiments. The results are plotted in Figure 3. From the graph, we can see that as s increases, both UK-means and CK-means take more time to finish. This is because a larger s means a higher cost of evaluating the integral in Equation (4). Since UK-means performs many ED calculations, a larger s causes it to take more time. As for CK-means, we note that the only step that is affected by s is the computation of the centroids according to Equation (7). The more the number of samples per object, the higher the cost of centroid computations are. This can be confirmed from the curve for centroid calculations in this graph. CK-means consistently and significantly outperforms UK-means. It is 306–527 times faster than UK-means.

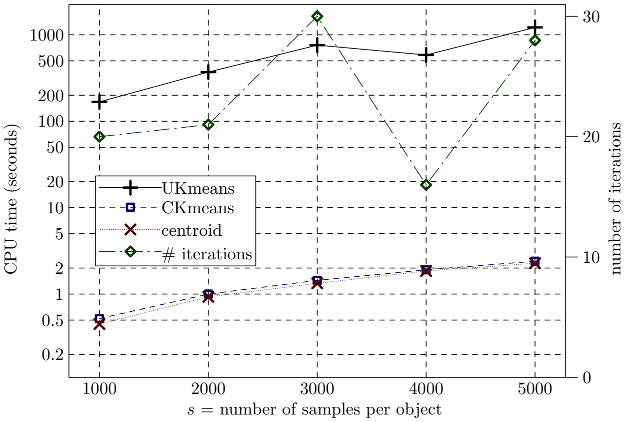

Figure 3. Effects of s

6. Discussions

Theorem 1 allows us to evaluate ED in an efficient manner. Further it also holds for vector spaces of any finite number of dimensions. The only restrictions are that

1. we use the mean squared error (MSE) for the definition of ED; and
2. the distance function is based on Euclidean norm.

While these restrictions are strong, we argue that in most cases in practice, these restrictions are satisfied. First of all, the MSE is widely used in other algorithms, too, because it is easier to compute than mean distance, and gives results similar to using mean distance. As for the distance being based on Euclidean norm, we note that this is the desirable norm to use when considering spatial problems, because the Euclidean norm $\|\cdot\|_2$ is rotation-invariant. Many other norms, such as Manhattan norm $\|\cdot\|_1$ or maximum norm $\|\cdot\|_\infty$, are not rotation-invariant. Norms that change upon rotation may be useful in some problem settings, but for spatial data, we argue that Euclidean norm is the most desirable and widely used norm.

7. Conclusion

This paper studies the clustering of uncertain data in m-dimensional Euclidean space. We have derived a simple formula for computing expected distances efficiently. This formula has been applied to the UK-means algorithms and empirically shown to reduce its computation time by over 95%. This corresponds to a speed up of at least 20 times.

The problem of clustering a set of n objects with only a probabilistic knowledge of their locations in m-dimensional Euclidean space has been successfully reduced to the clustering of a set of n points. These n points are the centroid, or mean position, of those n objects.

In the future, we would investigate on relaxing the restrictions discussed in Section 6. It would be interesting if we can extend these results to non-Euclidean norms, as well as using mean distance for the definition of ED.

References

[1] M. Chau, R. Cheng, B. Kao, and J. Ng. Uncertain data mining: An example in clustering location data. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, volume 3918 of *Lecture Notes in Computer Science*, pages 199–204, Singapore, 9–12 Apr. 2006. Springer.

[2] R. D. Gregory. *Classical Mechanics: An undergraduate text*. Cambridge University Press, 2006. ISBN: 0521826780.

[3] H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 672–677, Chicago, Illinois, USA, 21–24 Aug. 2005. ACM.

[4] H.-P. Kriegel and M. Pfeifle. Hierarchical density-based clustering of uncertain data. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 689–692, Houston, Texas, USA, 27–30 Nov. 2005. IEEE Computer Society.

[5] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip. Efficient clustering of uncertain data. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pages 436–445, Hong Kong, China, 18–22 Dec. 2006. IEEE Computer Society.

[6] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium Advances in Spatial Databases, (SSD'99)*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132, Hong Kong, China, 20–23 July 1999. Springer.