symmetry

MDPI

Article

A Fast K-prototypes Algorithm Using Partial Distance Computation

Byoungwook Kim

Creative Informatics & Computing Institute, Korea University, Seoul 02841, Korea; byoungwook.kim@inc.korea.ac.kr; Tel.: +82-2-3290-1674

Academic Editor: Doo-Soon Park
Received: 6 April 2017; Accepted: 18 April 2017; Published: 21 April 2017

Abstract: The k-means is one of the most popular and widely used clustering algorithm; however, it is limited to numerical data only. The k-prototypes algorithm is an algorithm famous for dealing with both numerical and categorical data. However, there have been no studies to accelerate it. In this paper, we propose a new, fast k-prototypes algorithm that provides the same answers as those of the original k-prototypes algorithm. The proposed algorithm avoids distance computations using partial distance computation. Our k-prototypes algorithm finds minimum distance without distance computations of all attributes between an object and a cluster center, which allows it to reduce time complexity. A partial distance computation uses a fact that a value of the maximum difference between two categorical attributes is 1 during distance computations. If data objects have m categorical attributes, the maximum difference of categorical attributes between an object and a cluster center is m. Our algorithm first computes distance with numerical attributes only. If a difference of the minimum distance and the second smallest with numerical attributes is higher than m, we can find the minimum distance between an object and a cluster center without distance computations of categorical attributes. The experimental results show that the computational performance of the proposed k-prototypes algorithm is superior to the original k-prototypes algorithm in our dataset.

Keywords: clustering algorithm; k-prototypes algorithm; partial distance computation

1. Introduction

K-means algorithm is one of the simplest clustering algorithm as unsupervised learning, so that is very widely used [1]. As it is a partitioning-based clustering method in cluster analysis, a dataset is partitioned into several groups according to a similarity measure as a distance to the average of a group. A K-means algorithm minimizes the objective function known as squared error function iteratively by finding a new set of cluster centers. In each iteration, the value of the objective function becomes lower. In a k-means algorithm, the objective function is defined by the sum of square distances between an object and a cluster center.

The purpose of using k-means is to find clusters that minimize the sum of square distances between each cluster center and all objects in each cluster. Even though the number of clusters is small, the problem of finding an optimal k-means algorithm solution is NP-hard [2,3]. For this reason, a k-means algorithm adapts heuristics and finds local minimum as approximate optimal solutions. The time complexity of a k-means algorithm is $O(i*k*n*d)$, where i iterations, k centers, and n points in d dimensions.

K-means algorithm spends a lot of processing time for computing the distances between each of the k cluster centers and the n objects. So far, many researchers have worked on accelerating k-means algorithms by avoiding unnecessary distance computations between an object and cluster centers. Because objects usually remain in the same clusters after a certain number of iterations, much of the repetitive distance computation is unnecessary. So far, a number of studies on accelerating k-means algorithms to avoid unnecessary distance calculations have been carried out [4–6].

The K-means algorithm is efficient for clustering large datasets, but it only works on numerical data. However, the real-world data is a mixture of both numerical and categorical features, so a k-means algorithm has a limitation of applying cluster analysis. To overcome this problem, several algorithms have been developed to cluster large datasets that contain both numerical and categorical values, and one well-known algorithm is the k-prototypes algorithm by Huang [7]. The time complexity of the k-prototypes algorithm is also $O(i{*}k{*}n{*}d)$: one of the k-means. In case of large datasets, time cost of distance calculation between all data objects and the centers is high. To the best of our knowledge, however, there have been no studies that reduce the time complexity of the k-prototypes algorithm.

Recently, big data has become a big issue for academia and various industries. Therefore, there is a growing interest in the technology to process big data quickly from the viewpoint of computer science. The research on the fast processing of big data in clustering has been limited to numerical data. However, big data deals with numerical data as well as categorical data. Because numerical data and categorical data are processed differently, it is difficult to improve the performance of clustering algorithms that deal with categorical data in the existing way of improving performance.

In this paper, we propose a fast k-prototypes algorithm for mixed data (FKPT). The FKPT reduces distance calculation using partial distance computation. The contributions of this study are summarized as follows.

1. Reduction: computational cost is reduced without an additional data structure and memory spaces.
2. Simplicity: it is simple to implement because it does not require a complex data structure.
3. Convergence: it can be applied to other fast k-means algorithms to compute the distance between each cluster center and an object for numerical attributes.
4. Speed: it is faster than the conventional k-prototypes.

This study presents a new method of accelerating k-prototypes algorithm using partial distance computation by avoiding unnecessary distance computations between an object and cluster centers. As a result, we believe the algorithm proposed in this paper will become the algorithm of choice for fast k-prototypes clustering.

The organization of the rest of this paper is as follows. In Section 2, various methods of accelerating k-means and traditional k-prototypes algorithm are described, for the proposed k-prototypes are defined. A fast k-prototypes algorithm proposed in this paper is explained and its time complexity is analyzed in Section 3. In Section 4, experimental results demonstrate the scalability and effectiveness of the FKPT using partial distance computation by comparison with traditional k-prototypes algorithm. Section 5 concludes the paper.

2. Related Works

In this section, we briefly describe various methods of accelerating k-means and traditional k-prototypes algorithm.

2.1. K-means

The k-means is one of the most popular clustering algorithm due to its simplicity and scalability for large datasets. The k-means algorithm is to partition n data objects into k clusters while minimizing the Euclidean distance between each data object and the cluster center it belongs to [1]. The fundamental concept of k-means clustering is as follows.

1. It chooses k cluster centers in some manner. The final result of the algorithm is sensitive to the initial selection of k initial centers, and many efficient initialization methods have been proposed to calculate better final k centers.

2. The k-means repeats the process of assigning individual objects to their nearest centers and updating each k center as the average of a value of object's vector assigned to the centers until no further changes occur on the k centers.

K-means algorithm spend most of the time computing distance between an object and current cluster centers. However, much of these distance computations are unnecessary, because objects usually remain in the same clusters after a few iterations [6]. Thus, k-means is popular and easy to implement, but it is wasting processing time on redundant and unnecessary distance computations.

The reason why the k-means are inefficient is because, in each iteration, all objects must identify the closest center. In one iteration, all nk distance computations is needed between the n objects and the k centers. After the end of one iteration, the centers are changed and the nk distance computations occur again in the next iteration.

Pelleg and Moore (1999) and Kanungo et al. (2002) adapted a k–d tree to store datasets for accelerating k-means [8,9]. These algorithms are effective for large datasets, but not effective for high (d > 10) dimensions. For low dimensional data, indexing the large data to be clustered is an effective way for fast k-means. These results can be explained as the curse of dimensionality, namely a distance between points and centers tend to be far from one another. Therefore, the performance of pruning is degraded for many dimensions. These algorithms must consider the overhead costs of constructing the k–d tree. A time complexity of constructing the k–d tree is $O(n\log(n))$ for n data points.

One way to accelerate algorithms is to search using partial distance [10,11] in a processing to identify the closest points. The reason for calculating distance is to identify the closest center point from an object point. Therefore, we do not need to exact distances if we can confirm the minimum distance. In general, the distance calculation is the sum of the square for all point attributes. If we calculate $\|x - c\|^2$, the distance between a point x and a center c can be calculated by summing squared distances in each dimension. In a distance calculation between point x and another center c′, if the sum exceeds $\|x - c\|^2$, the distance $\|x - c'\|^2$ cannot be the minimum distance, so the distance calculation stops before all attribute calculations. The cost of a partial distance search is usually effective in high dimension.

Our proposed idea of this study was inspired by this partial distance search [10,11]. We extend this partial distance method to the pruning technique of k-prototypes algorithm.

2.2. K-prototypes

K-prototypes algorithm integrates the k-means and k-modes algorithms to deal with the mixed data types [7]. The k-prototypes algorithm is more useful practically because data collected in the real world are mixed type objects. Assume a set n objects, $X = \{X_1,\ X_2, \cdots, X_{\mathrm{n}}\}$. $X_i = \{X_{i1}, X_{i2}, \cdots, X_{im}\}$ consists of m attributes (m_r is numerical attributes, m_c is categorical attributes, $m = m_r + m_c$). The goal of clustering is to partition n objects into k disjoint clusters $C = \{C_1, C_2, \cdots, C_k\}$, where C_i is an i-th cluster center. The distance $d(X_i, C_j)$ between X_i and C_j can be calculated as follows:

$$d(X_i, C_j) = d_r(X_i, C_j) + \gamma\, d_c(X_i, C_j) \tag{1}$$

where $d_r(X_i, C_j)$ is the distance between numerical attributes, $d_c(X_i, C_j)$ is the distance between categorical attributes, and γ is a weight for categorical attributes.

$$d_r(X_i, C_j) = \sum_{l=1}^{p} \left| x_{il} - c_{jl} \right|^2 \tag{2}$$

$$d_c(X_i, C_j) = \sum_{l=p+1}^{m} \delta\left(x_{il}, c_{jl}\right) \tag{3}$$

$$\delta\left(x_{il}, c_{jl}\right) = \begin{cases} 0, & \text{when} \quad x_{il} = c_{jl} \\ 1, & \text{when} \quad x_{il} \neq c_{jl} \end{cases}. \tag{4}$$

In Equation (2), $d_r(X_i, C_j)$ is the squared Euclidean distance measure between cluster centers and an object on the numerical attributes. $d_c(X_i, C_j)$ is the simple matching dissimilarity measure on the categorical attributes, where $\delta\left(x_{il}, c_{jl}\right) = 0$ for $x_{il} = c_{jl}$ and $\delta\left(x_{il}, c_{jl}\right) = 1$ for $x_{il} \neq c_{jl}$. x_{il} and c_{jl}, $1 \leq l \leq p$, are values of numerical attributes, whereas x_{il} and c_{jl}, $p+1 \leq l \leq m$ are values of categorical attributes for object i and the cluster center j. p is the numbers of numerical attributes and $m - p$ is the numbers of categorical attributes.

3. K-prototypes Using Partial Distance Computation

The existing k-prototypes algorithm allocates objects to the cluster with the smallest distance by calculating the distance between each cluster center and a new object to be allocated to the cluster. Distance is calculated by comparing all attributes of an object with all attributes of each cluster center using the brute force method. Figure 1 illustrates how k-prototypes algorithm organizes clusters with a target object. In this figure, an object consists of two numerical attributes and two categorical attributes. The entire dataset is divided into three clusters, $C = \{C_1, C_2, C_3\}$. The center of each cluster is $C_1 = (3, 3, C, D)$, $C_2 = (6, 6, A, B)$, and $C_3 = (9, 4, A, B)$. The traditional k-prototypes algorithm calculates distance with each cluster center to find the cluster to which X_i = (5, 3, A, B) is assigned. The distance about the numerical attribute of X_i and C_1, C_2, C_3 is $(3-5)^2 + (3-3)^2 = 4$, $(6-5)^2 + (6-3)^2 = 10$, $(9-5)^2 + (4-3)^2 = 17$, respectively. The distance about the categorical attribute of X_i and C_1, C_2, C_3 is $1 + 1 = 2 \because \mathrm{C} \neq \mathrm{A}$, $D \neq \mathrm{B}$, $0 + 0 = 0 \because \mathrm{A} = \mathrm{A}$, $\mathrm{B} = \mathrm{B}$, respectively. The total distance of X_i and C_1, C_2, and C_3 is 6, 10, and 17, respectively, and the cluster closest to X_i is X_i. Thus, the traditional k-prototypes algorithm computes the distance as a brute force method that compares both numerical and categorical properties.

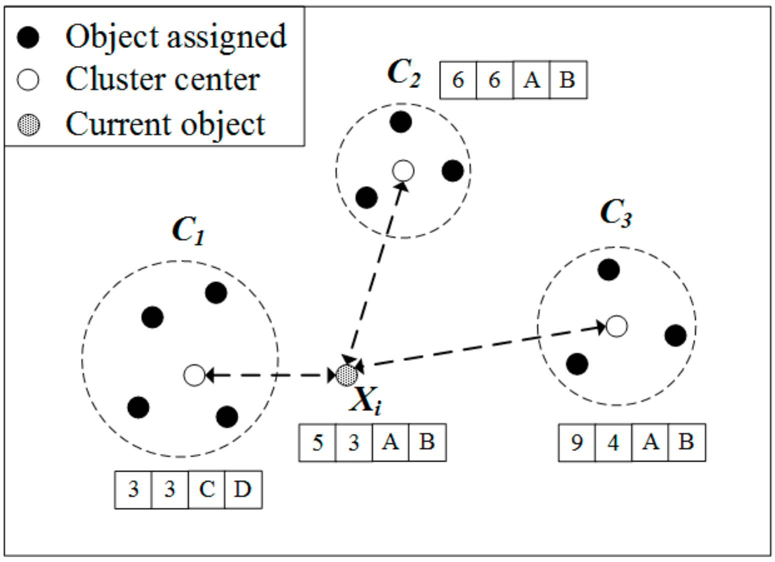

Figure 1. A process of assigning an object X_i to a cluster of which the center is the closest to the objects.

The purpose of distance computation is to find a cluster center closest to an object. However, there is an unnecessary distance calculation in the traditional k-prototypes algorithm. According to Equation (4), the maximum value that can be obtained is 1 when comparing a single categorical attribute. In Figure 1, an object has two categorical properties, so the maximum value that can be extracted from the distance comparing the categorical property is 2. In Figure 1, when using a numerical attribute, the closest cluster with the object is C_1 and a distance of 4, the second closest cluster is C_2, a distance of 4. Since the difference between these two values is greater than 2, comparing the numerical property, nevertheless the measured minimum distance, 4 added to the categorical property comparison maximum value 2, it does not exceed 10. In such a case, the cluster center closest to an object can be determined by calculating the numerical attribute value without calculating the category curl attribute in the distance calculation. Of course, the minimum distance cannot be obtained by comparing numerical properties for all cases only. In Figure 1, at $X_j = (0,0,0,0)$, the distance of the numerical attribute of X_j and C_1, C_2, C_3 is $(3-x)^2 + (3-x)^2 = \text{x}$, $(6-x)^2 + (6-x)^2 = \text{x}$, $(9-x)^2 + (4-x)^2 = \text{x}$, respectively.

This paper studies a method to find the closest center to an object without comparison for all attributes in a distance computation. We prove that the closest center to an object can be found without comparison for all attributes. For a proof, we define a computable max difference value as follows.

Definition 1. *The computable max difference value means that the maximum difference value can be calculated in the distance measured between an object and cluster centers for one attribute.*

According to Equation (4), the distance for a single categorical attribute between cluster centers and an object is either 0 or 1. Therefore, a computable max difference value for a categorical attribute, according to Definition 1, becomes 1 without taking the value of the attribute into account. If an object in the dataset consists of m categorical attributes, then a computable max difference value between a cluster and object is m. A computable max difference value of a numerical attribute is a difference of a maximum value and a minimum value of the attribute. Thus, to know a computable max difference value of a numerical attribute, we have to scan full datasets so that maximum and minimum values are obtained.

The proposed k-prototypes algorithm finds a minimum distance without distance computations of all attributes between an object and a cluster center using the computable max difference value of the object. The k-prototypes algorithm updates a cluster center after an object is assigned to the cluster of the closest center by the distance measure. By Equation (1), the distance $d(X_i, C_j)$ between an object and a cluster center is computed by adding the distance of numerical attributes and the distance of categorical attributes. If a difference of the first and the second minimum distance on numerical attributes is higher than m, we can find a minimum distance between an object and a cluster center using only distance computation of numerical attributes without distance computations of categorical attributes.

Lemma 1. *For a set of objects with m categorical attribute, if $d_r(X_i, C_b) - d_r(X_i, C_a) > m$ then $d(X_i, C_b) > d(X_i, C_a)$.*

Proof. $d_r(X_i, C_b) - d_r(X_i, C_a) > m$

$d_r(X_i, C_b) > m + d_r(X_i, C_a)$.

By Equation (2), $d(X_i, C_a) = d_r(X_i, C_a) + d_c(X_i, C_a)$

$d(X_i, C_b) = d_r(X_i, C_b) + d_c(X_i, C_b)$

$d(X_i, C_b) - d_c(X_i, C_b) > m + d(X_i, C_a) - d_c(X_i, C_a)$.

According to Definition 1, the categorical distance between an object and a cluster center with m categorical attributes can be $0 \le d_c(X_i, C_a) \le m$ and $0 \le d_c(X_i, C_b) \le m$. $d(X_i, C_b) \ge d(X_i, C_b) - d_c(X_i, C_b) > m + d(X_i, C_a) - d_c(X_i, C_a) \ge d(X_i, C_a)$. $\therefore d(X_i, C_b) > d(X_i, C_a)$.

We introduce a way to determine the minimum distance between an object and each cluster center with only computation of numerical attribute by an example.

Example 1. We assume that $k = 3$. Each current cluster center is as follows: $C_1 = (A,\ A,\ A,\ 5)$, $C_2 = (B,\ B,\ B,\ 7)$, and $C_3 = (B,\ B,\ B,\ 8)$. As shown by Figure 2, we have to compute the distance between $X_i = (B,\ B, B, 4)$ and each of cluster centers (C_1, C_2, and C_3) for assigning X_i to the cluster of the closest center. Firstly, we compute the distance of numerical attributes, $d_r(X_i, C_j)$ is 1, 9, and 16, respectively. X_i is the closest to C_1 only with numerical attributes. In this example, objects consisted of three categorical attributes; the minimum value of possible distance is 0, and the maximum value is 3. The difference of numerical distance between $d_r(X_i, C_1)$ and $d_r(X_i, C_2)$ is 8. Thus, X_i continues to be the closest to C_1, even if $d_c(X_i, C_1)$ is calculated by 3 as the computable max difference value.

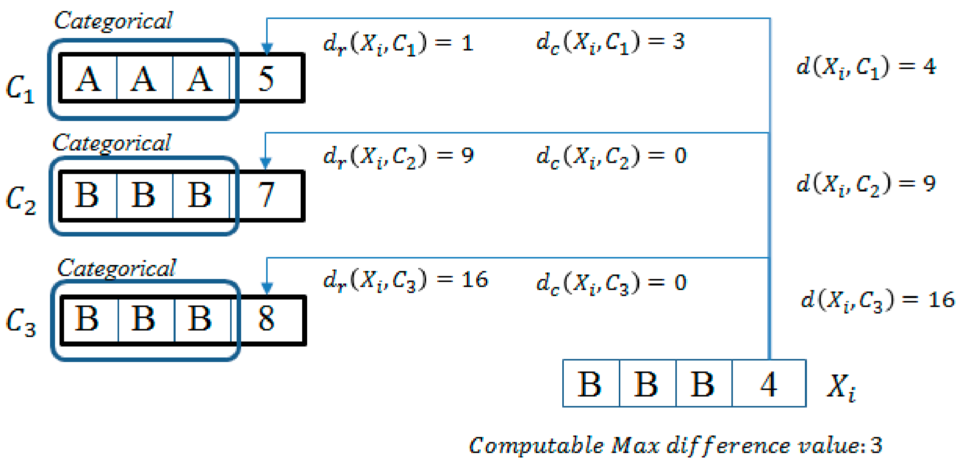

Figure 2. Finding the closest cluster center without computing categorical attributes.

3.1. Proposed Algorithm

In this section, we describe our proposed algorithm.

The proposed k-prototypes algorithm in this paper is similar to traditional k-prototypes. The difference between the proposed k-prototypes and traditional k-prototypes is that the distance between an object and cluster centers on the numerical attributes, $d_r(X_i, C_j)$, is calculated firstly.

In Line 4, firstly, you calculate the distance for a numerical attribute. You obtain the closest distance and the second closest distance value while calculating the distance. Using these two values and the number of the categorical attributes, m, the discriminant is performed. If the result of the discriminant is true, the distance to the categorical property is calculated, and then the result of the final distance is derived by adding the distance of the numerical property. If the result of the discriminant is false, the final distance is measured by the numerical attribute result only. By including X_i in the cluster measured at the smallest distance, the value of the corresponding cluster center is updated.

Definition 1 is a function that determines whether to compare the categorical attribute with the algorithm that implements it. Returns the true value if the difference between the second smallest distance and the first smallest distance is less than m in the distance measured only by a numerical property comparison between an object and cluster center. If a true value is returned, Algorithm 1 calls a function that compares the distance of the categorical attribute to calculate the final distance. If a false value is returned, the distance measured by only the numerical property comparison is set as the final results value without comparing the categorical property. The larger the difference between the two distances, the greater the number of categorical attributes that need not be compared.

Algorithm 1 Proposed k-prototypes algorithm

```
Input: n: the number of objects, k: the number of cluster, $p$: the number of numeric attribute, $q$: the number of categorical attribute
Output: $k$ cluster
01: INITIALIZE // Randomly choosing k object, and assigning it to $C_j$.
02: While not converged do
03:    for $i$ = 1 to $n$ do
04:       dist_n[] = DIST-COMPUTE-NUM($X_i$, C, $k$, $p$) // distance computation only numeric numerical attributes
05:         first_min = DIST-COMPUTE.first_min        // first minimum value among $d_r\left(X_i, C_j\right)$
06:         second_min = DIST-COMPUTE.second_min    // second minimum value among $d_r\left(X_i, C_j\right)$
07:       if (second_min − first_min < $m$) then
08:             dist[] = dist_n[] + DIST-COMPUTE-CATE($X_i$, C, $k$)
09:    else
10:             dist[] = dist_n[]
11:         num = $\underset{z}{\text{argmin}}\, dist[z]$
12:         $X_i$ is assigned to $C_{num}$
13:         UPDATE-CENTER($C_{num}$)
```

In Algorithm 2, DIST-COMPUTE-NUM() calculates a distance between an object and cluster centers for numerical attributes and returns all distances for each cluster. In this algorithm, first_min and second_min is calculated to determine whether calculation of categorical data in a distance computation.

Algorithm 2 DIST-COMPUTE-NUM()

```
Input: $X_i$: an object vector, C: a set of cluster center vectors, $k$: the number of clusters, $p$: the number of numeric attribute
Output: dist_n[], first_min, second_min
01: for $i$ = 1 to $k$ do
02:    for $j$ = 1 to $p$ do
03:       dist_n[j] $= (X[j] - C_i[j])^2$
04: first_min = dist_n[0]
05: second_min = dist_n[0]
06: for $i$ = 0 to $k$-1 do
07:    if (dist_n[i] < first_min) then
08:       second_min = first_min
09:       first_min = dist_n[i]
10:    else if (dist_n[i] < second_min) then
11:       second_min = dist_n[i]
12: Return dist_n[]
```

In Algorithm 3, a distance between an object and cluster centers is calculated for categorical attributes of each cluster.

Algorithm 3 DIST-COMPUTE-CATE()

```
Input: $X_i$: an object vector, C: cluster center vectors, $k$: the number of clusters, $p$: the number of numeric attribute
Output: dist_c[]
01: for $i$ = 1 to $k$ do
02:     for $j = p + 1$ to $m$ do
03:         if ($X[j] = C_i[j]$) then
04:             dist_c[i] += 0
05:         else
06: dist_c[i] += 1
07: Return dist_c[]
```

In Algorithm 4, the center vector of a cluster is assigned to new center vector. The center vectors consisted of two parts: numerical and categorical attributes. The numerical part of the center vector is calculated by an average value of each numerical attributes and the categorical part of the center vector is calculated by the value of the highest frequency in each categorical attribute.

Algorithm 4 UPDATE-CENTER()

```
Input: $C_i$: an $i$-th cluster center vectors
01: foreach o $\in C_i$ do
02: for $j$ = 1 to $p$ do
03:     sum[j] += o[j]
04: for $j$ = 1 to $p$ do
05:     C[j] = sum[j] / $|C_i|$
06: for $j = p$ +1 to $m$ do
07:     C[j] = argmax COUNT(o[j])
```

3.2. Time Complexity

The time complexity of traditional k-prototypes is $O(I * k * n * m)$, where I is the number of iterations, k is the number of clusters, n is the number of data objects, and m is the number of attributes. The best-case complexity of the proposed k-prototypes has a lower bound of $\Omega(I * k * n * p)$, where p is the number of numerical attributes and $p < m$. The best case is that the difference of the first and the second minimum distance between an object and cluster centers for all objects in a given dataset on numerical attributes is less than m. The worst-case complexity has an upper bound of $O(I * k * n * m)$. The worst case is that the difference of the first and the second minimum distance between an object and cluster centers for all objects in a given dataset on numerical attributes is higher than m.

4. Experimental Results

All experiments are conducted on an Intel(R) Pentium(R) 3558U 1.70 GHz, 4GB RAM. All programs are written in Java. We generate several independent, uniform distribution mix typed datasets. A distribution of numerical attributes is from 0 to 100, and one of the categorical attributes is from A to Z.

Effect of Cardinality

We set |X| (number of objects) = {500000, 800000, 1000000}, numerical attributes = 2, categorical attributes = 16 and k = 3. Figure 3 shows the CPU time versus cardinality in different datasets. In the figure, there are two lines. In general, the CPU time increases linearly when the cardinality increases linearly. The experimental shows proposed k-prototypes algorithm improves computational performance than original k-prototypes algorithm in our dataset.

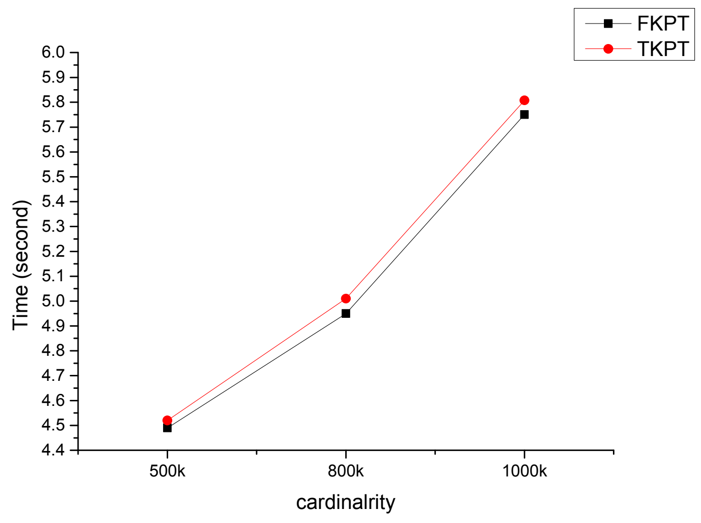

Figure 3. Effect of cardinality. FKPT (fast k-prototypes) is the result of our propose k-prototypes algorithm and TKPT (traditional k-prototypes) is the result of original k-prototypes algorithm.

To analyze the difference CPU time for each dataset, 114,844, 85,755, and 5,753,212 computation decreased in 500k, 800k and 1000k datasets, respectively. These computation reductions means that the number of calculation categorical attribute in distance calculation between a point and a center decreases. Decreasing the computation in distance calculation between a point and a center seems to reduce CPU time. The final clustering results of our proposed algorithm is the same as the clustering results of original k-prototypes algorithm. These results lead us to conclude that the proposed algorithm has better performance than the original k-prototypes algorithm.

5. Conclusions

In this paper, we have proposed a fast k-prototypes algorithm for clustering mixed datasets. Experimental results show that our algorithm is fast than the original algorithm. Previous fast k-means algorithm focused on reducing candidate objects for computing distance to cluster centers. Our k-prototypes algorithm reduces unnecessary distance computation using partial distance computation without distance computations of all attributes between an object and a cluster center, which allows it to reduce time complexity. The experimental shows proposed k-prototypes algorithm improves computational performance than the original k-prototypes algorithm in our dataset.

However, our k-prototypes algorithm does not guarantee that the computational performance will be improved in all cases. If the difference of the first and the second minimum distance between an object and cluster centers for all objects in a given dataset on numerical attributes is less than m, then the performance of our k-prototypes is the same as the original k-prototypes. Our k-prototypes algorithm is influenced by the variance of the numerical data values. The larger variance of the numerical data values, the higher probability that the difference of the first and the second minimum distance between an object and cluster centers is large.

The k-prototypes algorithm proposed in this paper simply reduces the computational cost without using additional data structures and memories. Our algorithm is faster than the original k-prototypes algorithm. The goal of the existing k-means acceleration algorithm is to reduce the number of dimensions to be compared when calculating the distance between center and object, in order to reduce the number of objects compared with the center of the cluster. K-means, which deals only with numerical data, is the most widely used algorithm among clustering algorithms. Various acceleration algorithms have been developed to improve the speed of processing large data. However, real-world data is mostly a mixture of numerical data and categorical data. In this paper, we propose a method to speed up the k-prototypes algorithm for clustering mixed data. The method proposed in this paper is a method of reducing the number of objects compared with the center of existing clusters, and is not exclusive, and the existing methods and the methods proposed in this paper can be integrated with each other.

In this study, we considered only an effect of cardinality. However, the performance of the algorithm may be affected by dimensionality, the size of k, and data distributions. In future works, firstly we have a plan to measure performance of our proposed algorithm by dimensionality, the size of k, and data distributions. Secondly, we can prune more objects by using partitioning data points into grid. We will extend our fast k-prototypes algorithm to improve for better performance.

Author Contributions: Byoungwook Kim: Research for the related works, doing the experiments, writing the paper, acquisition of data, analysis of data, interpretation of the related works, and design of the complete model.

Conflicts of Interest: The author declares no conflict of interest.

References

1. MacQueen, J.B. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability: Statistics, Oakland, CA, USA, 21 June–18 July 1965 and 27 November 1965–7 January 1966; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
2. Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. NP-hardness of euclidean sum-of-squares clustering. *Mach. Learn.* **2009**, *75*, 245–249. [CrossRef]
3. Dasgupta, S.; Freund, Y. Random projection trees for vector quantization. *IEEE Trans. Inf. Theory* **2009**, *55*, 3229–3242. [CrossRef]
4. Drake, J.; Hamerly, G. Accelerated k-means with adaptive distance bounds. In Proceedings of the 5th NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, NV, USA, 7–8 December 2012.
5. Elkan, C. Using the triangle inequality to accelerate k-means. In *Tom Fawcett and Nina Mishra*; Fawcett, T., Mishra, N., Eds.; AAAI Press: Washington, DC, USA, 2003; pp. 147–153.
6. Hamerly, G. Making k-means even faster. In Proceedings of the 2010 SIAM International Conference on Data Mining, Columbus, OH, USA, 29 April–1 May 2010; pp. 130–140.
7. Huang, Z. Clustering large data sets with mixed numeric and categorical values. In Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore, 23–24 February 1997; pp. 21–34.
8. Pelleg, D.; Moore, A.W. Accelerating exact k-means algorithms with geometric reasoning. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999; Association of Computing Machinery Press: NY, USA, 1999; pp. 277–281.
9. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
10. Cheng, D.; Gersho, A.; Ramamurthi, B.; Shoham, Y. Fast search algorithms for vector quantization and pattern matching. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, San Diego, CA, USA, 19–21 March 1984; pp. 372–375.
11. McNames, J. Rotated partial distance search for faster vector quantization encoding. *IEEE Signal Process. Lett.* **2000**, *7*, 244–246. [CrossRef]

© 2017 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).