A Fast Implementation of the ISOCLUS Algorithm

Nargess Memarsadeghi
NASA/GSFC, Code 588
Greenbelt, MD 20770
nargess@cs.umd.edu

David M. Mount
University of Maryland
College Park, MD 20742
mount@cs.umd.edu

Nathan S. Netanyahu
Bar-Ilan University &
University of Maryland
College Park, MD 20742
nathan@cfar.umd.edu

Jacqueline Le Moigne
NASA/GSFC, Code 935
Greenbelt, MD 20770
lemoigne@backserv.gsfc.nasa.gov

I. Introduction

Unsupervised clustering is a fundamental tool in numerous image processing and remote sensing applications. For example, unsupervised clustering is often used to obtain vegetation maps of an area of interest. This approach is useful when reliable training data are either scarce or expensive, and when relatively little *a priori* information about the data is available. Unsupervised clustering methods play a significant role in the pursuit of unsupervised classification [1].

One of the most popular and widely used clustering schemes for remote sensing applications is the ISOCLUS algorithm [2], which is based on the ISODATA method [3]–[5]. The algorithm is given a set of n data points (or samples) in d-dimensional space, an integer k indicating the initial number of clusters, and a number of additional parameters. The general goal is to compute a set of cluster centers in d-space. Although there is no specific optimization criterion, the algorithm is similar in spirit to the well known k-means clustering method [4] in which the objective is to minimize the average squared distance of each point to its nearest center, called the *average distortion*. One significant feature of ISOCLUS over k-means is that clusters may be merged or split, and so the final number of clusters may be different from the number k supplied as part of the input. This algorithm will be described in later in this paper. The ISOCLUS algorithm can run very slowly, particularly on large data sets. Given its wide use in remote sensing, its efficient computation is an important goal.

We have developed a fast implementation of the ISOCLUS algorithm. Our improvement is based on a recent acceleration to the k-means algorithm, the *filtering algorithm*, by Kanungo *et al.* [6]. They showed that, by storing the data in a kd-tree [7], it was possible to significantly reduce the running time of k-means. We have adapted this method for the ISOCLUS algorithm. For technical reasons, which are explained later, it is necessary to make a minor modification to the ISOCLUS specification. We provide empirical evidence, on both synthetic and Landsat image data sets, that our algorithm's performance is essentially the same as that of ISOCLUS, but with significantly lower running times. We show that our algorithm runs from 3 to 30 times faster than a straightforward implementation of ISOCLUS. Our adaptation of the filtering algorithm involves the efficient computation of a number of cluster statistics that are needed for ISOCLUS, but not for k-means.

II. Preliminaries

A. k-Means and Lloyd's algorithm

Lloyd's algorithm (often called the *k-means algorithm*) is a widely used heuristic for k-means clustering [8], [9]. It iteratively repeats the following two steps until convergence. First, for each cluster center, it computes the set of points for which this center is the closest. Next, it moves each center to the centroid of its associated set. It can be shown that with each step the average squared distortion decreases and that the algorithm converges to a local minimum [10].

The running time of the algorithm is dominated by the time to compute the nearest cluster center to each data point. As mentioned above, Kanungo *et al.* [6] presented a fast implementation of Lloyd's algorithm, called the *filtering algorithm*. Here is a high-level description of the algorithm. (See [6] for details and analysis.) The algorithm preprocesses the data points by storing them in a kd-tree [7]. This data structure hierarchically aggregates points into rectangular cells, one cell associated with each node of the tree. As part of the preprocessing, the weighted centroid of points lying within each cell is computed. The filtering algorithm visits the nodes of the tree in a top-down manner. For each node, it maintains the subset of centers, called *candidates*, that are closest to some point of the cell. If there is only one candidate center, then by using the weighted centroid, all the points of this node can be assigned to this center in constant time. If there are multiple centers, the algorithm propagates the candidates to the two children of this node in the kd-tree. It then uses a simple geometric test to prune the set of centers.

The filtering algorithm achieves its efficiency by assigning many points at once to each center. A straightforward implementation of Lloyd's algorithm requires $O(kn)$ time to compute the distance from each of the n points to each of the k centers. In contrast, if we consider the comparable quantity for the filtering algorithm, that is, the number of interactions between nodes and candidates, the number is smaller by factors ranging from 10 to 200 (for low dimensional clustered data sets). Even considering the additional preprocessing time and overhead, the speed-ups in actual CPU time are often quite significant. (See [6] for further details.) Other algorithms for accelerating k-means have been given by Pelleg and Moore [11] and Phillips [12].

B. ISOCLUS Algorithm

ISOCLUS is a clustering algorithm based on the ISODATA clustering algorithm [4], [13] with minor modifications [2]. Like the k-means algorithm, ISOCLUS tries to find the best cluster centers through an iterative approach, until some convergence criteria are met. ISOCLUS uses different heuristics to determine when to merge or split clusters. There are a number of user-supplied parameters. These include the following.

- The desired number of clusters (*NumClus*).
- The minimum number of samples in a cluster (*SampRm*).
- The maximum number of iterations (*MaxIter*).
- The maximum standard deviation per cluster (*StdDev*).
- The lumping parameter (*Lump*) and the maximum number of pairs that can be lumped per iteration (*MaxPair*).

Here is an overview of the ISOCLUS algorithm. (See [2] for details.) Since the algorithm can run very slowly on large data sets, the algorithm first samples points randomly from the original data set. Then, it randomly selects *NumClus* centers from the samples. After inputting the parameter values (Step 1), the following steps are then repeated until termination. First, the distances from points to the centers are calculated, and points are assigned to their closest centers (Step 2). Next, clusters with fewer than *SampRm* samples are deleted (Step 3), and the cluster centers are moved to the mean (centroid) of the samples in the remaining clusters (Step 4). Steps 2–4 are repeated until no clusters are deleted in Step 3. Observe that Steps 2 and 4 together constitute one iteration of Lloyd's algorithm. The algorithm calculates the average distances of samples from their nearest cluster center (Step 5) and computes the overall average distance (Step 6).

Next, the algorithm considers splitting and merging clusters. Depending on the relationships between the number of clusters and *NumClus* and the number of iterations and *MaxIter*, the algorithm may skip one or more of the following steps (Step 7). For each cluster, the standard deviation along each of the coordinate axes is computed (Step 8), as well as the maximum standard deviation value along each coordinate (Step 9). Based on this information and the parameters *StdDev* and *NumClus*, the algorithm splits large clusters with high standard deviations (Step 10). If a cluster was split, the algorithm returns to Step 2. Otherwise, the parameters *Lump* and *MaxPair* are used to merge pairs of nearby clusters (Steps 11–13). The final stage (Step 14) determines whether to terminate the algorithm or increment the number of iterations and return to Step 2.

The main difference between ISODATA and ISOCLUS is in Steps 2–4 (Lloyd's algorithm). In ISOCLUS these steps are repeated until no cluster is deleted in Step 3, while ISODATA performs these steps only once during each iteration.

III. Our Improvements

Most of the computational effort in the ISOCLUS algorithm is spent calculating and updating distances and distortions in Steps 2–5. These steps take $O(kn)$ time, whereas all the other steps can be done in $O(k)$ time, where k is the current number of centers. Our improvement is achieved by adapting the filtering algorithm to compute the desired information.

There is one wrinkle, however. The filtering algorithm achieves its efficiency by processing points in groups, rather than individually. We can preprocess the data set to compute the sums of *squared* Euclidean distances (and in general, any polynomial function of the input coordinates). In contrast, Steps 5 and 6 of ISOCLUS involve computing the sum of (unsquared) Euclidean distances. We know of no way to preprocess the data to compute sums of distances exactly, because of the square roots involved.

Thus, rather than implementing ISOCLUS exactly as described in [2], we modified Steps 5 and 6 to use sums of *squared* distances, rather than the sums of distances. Note that this can produce different results. Nonetheless, based on many executions on both synthetically generated data and real images, we have found that our algorithm's performance is quite similar to that of ISOCLUS, in terms of the number of clusters obtained and the positions of their centers. Thus, we believe that this modification does not significantly alter the nature of ISOCLUS. In order to calculate the coordinate standard deviations in Step 8, we need to add additional information to the kd-tree. In particular, each cell maintains the sum of squared coordinates for the points in each cell. Thus, we have three versions of the ISOCLUS algorithm which we will refer to as follows in the rest of this paper:

- **Standard version (Std):** The straightforward implementation of ISOCLUS as described [2], which uses Euclidean distances in Steps 5–6.
- **Regular version (Reg):** A modification using squared Euclidean distances in Steps 5–6.
- **Efficient version (Eff):** An implementation of the Regular version based on the filtering algorithm.

IV. Experiments

All experiments were run on a SUN Ultra 5 running Solaris 2.8, using the g++ compiler (version 2.95.3).

A. Synthetic data

We ran a number of different experiments to analyze the performance of our algorithm. For the first three experiments we generated $n = 10,000$ data points in 3-space. The points were distributed evenly among 25, 50, and 100 Gaussian clusters with a standard deviation of $\sigma = 0.005$ along each coordinate. The cluster centers were sampled uniformly at random from a hypercube of side length 2. The remaining three experiments were conducted the same way, but in 5-space. We ran the ISOCLUS algorithms with *MaxIter* $= 15$, *StdDev* $= 2\sigma = 0.01$, *Lump* $= 0.001$. The initial number of centers, *NumClus*, was set to 25, 50, and 100, respectively, and *SampRm* $= n/(5 \cdot$ *NumClus*$)$. In each case, results were averaged over three runs using different random seeds.

The results are shown in Table I. For each run, we computed the running time in CPU seconds, the final number of centers, and the final average distortion. Since the regular and efficient versions implement the same functional specifications, the final numbers of centers and final distortions are almost identical. (Small differences were observed, due to floating

TABLE I

RESULTS OF SYNTHETIC DATA TEST INPUTS

Dim	NumClus	Final No. of Centers		Avg. Distortion $\times 1000$		CPU Seconds			Speed-up
		Std	Reg/Eff	Std	Reg/Eff	Std	Reg	Eff	
3	25	29	29	0.07242	0.07242	0.9018	0.8244	0.0751	10.98
	50	54	54	0.07407	0.07407	1.7147	1.5813	0.0778	20.33
	100	114	114	0.07213	0.07213	3.7593	3.4598	0.104	33.27
5	25	27	27	0.1238	0.1238	1.2489	1.1716	0.0969	12.09
	50	58	58	0.1216	0.1216	2.7633	2.5807	0.1233	20.93
	100	115	115	0.1205	0.1205	5.4737	5.1534	0.1816	28.38

TABLE II

RESULTS OF LANDSAT DATA TEST INPUTS

Dim	NumClus	Final No. of Centers		Avg. Distortion $\times 1000$		CPU Seconds			Speed-up
		Std	Reg/Eff	Std	Reg/Eff	Std	Reg	Eff	
3	25	9	9	46.83	46.96	3.385	3.30	0.533	6.19
7	25	14	14	59.08	58.19	2.994	2.95	0.886	3.33

point round-off errors.) So, we list them together in the table (under the heading "Reg/Eff"). We also computed the *speed-up*, which is defined as the ratio between the CPU times of the regular and the efficient versions.

In support of our claim that using squared distances does not significantly change the algorithm's clustering performance, observe that both algorithms performed virtually identically with respect to average distortions and the final number of centers. Also observe that the standard and regular versions run in roughly the same time, whereas the efficient version runs around 10 to 30 times faster than the other two.

B. Image data

For image data we ran two tests on a 256×256 Landsat image of Ridgely, Maryland ($n = 65,536$). The first involved 3-dimensional data using bands 3, 4, and 5, and the second used all seven bands. We ran both tests with all three versions of ISOCLUS, each for 20 iterations and with *StdDev* $=$ *Lump* $= 10$ and *SampRm* $= 100$. The results are presented in Table II. As with the synthetic tests, all versions performed essentially equivalently with respect to the number of centers and final distortions. The efficient version was faster by a factor of roughly 3 and 6. We believe that the differences in speed-up between the synthetic and real images are due to the fact that the clusters are fewer and not as well separated as in the synthetic case. The filtering algorithm performs best when there are many clusters that are well separated [6].

V. CONCLUSION

We have demonstrated the efficiency of a new implementation of the ISOCLUS algorithm, based on the use of the kd-tree data structure and the filtering algorithm. Our algorithm is a slight modification of the original ISOCLUS algorithm, because it uses squared, rather than unsquared, Euclidean distances in cluster splitting.

Our experiments indicate that using squared distances yields essentially the same results. The experiments on synthetic clustered data showed speed-ups in running times ranging from 10 to 30, while the experiments on Landsat image data showed speedups of 3 to 6.

REFERENCES

[1] J. Richards and X. Jia, *Remote Sensing Digital Image Analysis*. Berlin: Springer, 1999.

[2] PCI Geomatics Corp., "ISOCLUS–Isodata clustering program," http://www.pcigeomatics.com/cgi-bin/pcihlp/ISOCLUS.

[3] G. H. Ball and D. J. Hall, "Some fundamental concepts and synthesis procedures for pattern recognition preprocessors," in *Intl. Conf. on Microwaves, Circuit Theory, and Inform. Theory*, Tokyo, Japan, Sept. 1964.

[4] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[5] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. London: Addison-Wesley, 1974.

[6] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 881–892, 2002.

[7] J. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, 1975.

[8] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. 28, pp. 129–137, 1982.

[9] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Prob.*, vol. 1, Berkeley, CA, 1967, pp. 281–296.

[10] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 6, pp. 81–87, 1984.

[11] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," in *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, Aug. 1999, pp. 277–281.

[12] S. Phillips, "Reducing the computation time of the ISODATA and k-means unsupervised classification algorithms," in *Proc. 22nd IEEE Intl. Geosci. and Remote Sensing Symp.*, Toronto, Canada, June 2002.

[13] G. H. Ball and D. J. Hall, "ISODATA, A novel method of data analysis and pattern classification," Stanford Research Institute, Menlo Park, CA, Tech. Rep. AD 699616, 1965.