Balanced *K*-Means for Clustering

Mikko I. Malinen and Pasi Fränti

School of Computing, University of Eastern Finland,
Box 111, FIN-80101 Joensuu, Finland
{mmali,franti}@cs.uef.fi
http://cs.uef.fi/~mmali, http://cs.uef.fi/pages/franti

Abstract. We present a k-means-based clustering algorithm, which optimizes mean square error, for given cluster sizes. A straightforward application is balanced clustering, where the sizes of each cluster are equal. In k-means assignment phase, the algorithm solves the assignment problem by Hungarian algorithm. This is a novel approach, and makes the assignment phase time complexity $O(n^3)$, which is faster than the previous $O(k^{3.5}n^{3.5})$ time linear programming used in constrained k-means. This enables clustering of bigger datasets of size over 5000 points.

Keywords: clustering, balanced clustering, assignment problem, Hungarian algorithm.

1 Introduction

Euclidean sum-of-squares clustering is an NP-hard problem [1], which groups n data points into k clusters so that intra-cluster distances are low and inter-cluster distances are high. Each group is represented by a center point (centroid). The most common criterion to optimize is the mean square error (MSE):

$$\text{MSE} = \sum_{j=1}^{k} \sum_{X_i \in C_j} \frac{\| X_i - C_j \|^2}{n}, \tag{1}$$

where X_i denotes data point locations and C_j denotes centroid locations. K-means [19] is the most commonly used clustering algorithm, which provides a local minimum of MSE given the number of clusters as input. K-means algorithm consists of two repeatedly executed steps:

Assignment Step: Assign the data points to clusters specified by the nearest centroid:

$$P_j^{(t)} = \{X_i : \|X_i - C_j^{(t)}\| \le \|X_i - C_{j^*}^{(t)}\| \quad \forall \quad j^* = 1, ..., k\}$$

Update Step: Calculate the mean of each cluster:

$$C_j^{(t+1)} = \frac{1}{|P_j^{(t)}|} \sum_{X_i \in P_j^{(t)}} X_i$$

© Springer-Verlag Berlin Heidelberg 2014

These steps are repeated until centroid locations do not change anymore. K-means assignment step and update step are optimal with respect to MSE: The partitioning step minimizes MSE for a given set of centroids; the update step minimizes MSE for a given partitioning. The solution therefore converges to a local optimum but without guarantee of global optimality. To get better results than in k-means, slower agglomerative algorithms [10,13,12] or more complex k-means variants [3,11,21,18] are sometimes used.

In *balanced clustering* there are an equal number of points in each cluster. Balanced clustering is desirable for example in divide-and-conquer methods where the divide step is done by clustering. Examples can be found in circuit design [14] and in photo query systems [2], where the photos are clustered according to their content. Applications can also be used in workload balancing algorithms. For example, in [20] multiple traveling salesman problem clusters the cities, so that each salesman operates in one cluster. It is desirable that each salesman has equal workload. Networking utilizes balanced clustering to obtain some desirable goals [17,23].

We next review existing balanced clustering algorithms. In *frequency sensitive competitive learning* (FSCL) the centroids compete of points [5]. It multiplicatively increases the distance of the centroids to the data point by the times the centroid has already won points. Bigger clusters are therefore less likely to win more points. The method in [2] uses FSCL, but with additive bias instead of multiplicative bias. The method in [4] uses a fast ($O(kNlogN)$) algorithm for balanced clustering based on three steps: sample the given data, cluster the sampled data and populate the clusters with the data points that were not sampled. The article [6] and book chapter [9] present a *constrained k-means algorithm*, which is like k-means, but the assignment step is implemented as a linear program, in which the minimum number of points τ_h of clusters can be set as parameters. The constrained k-means clustering algorithm works as follows:

Given m points in $\mathbb{R}^n$, minimum cluster membership values $\tau_h \geq 0, h = 1, ..., k$ and cluster centers $C_1^{(t)}, C_2^{(t)}, ..., C_k^{(t)}$ at iteration t, compute $C_1^{(t+1)}, C_2^{(t+1)}$, ..., $C_k^{(t+1)}$ at iteration $t+1$ using the following 2 steps:

Cluster Assignment. Let $T_{i,h}^t$ be a solution to the following linear program with $C_h^{(t)}$ fixed:

$$\text{minimize}_T \sum_{i=1}^{m} \sum_{h=1}^{k} T_{i,h} \cdot (\frac{1}{2} ||X_i - C_h^{(t)}||_2^2) \tag{2}$$

$$\text{subject to} \sum_{i=1}^{m} T_{i,h} \geq \tau_h, h = 1, ..., k \tag{3}$$

$$\sum_{h=1}^{k} T_{i,h} = 1, i = 1, ..., m \tag{4}$$

$$T_{i,h} \geq 0, i = 1, ..., m, h = 1, ..., k. \tag{5}$$

Cluster Update. Update $C_h^{(t+1)}$ as follows:

$$C_h^{(t+1)} = \begin{cases} \frac{\sum_{i=1}^m T_{i,h}^{(t)} X_i}{\sum_{i=1}^m T_{i,h}^{(t)}} & \text{if} \quad \sum_{i=1}^m T_{i,h}^{(t)} > 0, \\ C_h^{(t)} & \text{otherwise.} \end{cases}$$

These steps are repeated until $C_h^{(t+1)} = C_h^{(t)}, \quad \forall h = 1, ..., k.$

A cut-based method *Ratio cut* [14] includes cluster sizes in its cost function

$$\text{RatioCut}(P_1, ..., P_k) = \sum_{i=1}^{k} \frac{\text{cut}(P_i, \bar{P}_i)}{|P_i|}.$$

Here P_i:s are the partitions. *Size regularized cut* SRCut [8] is defined as the sum of the inter-cluster similarity and a regularization term measuring the relative size of two clusters. In [16] there is a balancing aiming term in cost function and [24] tries to find a partition close to the given partition, but so that cluster size constraints are fulfilled. There are also application-based solutions in networking [17], which aim at network load balancing, where clustering is done by self-organization without central control. In [23], energy-balanced routing between sensors is aimed so that most suitable balanced amount of nodes will be the members of the clusters.

Balanced clustering, in general, is a 2-objective optimization problem, in which two aims contradict each other: to minimize MSE and to balance cluster sizes. Traditional clustering aims at minimizing MSE without considering cluster size balance. Balancing, on the other hand, would be trivial if we did not care about MSE; simply by dividing points to equal size clusters randomly. For optimizing both, there are two alternative approaches: *Balance-constrained* and *balance-driven* clustering.

In balance-constrained clustering, cluster size balance is a mandatory requirement that must be met, and minimizing MSE is a secondary criterion. In balance-driven clustering, balance is an aim but not mandatory. It is a compromize between these two goals, namely the balance and the MSE. The solution can be a weighted compromize between MSE and the balance, or a heuristic that aims at minimizing MSE but indirectly creates a more balanced result than standard k-means. Existing algorithms are grouped into these two classes in Table 1.

In this paper, we formulate balanced k-means, so that it belongs to the first category. It is otherwise the same as standard k-means but it guarantees balanced cluster sizes. It is also a special case of constrained k-means, where cluster sizes are set equal. However, instead of using linear programming in the assignment phase, we formulate the partitioning as a pairing problem [7], which can be solved optimally by Hungarian algorithm in $O(n^3)$ time.

Table 1. Classification of some balanced clustering algorithms

Balance-constrained
Balanced k-means (proposed)
Constrained k-means [6]
Size constrained [24]
Balance-driven
FSCL [5]
FSCL with additive bias [2]
Cluster sampled data [4]
Ratio cut [14]
SRcut [8]
Submodular fractional programming [16]

2 Balanced *k*-Means

To describe balanced k-means, we need to define what is an assignment problem. The formal definition of assignment problem (or linear assignment problem) is as follows. Given two sets (A and S), of equal size, and a weight function $W : A \times S \to \mathbb{R}$. The goal is to find a bijection $f : A \to S$ so that the cost function is minimized:

$$\text{Cost} = \sum_{a \in A} W(a, f(a)).$$

In the context of the proposed algorithm, sets A and S correspond respectively to cluster slots and to data points, see Figure 1.

In balanced k-means, we proceed as in k-means, but the assignment phase is different: Instead of selecting the nearest centroids we have n pre-allocated slots (n/k slots per cluster), and datapoints can be assigned only to these slots, see Figure 1. This will force all clusters to be of same size assuming that $\lceil n/k \rceil = \lfloor n/k \rfloor = n/k$. Otherwise there will be $(n \mod k)$ clusters of size $\lceil n/k \rceil$, and $k - (n \mod k)$ clusters of size $\lfloor n/k \rfloor$.

To find assignment that minimizes MSE, we solve an assignment problem using Hungarian algorithm [7]. First we construct a bipartite graph consisting n datapoints and n cluster slots, see Figure 2. We then partition the cluster slots in clusters of as even number of slots as possible.

We give centroid locations to partitioned cluster slots, one centroid to each cluster. The initial centroid locations can be drawn randomly from all data points. The edge weight is the squared distance from the point to the cluster centroid it is assigned to. Contrary to standard assignment problem with fixed weights, here the weights dynamically change after each k-means iteration according to the newly calculated centroids. After this, we perform the Hungarian algorithm to get the minimal weight pairing. The squared distances are stored in a $n \times n$ matrix, for the sake of the Hungarian algorithm. The update step is

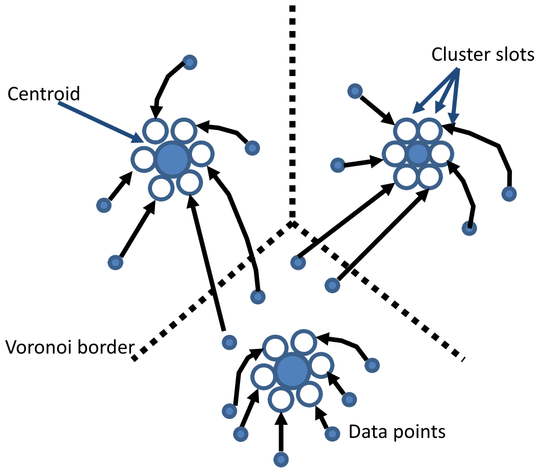

Fig. 1. Assigning points to centroids via cluster slots

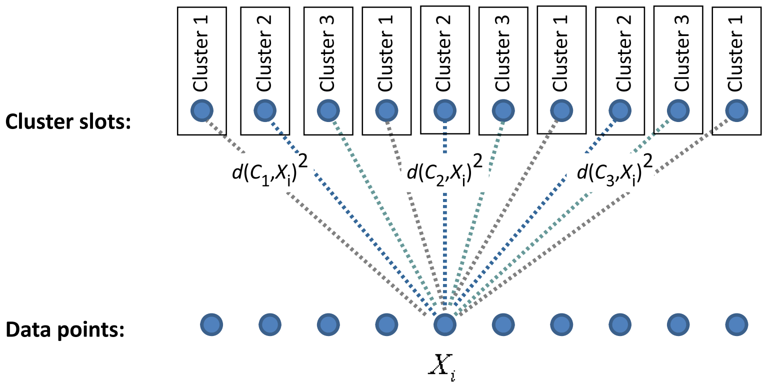

Fig. 2. Minimum MSE calculation with balanced clusters. Modeling with bipartite graph.

similar to that of k-means, where the new centroids are calculated as the means of the data points assigned to each cluster:

$$C_i^{(t+1)} = \frac{1}{n_i} \cdot \sum_{X_j \in C_i^{(t)}} X_j. \tag{6}$$

The weights of the edges are updated immediately after the update step. The pseudocode of the algorithm is in Algorithm 1. In calculation of edge weights, the number of cluster slot is denoted by a and mod is used in calculation of cluster where a cluster slot belongs to. The edge weights are calculated by

$$W(a,i) = dist(X_i, C^t_{(a \bmod k)+1})^2 \quad \forall a \in [1,n] \quad \forall i \in [1,n]. \tag{7}$$

Algorithm 1. Balanced k-means
Input: dataset X, number of clusters k
Output: partitioning of dataset.

```
Initialize centroid locations $C^0$.
$t \leftarrow 0$
repeat
    Assignment step:
        Calculate edge weights.
        Solve an Assignment problem.
    Update step:
        Calculate new centroid locations $C^{t+1}$
    $t \leftarrow t + 1$
until centroid locations do not change.
Output partitioning.
```

After convergence of the algorithm the partition of points X_i, $i \in [1, n]$, is

$$X_{f(a)} \in P_{(a \bmod k)+1}. \tag{8}$$

There is a convergence result in [6] (Proposition 2.3) for constrained k-means. The result says that the algorithm terminates in a finite number of iterations at a partitioning that is locally optimal. At each iteration, the cluster assignment step cannot increase the objective function of constrained k-means (3) in [6]. The cluster update step will either strictly decrease the value of the objective function or the algorithm will terminate. Since there are a finite number of ways to assign m points to k clusters so that cluster h has at least τ_h points, since constrained k-means algorithm does not permit repeated assignments, and since the objective of constrained k-means (3) in [6] is strictly nonincreasing and bounded below by zero, the algorithm must terminate at some cluster assignment that is locally optimal. The same convergence result applies to balanced k-means as well. The assignment step is optimal with respect to MSE because of pairing and the update step is optimal, because MSE is clusterwise minimized as is in k-means.

3 Time Complexity

Time complexity of the assignment step in k-means is $O(k \cdot n)$. Constrained k-means involves linear programming. It takes $O(v^{3.5})$ time, where v is the number of variables, by Karmarkars projective algorithm [15,22], which is the fastest interior point algorithm known to the authors. Since $v = k \cdot n$, the time complexity is $O(k^{3.5}n^{3.5})$. The assignment step of the proposed balanced k-means algorithm can be solved in $O(n^3)$ time with the Hungarian algorithm. This makes it much faster than in the constrained k-means, and allows therefore significantly bigger datasets to be clustered.

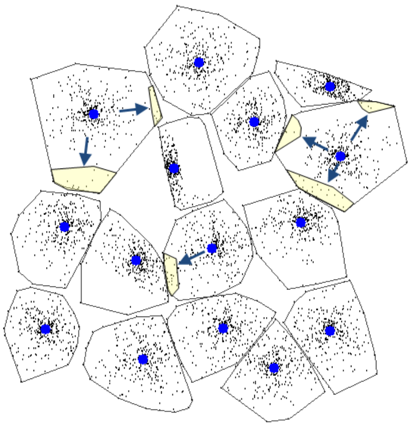

Fig. 3. Sample clustering result. Most significant differences between balanced clustering and standard k-means (non-balanced) clustering are marked and pointed out by arrows.

Table 2. MSE, standard deviation of MSE and time/run of 100 runs

Dataset	Size	Clusters	Algorithm	Best	Mean	St.dev.	Time
s2	5000	15	Balanced k-means	**2.86**	(one run)	(one run)	**1h 40min**
			Constrained k-means	-	-	-	-
s1 subset	1000	15	Balanced k-means	2.89	(one run)	(one run)	**47s**
			Constrained k-means	**2.61**	(one run)	(one run)	26min
s1 subset	500	15	Balanced k-means	3.48	3.73	0.21	**8s**
			Constrained k-means	**3.34**	**3.36**	0.16	30s
			K-means	2.54	4.21	1.19	0.01s
s1 subset	500	7	Balanced k-means	14.2	15.7	1.7	10s
			Constrained k-means	**14.1**	**15.6**	1.6	**8s**
s2 subset	500	15	Balanced k-means	3.60	3.77	0.12	**8s**
			Constrained k-means	**3.42**	**3.43**	0.08	29s
s3 subset	500	15	Balanced k-means	3.60	3.69	0.17	**9s**
			Constrained k-means	**3.55**	**3.57**	0.12	35s
s4 subset	500	15	Balanced k-means	3.46	3.61	1.68	**12s**
			Constrained k-means	**3.42**	**3.53**	0.20	45s
thyroid	215	2	Balanced k-means	**4.00**	**4.00**	0.001	2.5s
			Constrained k-means	**4.00**	**4.00**	0.001	**0.25s**
wine	178	3	Balanced k-means	**3.31**	3.33	0.031	0.36s
			Constrained k-means	**3.31**	**3.31**	0.000	**0.12s**
iris	150	3	Balanced k-means	**9.35**	3.39	0.43	0.34s
			Constrained k-means	**9.35**	**3.35**	0.001	**0.14s**

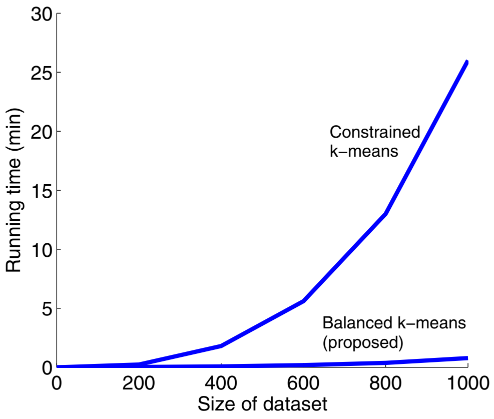

Fig. 4. Running time with different-sized subsets of s1 dataset

4 Experiments

In the experiments we use artificial datasets s1-s4, which have Gaussian clusters with increasing overlap and real-world datasets thyroid, wine and iris. The source of the datasets is `http://cs.uef.fi/sipu/datasets/`. As a platform, Intel Core i5-3470 3.20GHz processor was used. We have been able to cluster datasets of size 5000 points. One example partitioning can be seen in Figure 3, for which the running time was 1h40min. Comparison of MSE values of constrained k-means and balanced k-means is shown in Table 2, running times in Figure 4. The results indicate that constrained k-means gives slightly better MSE in many cases, but that balanced k-means is significantly faster when the size of dataset increases. For dataset of size 5000 constrained k-means could no longer provide result within one day. The difference in MSE is most likely due to the fact that balanced k-means strictly forces balance within ± 1 points, but constrained k-means does not. It may happen, that constrained k-means has many clusters of size $\lfloor n/k \rfloor$, but some smaller amount of clusters of size bigger than $\lceil n/k \rceil$.

5 Conclusions

We have presented balanced k-means clustering algorithm which guarantees equal-sized clusters. The algorithm is a special case of constrained k-means, where cluster sizes are equal, but much faster. The experimental results show that the balanced k-means gives slightly higher MSE-values to that of the constrained k-means, but about 3 times faster already for small datasets. Balanced k-means is able to cluster bigger datasets than constrained k-means. However, even the proposed method may still be too slow for practical application and therefore, our future work will focus on finding some faster sub-optimal algorithm for the assignment step.

References

1. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of Euclidean sum-of-squares clustering. Mach. Learn. 75, 245–248 (2009)
2. Althoff, C.T., Ulges, A., Dengel, A.: Balanced clustering for content-based image browsing. In: GI-Informatiktage 2011. Gesellschaft für Informatik e.V. (March 2011)
3. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: SODA 2007: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia (2007)
4. Banerjee, A., Ghosh, J.: On scaling up balanced clustering algorithms. In: Proceedings of the SIAM International Conference on Data Mining, pp. 333–349 (2002)
5. Banerjee, A., Ghosh, J.: Frequency sensitive competitive learning for balanced clustering on high-dimensional hyperspheres. IEEE Transactions on Neural Networks 15, 719 (2004)
6. Bradley, P.S., Bennett, K.P., Demiriz, A.: Constrained k-means clustering. Tech. rep., MSR-TR-2000-65, Microsoft Research (2000)
7. Burkhard, R., Dell'Amico, M., Martello, S.: Assignment Problems (Revised reprint). SIAM (2012)
8. Chen, Y., Zhang, Y., Ji, X.: Size regularized cut for data clustering. In: Advances in Neural Information Processing Systems (2005)
9. Demiriz, A., Bennett, K.P., Bradley, P.S.: Using assignment constraints to avoid empty clusters in k-means clustering. In: Basu, S., Davidson, I., Wagstaff, K. (eds.) Constrained Clustering: Advances in Algorithms, Theory, and Applications. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series (2008)
10. Equitz, W.H.: A New Vector Quantization Clustering Algorithm. IEEE Trans. Acoust., Speech, Signal Processing 37, 1568–1575 (1989)
11. Fränti, P., Kivijärvi, J.: Randomized local search algorithm for the clustering problem. Pattern Anal. Appl. 3(4), 358–369 (2000)
12. Fränti, P., Virmajoki, O.: Iterative shrinking method for clustering problems. Pattern Recognition 39(5), 761–765 (2006)
13. Fränti, P., Virmajoki, O., Hautamäki, V.: Fast agglomerative clustering using a k-nearest neighbor graph. IEEE Trans. on Pattern Analysis and Machine Intelligence 28(11), 1875–1881 (2006)
14. Hagen, L., Kahng, A.B.: New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on Computer-Aided Design 11(9), 1074–1085 (1992)
15. Karmarkar, N.: A new polynomial time algorithm for linear programming. Combinatorica 4(4), 373–395 (1984)
16. Kawahara, Y., Nagano, K., Okamoto, Y.: Submodular fractional programming for balanced clustering. Pattern Recognition Letters 32(2), 235–243 (2011)
17. Liao, Y., Qi, H., Li, W.: Load-Balanced Clustering Algorithm With Distributed Self-Organization for Wireless Sensor Networks. IEEE Sensors Journal 13(5), 1498–1506 (2013)
18. Likas, A., Vlassis, N., Verbeek, J.: The global k-means clustering algorithm. Pattern Recognition 36, 451–461 (2003)
19. MacQueen, J.: Some methods of classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symp. Mathemat. Statist. Probability, vol. 1, pp. 281–296 (1967)

20. Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., Parthiban, P.: Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. International Journal of Nonlinear Science 9(2), 171–177 (2010)
21. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 727–734. Morgan Kaufmann, San Francisco (2000)
22. Strang, G.: Karmarkars algorithm and its place in applied mathematics. The Mathematical Intelligencer 9(2), 4–10 (1987)
23. Yao, L., Cui, X., Wang, M.: An energy-balanced clustering routing algorithm for wireless sensor networks. In: 2009 WRI World Congress on Computer Science and Information Engineering, vol. 3. IEEE (2006)
24. Zhu, S., Wang, D., Li, T.: Data clustering with size constraints. Knowledge-Based Systems 23(8), 883–889 (2010)