Automatic Cluster Number Selection using a Split and Merge K-Means Approach

Markus Muhr
Knowledge Relationship Discovery
Know-Center Graz
Graz, Austria
mmuhr@know-center.at

Michael Granitzer)
Institute of Knowledge Management
Technical University Graz
Graz, Austria
mgranitzer@tugraz.at

***Abstract*—The k-means method is a simple and fast clustering technique that exhibits the problem of specifying the optimal number of clusters preliminarily. We address the problem of cluster number selection by using a k-means approach that exploits local changes of internal validity indices to split or merge clusters. Our split and merge k-means issues criterion functions to select clusters to be split or merged and fitness assessments on cluster structure changes. Experiments on standard test data sets show that this approach selects an accurate number of clusters with reasonable runtime and accuracy.**

***Keywords*-k-means; validity indices; cluster number selection; split and merge;**

I. Introduction

Document clustering is widely used in text information retrieval systems to enhance retrieval models or to provide richer navigation facilities. However, the high-dimensional, very sparse, large-scale nature of text data limits the number of applicable algorithms and optimization criteria. For document clustering, standard algorithms like *hierarchical agglomerative clustering* methods have been shown to be outperformed by *partititional algorithms* like k-means [1] in terms of runtime and precision [2], [3]. Furthermore, the evaluation of different cost functions in [2] clearly favors the maximization of the average *cosine similarity*.

The simple, yet powerful form of representation commonly used with the cosine similarity is the bag-of-word based vector space model. In this field, the direction of a vector is more important than its length, which yields to the *spherical k-means algorithm* [4], [5]. Spherical k-means algorithms operate on the sphere around the origin, or, viewed in a data centric sense, on input data normalized to unit-length. Another precision improvement could be achieved in [6], where online updates outperformed the more commonly used batch k-means variants.

Despite of their good performance, (spherical) k-means algorithms exhibit the problem that the *cluster number must be known a-priori*. Pelleg addressed this issue by developing a dynamic k-means variant known as *x-means* [7]. X-means performs bisecting splits of clusters where the resulting clustering solution is assessed by the *Bayesian Information Criterion (BIC)*. However, in [7] x-means issues only euclidean distance and does not take into account other existing validity indices (e.g. Calinski-Harabash, Hartigan and Krzanowski-Lai [8]).

ISODATA [9], another k-means variant, guesses the number of clusters by using splitting *and* merging. However, this algorithm does not measure the fitness of splits or merges via well defined criteria, but uses several size based thresholds to split or merge clusters.

In this work, we combine ISODATA and x-means by incorporating a *merge step into x-means* as well as *online updates*. Changes in the cluster structure that yield from split and merge steps are assessed by using *different internal validity indices*, not only the BIC criterion. Our experiments, conducted on selected Cluto datasets [2], show, that the *accurate number of clusters can be found* automatically with reasonable runtime and the accuracy is comparable to the results obtained in [2].

With this work we add the following contributions to the field of document clustering:

- We extend the x-means algorithm with merge steps, online updates and extend it to spherical optimization criteria.
- We perform an in-depth evaluation on standard document clustering data sets comparing several validity indices on the extended x-means.
- Finally, we heuristically adapt the BIC criterion which improves clustering precision further.

The rest is organized as follows. Section II depicts the split and merge k-means in detail, followed by the discussion on validity indices in section III. In sections IV and V we report the experimental setup and the results of our experiments on real-life datasets. At the end, section VI concludes our work.

II. Split and Merge Spherical K-Means

In contrast to the classical, distance based k-means algorithms, spherical k-means methods maximize the cosine similarity between the data samples and their most similar centroid. Calculating the cosine similarity is equivalent to calculating the inner product on a normalized set of vectors. More formally, given a set of data samples $\mathcal{X} = \{x_1, \ldots, x_N\}$ with $x_i \in \Re^d$ being the bag-of-word vector representation of the i^{th} document and given a set of cluster

$\mathcal{C} = \{c_1, \ldots, c_K\}$ with $c_j \in \Re^d$ being the cluster centroid of the j^{th} cluster, the optimization criterion is given as

$$L = \sum_{i=1}^{N} x_i^T c_{y_i} \quad (1)$$

where $y_i = \text{argmax}_{1 \leq k \leq K} x_i^T c_k$ with $||x_i|| = 1$ as well as $||c_k|| = 1$. The hard assignment of samples to cluster is denoted as set $\mathcal{Y} = \{y_1, \ldots, y_N\}$.

Algorithm 1 Batch Spherical K-Means

Require: $\mathcal{X}$, K, T
Ensure: $\mathcal{C}$, $\mathcal{Y}$
initialize centroids $c_1, \ldots, c_K$
for $t = 1$ to T **do**
  for $n = 1$ to N **do**
    $y_n = \arg\max_{1 \leq k \leq K} x_n^T c_k$
  for $k = 1$ to K **do**
    $c_k = \sum_{x_i \in \mathcal{X}_k} x_i$ where $\mathcal{X}_k = \{x_n | y_n = k\}$
    $c_k = \frac{c_k}{\|c_k\|}$

Cluster updates are done either in a batch or online manner. In the batch version (see algorithm 1), cluster vectors are updated and normalized to unit length after assigning all data samples, i.e. after calculating $\mathcal{Y}$. In contrast, the online version (see algorithm 2) performs the adaption of the winning centroid right after the assignment of a data sample; a competitive learning technique with a Winner-Take-All approach. Both algorithms terminate after a user-defined number of iterations T or if the amount of changes in the assignment of samples to cluster is below a given threshold. Also, both algorithms require the number of clusters K to be found a-priori.

Algorithm 2 Online Spherical K-Means

Require: $\mathcal{X}$,K, T
Ensure: $\mathcal{C}$,$\mathcal{Y}$
initialize centroids c_k
for all $t = 1$ to T **do**
  for all $n = 1$ to N **do**
    $y_n = \arg\max_{1 \leq k \leq K} x_n^T c_k$
    $c_{y_n} = \frac{c_{y_n} + \eta(x_n - c_{y_n})}{||c_{y_n} + \eta(x_n - c_{y_n})||}$

X-means [7] automatically finds the number of clusters by using bisecting k-means, combined with internal validity indices. Here, the bisecting k-means algorithm splits the data samples into two disjoint clusters by using batch or online k-means with $K = 2$. If the split increases the overall fitness measured by internal validity indices, the cluster is split and the bisecting k-means continues recursively. If none of the bisecting steps of each existing cluster leads to an improved result, the algorithm stops and returns the current clusters as result. Although good results can be achieved, errors made in early splits are propagated to subsequent splits.

Algorithm 3 Split and Merge K-Means

Require: $\mathcal{X}$,K, $s(\mathcal{C})$, $m(\mathcal{C})$, $v(\mathcal{C})$
Ensure: $\mathcal{C}$, $\mathcal{Y}$
$\mathcal{C}$ = k-means ($\mathcal{X}_t$, K)
repeat
  $c_s = s(\mathcal{C})$, $\mathcal{X}_s = \{x_n | y_n = s\}$
  $\{c_i, c_j\}$ = k-means ($\mathcal{X}_s$, $K = 2$)
  if $v(\mathcal{C}) > v(\mathcal{C}/c_s \cup \{c_i, c_j\})$ **then**
    $\mathcal{C} = \mathcal{C}/c_s \cup \{c_i, c_j\}$
until $|\mathcal{C}|$ is not changing
repeat
  $c_i, c_j = m(\mathcal{C})$
  $Y_j = Y_i$, $\mathcal{C} = \mathcal{C}/c_j$
  if $v(\mathcal{C}) > v_(\mathcal{C}/c_j)$ **then**
    $\mathcal{C} = \mathcal{C}/c_j$
until $|\mathcal{C}|$ is not changing
$\mathcal{C}$ = k-means ($\mathcal{X}_t$, $\mathcal{C}$)

To reduce this effect, we extended the x-means approach with a merge step, as depicted in algorithm 3. Basically, this *split and merge k-means* creates an initial partitioning through a first k-means step with a predefined number of clusters. Afterwards consecutive split and merge steps are invoked where the changes on the cluster result are assessed using some internal validity measure $v(\mathcal{C})$ like the Bayesian Information Criterion (BIC). Those split and merge steps are repeated until changes no longer improve the fitness. At the end of the algorithm, an optional k-means step can further refine the results of the dynamic updates. Note that the input parameter K is optional and per default 2, but the algorithm allows setting a preliminary expectation on the cluster number to reduce runtime.

In order to reduce the number of splits and merges, algorithm 3 also introduces a splitting criterion $s(\mathcal{C})$ and a merging criterion $m(\mathcal{C})$ for selecting the cluster to split or merge in a step. In our approach, $s(\mathcal{C})$ selects the cluster with the lowest average data sample similarity. Similarly, $m(\mathcal{C})$ selects the two most similar clusters as merging candidates. However preliminary experiments showed that this merge criterion favored the largest clusters to be merged. Hence, a penalty factor on the number of connected children was introduced. That is, take the fewer children of the clusters, take the square root of it and divide the similarity measure by this outcome.

The introduction of criterion functions to retrieve specific clusters that should be split or merged decreases the amount of investigated splits and merges considerable at probable cost of lessened precision.

III. Internal Validity Indices

Clustering precision strongly depends on the chosen internal validity indices to measure the result fitness. Note, that our selection of indices is based on some preliminary evaluations that ruled out common indices like Davis-Bouldin and Dunn-Index (see also [8]).

One well known measure is the Bayesian Information Criterion (BIC) [7]. Assuming a univariate distribution of within cluster sample distances, the BIC is given as

$$-\frac{n_i}{2}\log 2\pi - \frac{n_i m}{2}\log\sigma^2 - \frac{n_i - k}{2} + n_i \log\frac{n_i}{n} - \frac{k}{2}\log n \quad (2)$$

where n_i is the size of the i^{th} cluster, n the size of the data set, k the cluster number, m the dimension of the data, and $\sigma^2 = \frac{1}{n_i - k}\sum_i (x_i - c_{y_i})^2$

However, for our extended x-means algorithm preliminary experiments turned out that only two terms are of significance, reducing the BIC in its simplified, heuristic form to:

$$BIC_h = -\frac{n_i m}{2}\log\sigma^2 - \frac{k}{2}\log n \quad (3)$$

Besides BIC, multivariate indices can measure cluster quality based on the total scatter of between-cluster and within-cluster sum-of-squares, denoted as B_k respectively W_k. One such measure is the Calinski and Harabasz index [10] defined as

$$CH_k = \frac{tr(B_k)/(k-1)}{tr(W_k)/(n-k)} \quad (4)$$

with $tr(W_k)$ or $tr(B_k)$ being the trace of these matrices. Similarly, the Hartigan index [11] is defined as

$$H_k = \left(\frac{tr(W_k)}{tr(W_{k+1})} - 1\right)(n-k-1) \quad (5)$$

and the Krzanowski and Lai index [12] as

$$diff_k = (k-1)^{2/m} tr(W_{k-1}) - k^{2/m} tr(W_k) \quad (6)$$

$$KL_k = |diff_k| / |diff_{k+1}| \quad (7)$$

IV. Experimental Setup

We experimentally evaluated the performance of the split and merge k-means using the four internal validity indices on different real-world datasets in the field of text mining. In the remainder of this section we will describe the different data sets, our experimental methods and the experimental results.

The data sets for our experiments originate from the CLUTO clustering toolkit[1] and are described in table I. The data sets are grouped into three parts depending on the number of classes contained in each data set, which allows to analyze the performance on either few, medium or many classes.

[1] http://www.cs.umn.edu/~karypis/cluto

Name	Source	$\lvert\mathcal{X}\rvert$	# Classes
hit	S. J. M. (TREC)	2301	6
rev	S. J. M. (TREC)	4069	5
la1	LA Times (TREC)	3204	6
la2	LA Times (TREC)	3075	6
tr31	TREC	927	7
k1b	WebACE	2340	6
tr41	TREC	878	10
re0	Reuters-21578	1504	13
fbis	FBIS (TREC)	2463	17
k1a	WebACE	2340	20
wap	WebACe	1560	20
re1	Reuters-21578	1657	25

Table I
Real world data sets

The datasets have already been preprocessed by removing stop words and are stemmed using Porter's algorithm. All terms that occur only in one document were eliminated (see [2] for further details). In addition, we performed tf-idf weighting and normalized the documents to unit-length.

Cluster performance evaluation uses the well-known F-score measure which is defined as

$$F(L_r, S_i) = \frac{2 * R(L_r, S_i) * P(L_r, S_i)}{(R(L_r, S_i) + P(L_r, S_i))} \quad (8)$$

where L_r defines a particular class of size n_r and S_i a particular cluster of size n_i. $R(L_r, S_i)$ is the recall value defined as n_{ri}/n_r and $P(L_r, S_i)$ is the precision value defined as n_{ri}/n_i. By taking the maximum over all clusters the final F-score is given as

$$FScore = \sum_{r=1}^{C} \frac{n_r}{n} \operatorname{argmax}_{S_i \in T} F(L_r, S_i) \quad (9)$$

We carried out multiple experiments with different parameter settings on all 12 data sets. Firstly, all introduced validity indices are evaluated to assess their impact on splitting or merging quality including a comparison between online and batch updates. Secondly, we evaluated whether our approach dynamically finds the correct number of clusters independent from the initial k-means cluster number. Therefore, we evaluated three different settings on the initial numbers of clusters. For all datasets with less or equals 10 classes, the minimum number of clusters is set to 2 and the maximum number to 15. Furthermore, the initial cluster number is either set to 2, 8 or 15. For all other datasets the minimum number of clusters is set to 5 and the maximum to 35 with initial cluster numbers of 5, 15, and 35.

As seeding mechanism we used the directed random seeding, also known as k-means++, introduced in [13].

Regarding online update learning rate η, preliminary experiments showed that in case of online updates the update factor starting at 0.2 and using the square root of the connected elements as decreasing factor yields best results. We kept this setting throughout the experiments.

data	ind.	f_μ (k_μ)	f_σ (k_σ)	f_m (k_m)
hit	KL	0.52 (5.4)	0.05 (0.97)	0.6 (5)
rev	BIC_h	0.72 (5.6)	0.05 (1.26)	0.77 (5)
la1	BIC_h	0.68 (5.9)	0.08 (1.79)	0.79 (5)
la2	BIC_h	0.71 (10)	0.06 (1.25)	0.76 (10)
tr31	CH	0.78 (7.9)	0.06 (0.74)	0.87 (7)
k1b	KL	0.78 (6)	0.09 (0.82)	0.91 (5)
tr41	H	0.66 (9.9)	0.04 (2.42)	0.74 (10)
re0	CH	0.51 (12.2)	0.02 (0.42)	0.53 (12)
fbis	CH	0.63 (14.1)	0.04 (1.2)	0.68 (14)
k1a	$Hart$	0.56 (16.1)	0.04 (1.91)	0.62 (13)
wap	KL	0.55 (18.5)	0.03 (0.71)	0.61 (18)
re1	KL	0.52 (26.9)	0.04 (2.13)	0.57 (26)

Table II
BEST RESULTS USING BATCH UPDATES

data	ind.	f_μ (k_μ)	f_σ (k_σ)	f_m (k_m)
hit	BIC_h	0.55 (5.4)	0.02 (0.84)	0.59 (6)
rev	KL	0.73 (5.6)	0.06 (0.84)	0.78 (5)
la1	BIC_h	0.72 (5.6)	0.06 (0.7)	0.8 (6)
la2	CH	0.75 (5.4)	0.04 (0.52)	0.79 (5)
tr31	CH	0.82 (7.5)	0.06 (1.18)	0.92 (7)
k1b	CH	0.85 (5.4)	0.04 (0.52)	0.93 (6)
tr41	BIC	0.7 (13.1)	0.06 (0.99)	0.84 (12)
re0	BIC_h	0.52 (10.5)	0.05 (4.35)	0.57 (8)
fbis	CH	0.63 (15.9)	0.03 (2.33)	0.68 (17)
k1a	BIC	0.57 (16.7)	0.04 (2)	0.64 (19)
wap	KL	0.56 (18.8)	0.04 (0.92)	0.62 (20)
re1	H	0.55 (30.9)	0.02 (1.29)	0.58 (29)

Table III
BEST RESULTS USING ONLINE UPDATES

To overcome seeding problems, 10 runs are conducted for each parameter setting and we report mean f-score f_μ, standard deviation f_σ, and maximum f-score f_m. Similarly, we report k_μ, k_σ, and k_m indicating mean, standard deviation and maximum number of clusters found.

V. RESULTS AND DISCUSSION

In table II the best results for each data set using batch updates are shown and in table III the best ones for online update. The abbrevations for the indices are directly taken from section III.

First of all, we compare our results to the bisecting k-means approach on the same data set with predefined, correct cluster number reported in [2]. Noteworthy, in [2] only the best result over several runs are shown, corresponding to our f_m, without a mean and standard deviation. Using the best result only, our variant achieves better results for k1b, but slightly worse performance on the Reuter splits re0 and re1. Overall the f-scores seem to be comparable, so that our approach can compete with the bisecting k-means using a predefined cluster number.

Next, we investigate if the found cluster numbers are in the range of the given class numbers. Results confirm, that in most cases the found cluster number is close to the class number. Especially in cases of few classes (5 - 7) the mean cluster number is 11 out of 12 within a deviation of one class. On the k1b data set for example, the mean of the cluster number is 5.4 with the best result at the real class number of 6. It is of course more likely that the class number can be guessed in small class problems as well as for problems that can be better separated by k-means algorithms. For data sets with more classes the deviation of the cluster number to the class number increases, but nevertheless in most cases the cluster number is quite near the class number. Sometimes it fails completely like for tr41 or re0, but for fbis the cluster number is at 15.9 with best result achieved at 17 which is exactly the class number.

Now, we look at the different validity indices and their suitability to assess splits and merges to provide a dynamic cluster structure. The results show that although each index is likely to provide a good criterion to assess the fitness the heuristically simplified BIC criterion as well as the adapted Calinski-Harabasz index seem to provide mostly the best results.

Finally, we point out that the initial setting on the cluster number for the first k-means run does not alter very much the final cluster number. In other words starting at 2, 6 or 15 for the few class problems does lead to similar results. However, one must mention that if the initial cluster number is much lower than the contained count of classes, the final cluster number underestimates it most likely. The same applies if the initial cluster number is much higher than the class number, so that an overestimates happens. For this reason, it is better to set the initial cluster number in the near range of the correct cluster number, if this information is available.

Finally, we want to investigate the runtime behavior of our algorithm. Compared to x-means our approach is faster, because we have fewer split steps (one compared to k steps for a current cluster number for x-means) and the merge steps are computationally less complex. Furthermore, the calculations of the validity indices are also computationally cheap through caching distances, so that our algorithm is not really much slower than a simple bisecting k-means with a prespecified cluster number.

However, there is a considerable difference between using our approach with batch or online updates. Using the geometrical correct online update ($c_{y_n} = \frac{c_{y_n}+\eta(x_n-c_{y_n})}{||c_{y_n}+\eta(x_n-c_{y_n})||}$) leads to a faster convergence of the algorithm, but the actual runtime is mostly higher compared to batch updates. This is a domain problem, because of the fact that text data is highly sparse, implementations can optimize a simple batch update by looking only at non-zero dimensions of the documents to sum up to a final centroid. In case of online updates, each non-zero dimension of the centroid that tends to be dense must be updated in each step. Fortunately, there exists heuristics in the spherical scenario to boost the performance to get a comparable runtime results as batch k-means [14]. However, we wanted to stick to the geometrical correct

update in this first approach.

Exemplarily, the runtime for one of the larger problems like *rev* (4069 docs 23220 feats) is 3 sec. for batch updates and 39 sec. for online updates, whereas for a smaller problem like for *tr31* (927 docs 10128 feats) we got 1.5 sec. compared with 3 sec.

VI. Conclusion & Future Work

We showed that our split and merge k-means reaches the goal of providing a clustering structure that dynamically selects its cluster number with an acceptable runtime and a favorable precision. Our approach can be highly effective to generate an initial clustering result with an automatically detected number of clusters as well as in incremental applications where the given cluster hierarchy should be updated dynamically as new documents are added or old documents are removed.

Furthermore, it has been shown that online updates are favorable to batch updates, but compared with their increased computational power might not been applicable for real world data in the field of text mining without using the referenced heuristics.

Concerning validity indices, the adapted Calinski-Harabasz index and the simplified Bayesian Information Criterion lead to the best result to assess the clustering fitness.

As a final remark, our split and merge approach seems to reach the goal of providing a clustering structure that dynamically selects its cluster number with an acceptable runtime and a favorable precision.

Acknowledgment

The Know-Center is funded within the Austrian COMET Program under the auspices of the Austrian Ministries BMVIT, BMWFJ and by the State of Styria, managed by the Austrian Research Promotion Agency FFG. Results have been partially developed in the RAVEN project, which is financed by the Austrian Research Promotion Agency within the strategic objective FIT-IT.

References

[1] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Wiley New York, 2001.

[2] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, McLean, Virginia, 2002, pp. 515–524.

[3] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," *In Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD'00)*, pp. 109–110, August 20–23 2000.

[4] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1/2, pp. 143–175, 2001.

[5] I. Dhillon, J. Fan, and Y. Guan, "Efficient clustering of very large document collections," in *Data Mining for Scientific and Engineering Applications*, R. Grossman, C. Kamath, and R. Naburu, Eds. Kluwer Academic Publishers, 2001.

[6] A. Banerjee and J. Ghosh, "Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 702–719, 2004.

[7] D. Pelleg and A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 727–734.

[8] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1650–1654, 2002.

[9] G. Ball and D. Hall, "A clustering technique for summarizing multivariate data," *Behaviorial Sciences*, vol. 12, no. 2, pp. 153–155, March 1967.

[10] R. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, no. 3, pp. 1–27, 1974.

[11] J. Hartigan, "Statistical theory in clustering," *Journal of Classification*, vol. 2, no. 1, pp. 63–76, December 1985.

[12] W. Krzanowski and Y. Lai, "A criterion for determining the number of groups in a dataset using sum of squares clustering." *Biometrics*, no. 44, pp. 23–34, December 1985.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding." in *SODA*, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035.

[14] S. Zhong, "Efficient online spherical k-means clustering," *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 5, pp. 3180–3185 vol. 5, July-4 Aug. 2005.