Removing redundant features via clustering: preliminary results in mental task separation

Renato Cordeiro de Amorim[1,2] and Boris Mirkin[2]

[1] Department of Computing, Glyndŵr University, Mold Road, Wrexham LL11 2AW, UK.
[2] Department of Computer Science and Information Systems, Birkbeck University of London, Malet Street, London WC1E 7HX
r.amorim@glyndwr.ac.uk, mirkin@dcs.bbk.ac.uk

Abstract. Recent clustering algorithms have been designed to take into account the degree of relevance of each feature, by automatically calculating their weights. However, as the tendency is to evaluate each feature at a time, these algorithms may have difficulties dealing with features containing similar information. Should this information be relevant, these algorithms would set high weights to all such features instead of removing some due to their redundant nature.
In this paper we introduce an unsupervised feature selection method that targets redundant features. Our method clusters similar features together and selects a subset of representative features for each cluster. This selection is based on the maximum information compression index between each feature and its respective cluster centroid.
We empirically validate out method by comparing with it with a popular unsupervised feature selection on three EEG data sets. We find that ours selects features that produce better cluster recovery, without the need for an extra user-defined parameter.

Keywords: Unsupervised feature selection, feature weighting, redundant features, clustering, mental task separation.

1 Introduction

Given a data set Y of n entities over m features $V = \{v_1, v_2, ..., v_m\}$, feature selection aims to reduce the cardinality of V by removing those features that are redundant or have no relevance to the task at hand. There are a number of reasons to motivate such reduction in V. For instance, (i) the amount of time a classification or clustering algorithm takes to process Y tends to be inversely proportional to the data set size and dimension; (ii) it reduces the chances of issues related to overfitting; (iii) it is possible that there will be a general improvement in the accuracy of predictions [23, 13].
Feature weighting is a generalization of feature selection. While the latter either selects or removes a feature from V, the former assigns a weight w in the interval $[0, 1]$ to each feature in V. This weight, w_v, aims to be directly proportional to the degree of relevance of feature v. Feature

weighting algorithms not only select features, by setting $w_v > 0$, but also take into account the intuitive idea that even among relevant features, there may be different degrees of relevance.

The concept of feature weighting has been applied to clustering algorithms. In this paper we are particularly interested in improving the cluster recovery of Weighted K-Means (WK-Means) [3], and its generalization, the intelligent Minkowski Weighted K-Means (iMWK-Means) [8], detailed in Section 2. Both algorithms optimize their results by assigning cluster dependent weights to each feature. This allows a given feature v to have different degrees of relevance at different clusters $k = \{1, 2, ..., K\}$, where K is the total number of clusters. WK-Means and iMWK-Means set each weight w_{kv} by following the intuitive assumption that features with a low relative dispersion in a particular cluster should have a high weight.

These two algorithms have proven themselves in several publications [3, 14, 15, 8, 7]. However, they do introduce a new drawback. Both WK-Means and iMWK-Means set w_{kv} by evaluating one feature at a time. Therefore, should a subset of features in V contain the same relevant information, none will be excluded by receiving a weight of zero. In fact, since they will all have similar small dispersions, each of their weights will be equally high.

In this paper we address the above problem by introducing a novel, clustering-based, unsupervised feature selection algorithm used to remove redundant features from V, we call it the intelligent K-Means for Feature Selection (iKFS). Our method creates clusters of similar features in V and selects representative features from each cluster. The iKFS algorithm is based on the use of a clustering algorithm called intelligent K-Means [22] and the maximum compression index [23]. We find iKFS to be an excellent pre-processing step for feature weighting clustering algorithms such as WK-Means and iMWK-Means.

We evaluate our method by clustering three data sets of Electroencephalography (EEG) signals with WK-Means and iMWK-Means, using the features selected by iKFS. These data sets contain 5,680 features each, with patterns that are difficult to discern. For comparison we run similar experiments using the features selected using the popular feature selection using feature similarity (FSFS) [23]. We find that iKFS tends to select a smaller amount of features that are in fact more relevant than those selected by FSFS, with the added benefit that iKFS does not require an extra user-defined parameter.

2 Background

One of the objectives of our proposed method, iKFS, is to find clusters of similar features in a dataset. Clustering is the non-trivial task of creating K groups of entities so that those within the same group are similar and those between groups are dissimilar. Clustering algorithms have been used to solve problems in various fields of research, such as data mining, computer vision, bioinformatics, text mining, etc [16, 22, 25, 29].

K-Means [1, 17] is among the most popular clustering algorithms. It performs partitional clustering, dividing a data set Y into K disjoint clusters

$S = \{S_1, S_2, ..., S_K\}$. K-Means represents a given cluster S_k by its centre of gravity, the centroid c_k, equivalent to the average of each entity $y_i \in S_k$, assuming Euclidean distance. K-Means iteratively minimizes the the sum of the distances between each entity $y_i \in Y$ and its respective centroid.

$$W(S, C) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} (y_{iv} - c_{kv})^2, \quad (1)$$

where $C = \{c_1, c_2, ..., c_K\}$. The popularity of K-Means has various sources. It is a relatively fast, easy to implement algorithm, which is also intuitively easy to understand. In fact, the minimization of the K-Means criterion (1) has only three steps: 1. assign the values of K random entities from Y to the initial centroids $c_1, c_2, ..., c_K$; 2. assign each entity $y_i \in Y$ to the cluster represented by its closest centroid; 3. update each centroid to the centre of gravity of its cluster, and go back to Step 2. Iterations cease when the algorithm converges.

The complexity of K-Means is of $\mathcal{O}(nKt)$, where t is the number of iterations K-Means takes to converge, and n the number of entities in Y. Although it is difficult to determine the value of t beforehand, we have shown that this tends to be small, particularly when K-Means is initialized with relevant, rather than random, centroids [6]. Other clustering algorithms can be much slower, for instance hierarchical algorithms have a complexity of at least $\mathcal{O}(n^2)$. Implementations of K-Means can be frequently found in software packages, such as MATLAB, R, SPSS, etc.

Although popular, K-Means does have drawbacks. Some of which have been target of research effort for a long time. For instance, K-Means requires K (the number of clusters in Y) to be known beforehand, and the clustering produced by K-Means can be heavily affected by the initial centroids used in its first step [28, 4, 24, 26, 18, 2].

Among the many algorithms addressing these two interrelated issues, intelligent K-Means (iK-Means) seems quite successful [22, 6, 4]. This algorithm finds the clusters in a data set by extracting one anomalous pattern at a time, as per below.

1. Set c_c, the centre of the data set Y.
2. Set a tentative centroid c_t, the entity $y_i \in Y$ that is the farthest from c_c.
3. Run K-Means on Y, using c_c and c_t as initial centroids. Do not allow c_c to move during the clustering.
4. If the $|S_{c_t}| \geq \theta$, add c_t to C_{init}, otherwise discard c_t. In any case, remove the entities in S_{c_t} from Y.
5. If there are entities in Y, go to Step 2.
6. Run K-Means, initialized with the centroids in C_{init} and $K = |C_{init}|$.

More recently research has also focused on the inability of K-Means to take into account that each feature $v \in V$ may have a different degree of relevance [11, 19]. Weighted K-Means (WK-Means) and the intelligent Minkowski Weighted K-Means (iMWK-Means), are of particular interest to us, because of their high ability to recover clusters [3, 8, 14, 15, 7].

The main difference between WK-Means and iMWK-Means is the distance measure in use. While the former applies the squared Euclidean distance, the latter makes a generalization of this by using the p^{th} root

of the Minkowski (L_p) distance. Both algorithms add a cluster dependent weight, w_{kv} to the distance in use. To avoid linearity, this weight is put to the power of an user-defined exponent. The distance measure between an entity $y_i \in Y$ and a centroid $c_k \in C$ in WK-Means is then $d(y_i, c_k) = \sum_{v \in V} w_{kv}^{\beta} |y_{iv} - c_{kv}|^2$. In iMWK-Means, the distance exponent is the same as the weight exponent, making it possible to interpret w_{kv} as a feature re-scaling factor, for any exponent. The distance used by iMWK-Means follows.

$$d_p(y_i, c_k) = \sum_{v \in V} w_{kv}^p |y_{iv} - c_{kv}|^p. \tag{2}$$

Substituting the distance in the K-Means criterion (1) by the adjusted weighted distance (2) we obtain the iMWK-Means criterion.

$$W_p(S, C, w) = \sum_{k=1}^{K} \sum_{y_i \in S_K} \sum_{v \in V} w_{kv}^p |y_{iv} - c_{kv}|^p. \tag{3}$$

While p is a user-defined parameter, the weights are not. We calculate the weights so that if a feature $v \in V$ has a smaller relative dispersion in cluster S_k than a different feature $u \in V$, then v should have a higher weight in S_k than u.

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kvp}/D_{kup}]^{1/(p-1)}}, \tag{4}$$

where the dispersion of v at cluster S_k and specific p is given by $D_{kvp} = \sum_{y_i \in S_k} |y_{iv} - c_{kv}|^p$. For a given cluster S_k, the weights are subject to $\sum_{v \in V} w_{kv} = 1$, and a crisp clustering, in which an entity $y_i \in Y$ can only be assigned to a single cluster S_k. The equations for WK-Means are similar to all the above, but with a distance exponent always equal to two. We formalise the iMWK-Means algorithm below.

1. Obtain the initial centroids $C = \{c_1, c_2, ..., c_K\}$ by applying the iK-Means algorithm, using the distance in Equation 2. Set each cluster $S_k \leftarrow \emptyset$.
2. Assign each entity $y_i \in Y$ to the cluster S_k represented by the closest c_k, using (2). Should there be no change in S, stop.
3. Update each centroid in C to the Minkowski centre of their respective clusters.
4. Update each weight w_{kv}, using Equation (4). Go to step 2.

The Minkowski centre of a given feature v for a cluster S_k can be found over each $y_i \in S_k$ by using a steepest descent algorithm [8]. The WK-Means algorithm applies the squared Euclidean distance. In this we still have a user-defined exponent to set, but this is solely a weight exponent, the distance exponent is always two. The iMWK-Means is initialized with a version of the intelligent K-Means algorithm [22], also using the weighted Minkowski distance (2).

The feature weighting procedure used by both WK-Means and iMWK-Means would not deal properly with a subset of V in which features contain relevant, but redundant information. Such features would have

similarly low dispersions. Since a single weight w_{kv} is calculated at a time these features would have their weights set to a similarly high value. We believe that this issue makes the removal of redundant features prior to the use of either WK-Means or iMWK-Means, beneficial.
Feature selection using feature similarity (FSFS) [23] is one of the most popular unsupervised algorithms that fits to our needs. In this the authors identify features containing similar information by using the maximum information compression index (MIC). This index is defined below, for the variables x and y.

$$2\lambda_2(x,y) = var(x)+var(y)-\sqrt{(var(x)+var(y))^2 - 4var(x)var(y)(1-p(x,y)^2)}, \quad (5)$$

where $p(x,y)$ is the correlation coefficient given by $\frac{cov(x,y)}{\sqrt{var(x)var(y)}}$. MIC has various interesting properties, such as being invariant to the rotation of the variables and to the translation of the data set [23]. FSFS applies MIC as applied as follows.

1. Choose an initial value for k, following the constrain $k \leq |V| - 1$. Set $R \leftarrow V$, standardise the features rather than entities.
2. For each feature $F_i \in R$, calculate r_i^k, the dissimilarity between F_i and its kth nearest neighbour feature in R, using Equation: 5.
3. Find the feature $F_{i'}$ for which r_i^k is minimum. Retain $F_{i'}$ and discard its k nearest features. Set $\varepsilon = r_{i'}^k$.
4. Adjust k in relation to the number of features. If $k < |R| - 1$, then $k = |R| - 1$.
5. If $k = 1$ stop and output R.
6. Adjust k in relation to the similarity. While $r_i^k > \varepsilon$
   (a) $k = k - 1$
   (b) $r_i^k = inf_{F_i \in R} r_i^k$
   (c) if $k = 1$ go to Step 5.
7. Go to Step 2.

The above is a popular and useful algorithm. However, we see two issues that deserve to be addressed: (i) FSFS does not take into account the structure of the data set while selecting features; (ii) FSFS requires the user to define a parameter, k, beforehand. This parameter may increase the algorithm's flexibility, but unfortunately we see no clear method to estimate it. These issues, added to the inability of WK-Means and iMWK-Means to deal with redundant features made us analyse the possibility of a clustering-based solution for feature selection that could be used as a pre-processing step. We present our method in the next section.

3 Algorithm

In this section we present our feature selection method, intelligent K-Means for feature selection (iKFS). Our aim is to cluster the features in V that are similar, rather than the entities. By assigning similar features to the same cluster we can identify and remove those that are redundant. Clearly there are issues to address under this framework, for instance: (i) how many clusters of feature there would be in a given data set Y;

(ii) given a cluster S_k of features, how many features should be selected from it.

Regarding issue (ii), it can be very tempting to keep a single feature from a cluster S_k of features, say the closest to the centroid c_k. However, we do not feel that each cluster should be treated the same, irrespective of its cardinality. With this in mind, we have decided to keep a subset of features of S_k, this subset cardinality is given by F_k.

$$F_k = \left\lceil \frac{|S_k|}{|Y|} * K \right\rceil, \tag{6}$$

where $|S_k|$ and $|Y|$ represent the cardinality of a given cluster of features S_k and the cardinality of the data set Y, respectively. One should note that since we are clustering features, the original data set has to be transposed, so the cardinality of Y is in fact the original number of features (Section 4 described experiments with 5,680 features). Equation (6) requires the number of clusters K to be known, taking us back to issue (i), its estimation. In our method we find K by using iK-Means, which can also be used to find good initial centroids for K-Means. The choice of iK-Means was based on its previous success as a clustering algorithm in different scenarios [22, 4, 7]. We introduce our method in full below.

1. Transpose the data set Y so that the original features become entities and then standardise the data set.
2. Apply the iK-Means algorithm setting $\theta = 0$.
3. For each cluster S_k, find F_k (Equation 6) features that have the highest maximum information compression (Equation 5) in relation to c_k. Put such features in R.
4. Output the features in R.

We are very interested in selecting features that are dissimilar to all others, such features will most likely become singletons during the clustering process. In order to avoid disregarding such features we set $\theta = 0$.

4 Experiments

Electroencephalography (EEG) signals are high-dimensional noise-prone signals that can be captured from a brain via a non-invasive procedure. There is considerable research supporting the belief that these signals contain information about the current state or intention of a subject's mind [12, 10, 5, 20, 9].

We have recorded data from three healthy subjects (A, B and C) for our experiments, using five bipolar electrodes (five channels), and a sampling frequency of 250Hz. These five electrodes were placed on the subjects head following the standard positions in the extended 10-20 system, using fc3 to pc3, fc1 to pc1, cz to pz, fc2 to pc2, and fc4 to pc4.

Our aim is to perform mental task separation, in other words, given a set of possible tasks we would like to know what particular task a subject is thinking about. We have three possible tasks: (i) movement of the left hand; (ii) movement of the right hand; (iii) movement of the feet. After visually suggesting what task the subject should be thinking about,

we recorded the EEG data for eight seconds, constituting a trial. Here we intend to cluster trials into the right tasks. Hence, the number of clusters is known to be three. We have gathered data from 240, 120 and 350 trials for each subject, respectively. The difference in the number of trials relates solely to the availability of subjects and staff.

We have pre-processed our data sets in two steps. First, we transformed the data into its power spectrum density (PSD). EEG patterns are normally found in the frequency space rather than amplitude, and PSD helps us to identify periodicities in the data. This transformation has been successfully applied in previous research [9, 10, 12, 5, 20].

Second, having a trial represented by 71 time-related samples each with 80 PSD-features, we generated a data matrix for each subject containing the respective number of trials (240, 120 and 350) over 5,680 features (71 x 80). We then standardised the data numerically.

$$y_{iv} = \frac{x_{iv} - \bar{x_v}}{0.5 * (max(x_v) - min(x_v))}, \tag{7}$$

where x_{iv} represents the PSD value of trial i in feature v, and $\bar{x_v}$ the average of feature v over all trials in the data set. The standard deviation is surely more popular in the standardization of data sets than the range. However, we have opted for the latter as the former favours unimodal distributions [21, 27].

The FSFS algorithm requires an user-defined parameter k. We have performed experiments with such parameter from 4,800 to 5,600 in steps of 100 for WK-Means and iMWK-Means independently. With this interval it is possible for FSFS to select a quantity of features close to that selected by iKFS. Note that the optimal k for WK-Means may not be the same as for iMWK-Means. Our method, iKFS, does not require any extra parameter, so the features used in WK-Means and iMWK-Means when the data is pre-processed with iKFS are exactly the same. Regarding the parameters required by the clustering algorithms themselves, WK-Means and iMWK-Means, we have run experiments from 1.0 to 5.0 in steps of 0.1. In this paper we do not deal with their estimation.

Since we have the labels for each trial in the data sets, we present the best possible results for each of these two algorithms in terms of their cluster recovery. This is calculated by using a confusion matrix.

We show the results of our experiments in Table 1. We are happy to see that in both algorithms, WK-Means and iMWK-Means, iKFS presents features that are more representative. This is visible thanks to the differences in cluster recovery when using FSFS and iKFS in both WK-Means and iMWK-Means. Table 1 also shows us that a much higher number of features does not necessarily means features that are more representative, nor better final accuracy.

Table 1 does not present average or standard deviation values for iMWK-Means because this is a deterministic algorithm, unlike WK-Means. This happens because iMWK-Means applies a version of iK-Means in its initialization, making it output the same clustering for a given data set irrespective of how many times it is run.

Table 1: Cluster recovery of WK-Means and iMWK-Means using the features selected by iKFS and FSFS. The number of features is in the parenthesis.

	Feature selection method	Exponent		Accuracy		
		Distance	Weight	Mean	Std	Max
Subject A						
WK-Means	FSFS (25)	2.0	3.9	48.2	2.5	52.1
WK-Means	iKFS (20)	2.0	1.5	54.1	1.4	56.2
iMWK-Means	FSFS (28)	3.2	3.2	-	-	51.2
iMWK-Means	iKFS (20)	4.5	4.5	-	-	59.2
Subject B						
WK-Means	FSFS (472)	2.0	4.7	59.2	4.6	73.3
WK-Means	iKFS (9)	2.0	3.6	68.5	4.7	76.7
iMWK-Means	FSFS (393)	2.0	2.0	-	-	65.0
iMWK-Means	iKFS (9)	2.5	2.5	-	-	66.7
Subject C						
WK-Means	FSFS (33)	2.0	4.7	39.6	1.8	42.3
WK-Means	iKFS (11)	2.0	4.5	56.5	0.3	56.9
iMWK-Means	FSFS (33)	4.6	4.6	-	-	42.3
iMWK-Means	iKFS (11)	1.8	1.8	-	-	58.6

5 Conclusion

Aiming to reduce the number of redundant features in a data set, this paper presents an algorithm called intelligent K-Means for feature selection (iKFS). Our method begins by clustering the features of a data set, rather than its entities, using iK-Means. The latter is particularly useful as it finds the number of clusters in a data set, as well as good initial centroids. We then select a representative amount of features from each cluster, based on the cluster relative cardinality and the features maximum information compression to the cluster centroid. Our method assumes that features clustered together will have a higher than usual degree of information redundancy among them.

We validate iKFS by performing experiments with data sets containing Electroencephalography (EEG) signals of three healthy subjects. We have chosen to use this type of data because of its high-dimensionality (5,680 features), and its widely acknowledge high-level of noise. The features selected by iKFS were used by two clustering algorithms that apply feature weighting, WK-Means [3] and iMWK-Means [8]. For comparison we have run similar experiments selecting features with feature selection using feature similarity (FSFS) [23].

Clustering algorithms that perform feature weighting, such as WK-Means and iMWK-Means, have difficulty dealing with redundant features. Given a set of relevant and redundant features these algorithms set similar weights to all, instead of removing some. We have found that pre-processing a data set by reducing the number of redundant features can be very ben-

eficial for such algorithms, particularly when this feature selection takes into account the data set structure - as its the case with iKFS.
Future research will aim to optimise even further the features selected by iKFS and apply this algorithm as a pre-processing step for a feature weighting clustering algorithm such as iMWK-Means in different scenarios.

References

1. G. H. Ball and D. J. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12(2):153–155, 1967.
2. M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.
3. E. Y. Chan, W. K. Ching, M. K. Ng, and J. Z. Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern recognition*, 37(5):943–952, 2004.
4. M. M. T. Chiang and B. Mirkin. Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads. *Journal of classification*, 27(1):3–40, 2010.
5. S. Chiappa and S. Bengio. Hmm and iohmm modeling of eeg rhythms for asynchronous bci systems. In *European Symposium on Artificial Neural Networks, ESANN*, pages 193–204, 2004.
6. R. C. de Amorim. An empirical evaluation of different initializations on the number of k-means iterations. *Lecture Notes in Computer Science*, 7629:15–26, 2013.
7. R. C. de Amorim and P. Komisarczuk. On initializations for the minkowski weighted k-means. *Lecture Notes in Computer Science*, 7619:45–55, 2012.
8. R. C. de Amorim and B. Mirkin. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition*, 45(3):1061–1075, 2012.
9. R. C. de Amorim, B. Mirkin, and J. Q. Gan. A method for classifying mental tasks in the space of eeg transforms. Technical report, Technical Report BBKS-10-01, Birkbeck University of London, London, 2010.
10. R. C. de Amorim, B. Mirkin, and J. Q. Gan. Anomalous pattern based clustering of mental tasks with subject independent learning–some preliminary results. *Artificial Intelligence Research*, 1(1):46–54, 2012.
11. H. Frigui and O. Nasraoui. Unsupervised learning of prototypes and attribute weights. *Pattern recognition*, 37(3):567–581, 2004.
12. J. Q. Gan. Self-adapting bci based on unsupervised learning. In *3rd International Workshop on Brain-Computer Interfaces*, pages 50–51, 2006.
13. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

14. J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.
15. J. Z. Huang, J. Xu, M. Ng, and Y. Ye. Weighting method for feature selection in k-means. *Computational Methods of Feature Selection*, pages 193–209, 2008.
16. A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
17. J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.
18. R. Maitra, A. D. Peterson, and A. P. Ghosh. A systematic evaluation of different methods for initializing the k-means clustering algorithm. *Transactions on Knowledge and Data Engineering*, pages 522–537, 2010.
19. V. Makarenkov and P. Legendre. Optimal variable weighting for ultrametric and additive trees and k-means partitioning: Methods and software. *Journal of Classification*, 18(2):245–271, 2001.
20. J. Millan and J. Mouriño. Asynchronous bci and local neural classifiers: an overview of the adaptive brain interface project. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):159–161, 2003.
21. G. W. Milligan and M. C. Cooper. A study of standardization of variables in cluster analysis. *Journal of Classification*, 5(2):181–204, 1988.
22. B. Mirkin. *Clustering for data mining: a data recovery approach*, volume 3. CRC Press, 2005.
23. P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.
24. J. M. Pena, J. A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.
25. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, pages 525–526. Boston, 2000.
26. D. Steinley and M. J. Brusco. Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification*, 24(1):99–121, 2007.
27. Douglas Steinley. Standardizing variables in k-means clustering. In *Classification, clustering, and data mining applications*, pages 53–60. Springer, 2004.
28. R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
29. R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.