Salem Alelyani, Jiliang Tang and Huan Liu

Feature Selection for Clustering: A Review

Contents

0.1 Introduction

The growth of the high-throughput technologies nowadays has led to exponential growth in the harvested data with respect to dimensionality and sample size. As a sequence, storing and processing these data becomes more challenging. Figure (1) shows the trend of this growth for UCI machine learning repository. This augmentation made manual processing for these datasets to be impractical. Therefore, data mining and machine learning tools were proposed to automating pattern recognition and knowledge discovery process. However, using data mining techniques on an ore data is mostly useless due to the high level of noise associated with collected samples. Usually, data noise is either due to imperfection in the technologies that collected the data or the nature of the source of this data itself. For instance, in medical images domain, any deficiency in the imaging device will be reflected as noise in the dataset later on. This kind of noise is caused by the device itself. On the other hand, text datasets crawled from the internet, are noisy by nature because they are usually informally written and suffer from grammatical mistakes, misspelling, and improper punctuation. Undoubtedly, extracting useful knowledge from such huge and noisy datasets is a painful task.

Dimensionality reduction is one popular technique to remove noisy (i.e. irrelevant) and redundant attributes (AKA features). Dimensionality reduction techniques can be categorized mainly into feature extraction and feature selection. In feature extraction approach, features are projected into a new space with lower dimensionality. Examples of feature extraction technique include Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), Singular Value Decomposition (SVD), to name a few. On the other hand, the feature selection approach aims to select a small subset of features that minimize redundancy and maximize relevance to the target (i.e. class label). Popular feature selection techniques include: Information Gain, Relief, Chi Squares, Fisher Score, and Lasso, to name a few.

Both dimensionality reduction approaches are capable of improving learning performance, lowering computational complexity, building better generalizable models, and decreasing required storage. However, feature selection is superior in terms of better readability and interpretability since it maintains the original feature values in the reduced space while feature extraction transforms the data from the original space into a new space with lower dimension, which cannot be linked to the features in the original space. Therefore, further analysis of the new space is problematic since there is no physical meaning for the transformed features obtained from feature extraction technique.

Feature selection is broadly categorized into four models, namely: filter model, wrapper model, embedded model and hybrid model. As we mentioned above, feature selection selects subset of highly discriminant features. In other words, it selects features that are capable of discriminating samples that belong to different classes. Thus, we need to have labeled samples as training samples in order to select these features. This kind of learning is called *supervised learning*, which means that the dataset is labeled. In supervised learning, it is easy to define what relevant feature means. It simply refers to the feature that is capable of distinguishing different classes. For example, a feature f_i is said to be relevant to a class c_j if f_i and c_j are highly correlated.

Unlabeled data poses a yet another challenge in feature selection. In such cases, defining relevancy becomes unclear. However, we still believe that selecting subset(s) of features may help improving *unsupervised learning* in a way similar to improving the supervised learning. One of the most utilized unsupervised learning technique is *data clustering*. Data clustering is the unsupervised classification of samples into groups. In other words, it is the

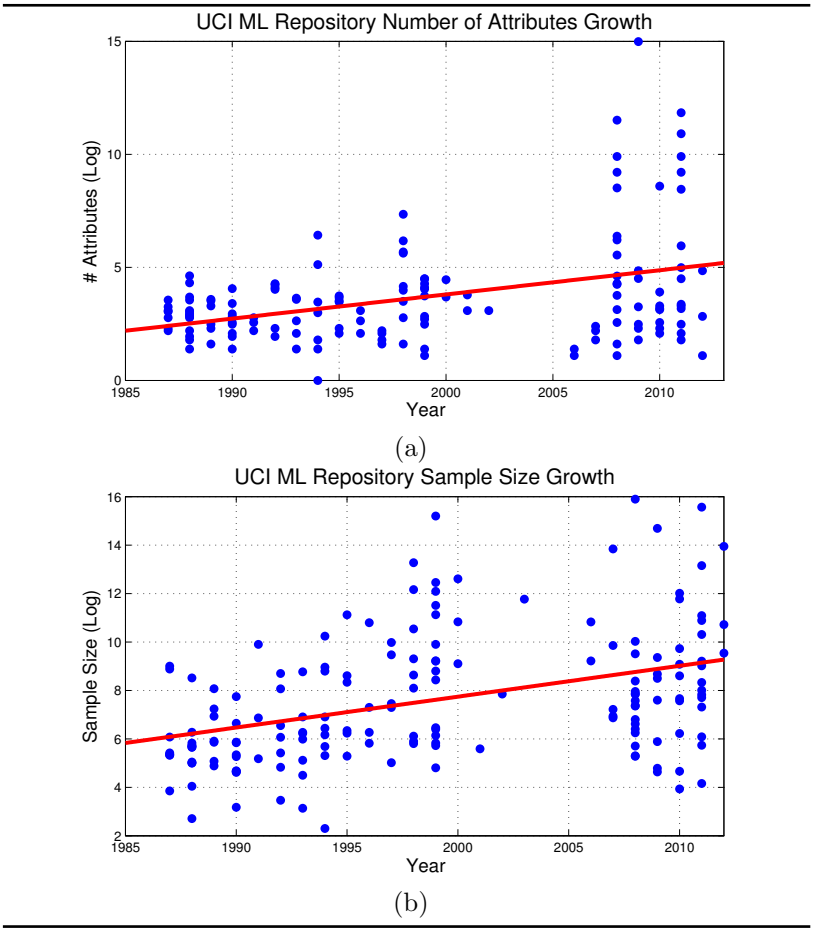


FIGURE 1: Plot (a) shows the dimensionality growth trend in UCI Machine Learning Repository from mid 80s to 2012 while (b) shows the growth in the sample size for the same period.

technique that aims to group similar samples into one group called a *cluster*. Each cluster has maximum within-cluster similarity and minimum between-cluster similarity based on certain similarity index. However, finding clusters in high-dimensional space is computationally expensive and may degrade the learning performance. Furthermore, equally good candidate of features' subsets may produce different clusters. Therefore, we demand to utilize feature selection for clustering to alleviate the effect of high-dimensionality.

In the following subsection, we will review the literature of data clustering in Section (0.1.1) followed by general discussion about feature selection models in Section (0.1.2) and feature selection for clustering in Section (0.1.3).

0.1.1 Data Clustering

Due to the increase in data size, human manual labeling has become extremely difficult and expensive. Therefore, automatic labeling has become indispensable step in data mining. Data clustering is one of the most popular data labeling techniques. In data clustering, we are given unlabeled data and we are to put similar samples in one pile, called a cluster, and

the dissimilar samples should be in different clusters. Usually, neither cluster's description nor its quantification is given in advance unless a domain knowledge exists, which poses a great challenge in data clustering.

Clustering is useful in several machine learning and data mining tasks including: image segmentation, information retrieval, pattern recognition, pattern classification, network analysis, and so on. It can be seen as either an exploratory task or preprocessing step. If the goal is to explore and reveal the hidden patterns in the data, clustering becomes a standalone exploratory task by itself. However, if the generated clusters are going to be used to facilitate another data mining or machine learning task, clustering will be a preprocessing step in this case.

There are many clustering methods in the literature. These methods can be categorized broadly into: partitioning methods, hierarchical methods, and density-based methods. The partitioning methods use a distance-based metric to cluster the points based on their similarity. Algorithms belonging to this type produce one level partitioning and non-overlapping spherical shaped clusters. K-means and k-medoids are popular partitioning algorithms. The hierarchical method, on the other hand, partitions the data into different levels that look at the end like a hierarchy. This kind of clustering helps in data visualization and summarization. Hierarchical clustering can be done in either bottom-up (i.e. agglomerative) fashion or top-down (i.e. divisive) fashion. Examples of this type of clustering are BIRCH, Chameleon, AGNES, DIANA. Unlike these two clustering techniques, density-based clustering can capture arbitrary shaped clusters such as S-shape. Data points in dense regions will form a cluster while data points from different clusters will be separated by low density regions. DBSCAN and OPTICS are popular examples of density-based clustering methods.

0.1.2 Feature Selection

In the past thirty years, the dimensionality of the data involved in machine learning and data mining tasks has increased explosively. Data with extremely high dimensionality has presented serious challenges to existing learning methods [38],known as the curse of dimensionality [25]. With the existence of a large number of features, a learning model tends to overfit and their learning performance degenerates. To address the problem of the curse of dimensionality, dimensionality reduction techniques have been studied, which form an important branch in the machine learning research area. Feature selection is one of the most used techniques to reduce dimensionality among practitioners. It aims to choose a small subset of the relevant features from the original ones according to certain relevance evaluation criterion [37, 23], which usually leads to better learning performance, e.g. higher learning accuracy, lower computational cost, and better model interpretability. Feature selection has been successfully applied in many real applications, such as pattern recognition [28, 58, 46], text categorization [74, 31, 52], Image processing [28, 56], bioinformatics [1, 2], and so forth.

According to whether the label information is utilized, different feature selection algorithms can be categorized into supervised [69, 60], unsupervised [18, 46], or semi-supervised algorithms [79, 73]. With respect to different selection strategies, feature selection algorithms can also be categorized as being of either the filter [39, 15], wrapper [33], hybrid, and embedded models [13, 51]. Feature selection algorithms of the filter model are independent of any classifier. They evaluate the relevance of a feature by studying its characteristics using certain statistical criteria. Relief [59], Fisher score [16], CFS [24], and FCBF [76] are among the most representative algorithms of the filter model. On the other hand, algorithms belonging to the wrapper model utilize a classifier as a selection criteria. In other words, they select a set of features that has the most discriminative power using a given classifier, such as: SVM, KNN, and so on. An example of the wrapper model is the FSSEM [17],

ℓ_1SVM [10]. Other examples of the wrapper model could be any combination of a preferred search strategy and given classifier. Since the wrapper model depends on a given classifier, cross-validation is usually required in the evaluation process. They are in general more computationally expensive and biased to the chosen classifier. Therefore, in real applications, the filter model is more popular, especially for problems with large datasets. However, the wrapper model has been empirically proven to be superior, in terms of classification accuracy, to those of a filter model. Due to these shortcomings in each model, the hybrid model [13, 40], was proposed to bridge the gap between the filter and wrapper models. First, it incorporates the statistical criteria, as filter model does, to select several candidate features subsets with a given cardinality. Second, it chooses the subset with the highest classification accuracy [40]. Thus, the hybrid model usually achieves both comparable accuracy to the wrapper and comparable efficiency to the filter model. Representative feature selection algorithms of hybrid model include: BBHFS [13], HGA [53]. Finally, the embedded model performs feature selection in the learning time. In other words, it achieves model fitting and feature selection simultaneously. Examples of embedded model include C4.5 [54], BlogReg [21], and SBMLR [21]. Based on different types of outputs, most feature selection algorithms fall into one of the three categories: subset selection [75], which returns a subset of selected features identified by the index of the feature; feature weighting [59], which returns weight corresponding to each feature; and the hybrid of subset selection and feature weighting, which returns a ranked subset of features.

Feature weighting, on the other hand, is thought of as a generalization of feature selection [70]. In feature selection, a feature is assigned a binary weight, where 1 means the feature is selected and 0 otherwise. However, feature weighting assigns a value, usually in the interval [0,1] or [-1,1], to each feature. The greater this value is, the more salient the feature will be. Feature weighting was found to outperform a feature selection in tasks where features vary in their relevance score [70], which is true in most real-world problems. Feature weighting could be, also, reduced to feature selection if a threshold is set to select features based on their weights. Therefore, most of feature selection algorithms mentioned in this chapter can be considered using feature weighting scheme.

Typically, a feature selection method consists of four basic steps [40], namely, subset generation, subset evaluation, stopping criterion, and result validation. In the first step, a candidate feature subset will be chosen based on a given search strategy, which is sent, in the second step, to be evaluated according to certain evaluation criterion. The subset that best fits the evaluation criterion will be chosen from all the candidates that have been evaluated after the stopping criterion are met. In the final step, the chosen subset will be validated using domain knowledge or validation set.

0.1.3 Feature Selection for Clustering

The existence of irrelevant features in the data set may degrade learning quality and consume more memory and computational time that could be saved if these features were removed. From the clustering point of view, removing irrelevant features will not negatively affect clustering accuracy whilst reducing required storage and computational time. Figure 2 illustrates this notion where (a) shows the relevant feature f_1 which can discriminate clusters. Figure 2(b) and (c) shows that f_2 and f_3 cannot distinguish the clusters; hence, they will not add any significant information to the clustering.

In addition, different relevant features may produce different clustering. Figure 3(a) shows four clusters by utilizing knowledge from f_1 and f_2, while Figure 3(b) shows two clusters if we use f_1 only. Similarly, Figure 3(c) shows two different clusters if we use f_2 instead.Therefore, different subset of relevant features may result in different clustering, which greatly help discovering different hidden patterns in the data.

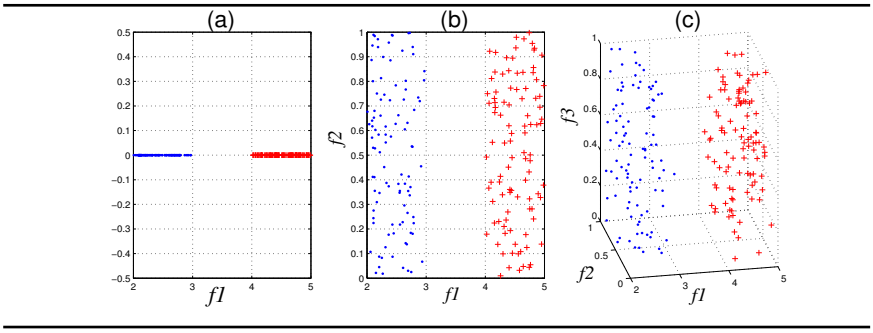


FIGURE 2: Feature f_1 is relevant while f_2 and f_3 are irrelevant. We are able to distinguish the two clusters from f_1 only. Thus, removing f_2 and f_3 will not effect the accuracy of clustering.

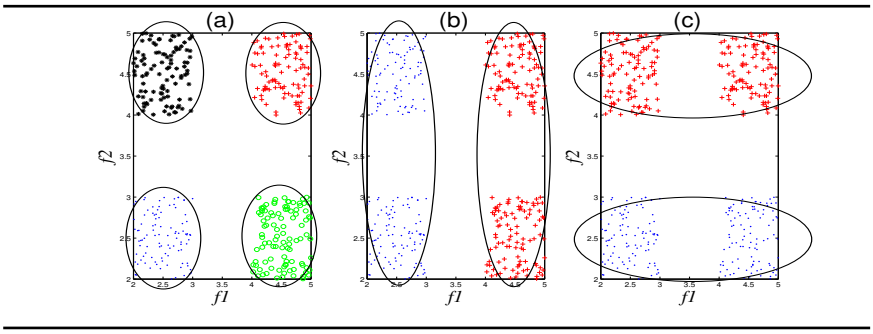


FIGURE 3: Different sets of features may produce different clustering.

Motivated by these facts, different clustering techniques were proposed to utilize feature selection methods that eliminate irrelevant and redundant features while keeping relevant features in order to improve clustering efficiency and quality. For simplicity and better organization, we are going to describe different feature selection for clustering (FSC) methods based on the domain. The following sections will be organized as follows: conventional FSC, FSC in text data, FSC in streaming data, and FSC link data.

Similar to feature selection for supervised learning, methods of feature selection for clustering are categorized into filter [15] wrapper [55], and hybrid models [19]. A wrapper model evaluates the candidate feature subsets by the quality of clustering while filter model is independent of clustering algorithm. Thus, the filter model is still preferable in terms of computational time and unbiased toward any clustering method, while the wrapper model produces better clustering if we know the clustering method in advance. To alleviate the computational cost in the wrapper model, filtering criteria are utilized to select the candidate feature subsets in the hybrid model.

In the following subsections, we will briefly discuss feature selection for clustering methods that falls in the filter, wrapper and hybrid models. For more about conventional methods, we refer the reader to [19].

0.1.3.1 Filter Model

Filter model methods do not utilize any clustering algorithm to test the quality of the features [19]. They evaluate the score of each feature according to certain criteria. Then,

TABLE 0.1: Nomenclature

D	Dataset
n	Sample size
m	Number of features
x_j	j^{th} sample
f_i	i^{th} feature
F	Selected feature set
l	Number of selected features
K	Number of clusters
C_k	k^{th} cluster

it selects the features with the highest score. It is called the filter since it filters out the irrelevant features using given criteria. Furthermore, feature evaluation could be either *univariate* or *multivariate*. Univariate means each feature is evaluated independently of the feature space. This approach is much faster and more efficient than the univariate, which evaluates features with respect to the other features. Therefore, the multivariate, unlike the univariate approach, is capable of handling redundant features. SPEC ,see Section (0.2.1.1), is an example of the univariate filter model, although it was extended to multivariate approach in [78]. Other examples of filter model criteria used in feature selection for clustering include: feature dependency [62], entropy-based distance [15], and laplacian score [26, 80].

0.1.3.2 Wrapper Model

The wrapper model utilizes a clustering algorithm to evaluate the quality of selected features. It starts by (1) finding a subset of features. Then, (2) it evaluates the clustering quality using the selected subset. Finally, it repeats (1) and (2) until the desired quality is found. Evaluating all possible subsets of features is impossible in high-dimensional datasets. Therefore, heuristic search strategy is adopted to reduce the search space. The wrapper model is very computationally expensive compared to filter model. Yet, it produces better clustering since we aim to select features that maximize the quality. It is still biased toward the used clustering method. Different wrapper feature selection methods for clustering were proposed by changing the combination of search strategy and the utilized clustering algorithm. The method proposed in [18] is an example of a wrapper that involves maximum likelihood criteria and feature selection and mixture of Gaussians as clustering method. Others use conventional clustering methods such as k-means and any search strategy as feature selector [32].

0.1.3.3 Hybrid Model

To overcome the drawback of filter and wrapper models a hybrid model is used to benefit from the efficient filtering criteria and better clustering quality from the wrapper model. A typical hybrid process goes through the following steps: (1) it utilizes filtering criteria to select different candidate subsets. Then, (2) it evaluates the quality of clustering of each candidate subsets. (3) The subset with highest clustering quality will be selected. Algorithms belonging to the hybrid model usually produce better clustering quality than those of filter model, yet, they are less efficient. Compared to the wrapper model, the hybrid model is much more efficient.

0.2 Feature Selection for Clustering

Several feature selection for clustering methods have been proposed in the literature. Some algorithms handle text data, while others handle streaming data. Still others are capable of handling different kind of data. In this section, we will discuss different methods with respect to data types they can handle. We will review algorithms for text, streaming and linked data, as well as algorithms that is able to handle generic data.

0.2.1 Algorithms for Generic Data

In this section, we will discuss feature selection for clustering methods that are able to handle generic datasets. In other words, it is not necessary to be designed to handle only text, data stream or linked data.

0.2.1.1 Spectral Feature Selection (SPEC)

Although the Spectral Feature selection (SPEC) algorithm is a unified framework that enables the joint study of supervised and unsupervised learning, we will use SPEC in this work as an example of filter-based unsupervised feature selection methods. SPEC [80] estimates the feature relevance by estimating feature consistency with the spectrum of a matrix derived from a similarity matrix S. SPEC uses the Radial-Bases Function RBF as a similarity function between two samples x_i and x_j:

$$S_{ij} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{0.1}$$

Graph G will be constructed from S and adjacency matrix W will be constructed from G. Then, degree matrix $\bar{D}$ will be computed from W. $\bar{D}$ is diagonal matrix where $\bar{D}_{ii} = \sum_{j=1}^{n} W_{ij}$. Given $\bar{D}$ and W Laplacian matrix L and the normalized Laplacian matrix $\mathcal{L}$ are computed as follows:

$$L = \bar{D} - W; \quad \mathcal{L} = \bar{D}^{-\frac{1}{2}} L \bar{D}^{-\frac{1}{2}} \tag{0.2}$$

The main idea behind SPEC is that the features consistent with the graph structure are assigned similar values to instances that are near to each other in the graph. Therefore, these features should be relevant since they behave similarly in each similar group of samples (i.e. clusters). Motivated by graph theory that states that graph structure information can be captured from its spectrum, SPEC studies how to select features according to the structure of the graph G induced from the samples similarity matrix S.

The weight of each feature f_i in SPEC is evaluated using three functions ψ_1, ψ_2, and ψ_3. These functions were derived from the normalized cut function with the spectrum of the graph, and extended to their more general forms. In this chapter, we will not explain these functions in detail, therefore, we refer the reader to [80] for more details. We assume here that each function ψ takes feature vector f_i and returns the weight based on the normalized Laplacian $\mathcal{L}$.

0.2.1.2 Laplacian Score (LS)

Laplacian Score (LS)[26] is a special case of SPEC if the ranking function used is:

$$F_i \leftarrow \frac{\hat{f}_i^T L \hat{f}_i}{\hat{f}_i^T \bar{D} \hat{f}_i} \quad where \quad \hat{f}_i = f_i - \frac{f_i^T \bar{D} \mathbf{1}}{\mathbf{1}^T \bar{D} \mathbf{1}} \mathbf{1} \tag{0.3}$$

Algorithm 1 Spectral Feature Selection (SPEC)

Input:
D: dataset
$\psi \in \{\psi_1, \psi_2, \psi_3\}$: feature weighting functions
n: number of samples
Output:
F: the ranked feature list

1: Construct similarity matrix S from D
2: Construct Graph G from S
3: Construct W from S
4: Construct $\bar{D}$ from W
5: Define L and $\mathcal{L}$ according to Eq (0.2)
6: **for each** feature vector f_i **do**
7: $\quad \hat{f}_i \leftarrow \frac{\bar{D}^{-\frac{1}{2}} f_i}{||\bar{D}^{-\frac{1}{2}} f_i||}$
8: $\quad F_i \leftarrow \psi(\hat{f}_i)$
9: **end for**
10: Rank F based on ψ

Where **1** is one vector. LS is very effective and efficient with respect to the data size. Similar to SPEC, the most time consuming in LS is constructing the similarity matrix S. The beauty of this algorithm is it can handle both labeled and unlabeled data.

0.2.1.3 Feature Selection for Sparse Clustering

Witten and Tibshirani in [71] proposed a framework for feature selection in sparse clustering. They applied Lasso-type , $\ell_1 - norm$, as feature selection method embedded in the clustering process. This framework can be applied to any similarity-based clustering technique, yet, they used k-means clustering in [71]. The number of selected features l is chosen using gap statistics in a similar fashion to choosing the number of clusters in [67]. The proposed method attempts to minimize the following objective function with respect to the clusters $\{C_1, \ldots, C_K\}$ and the feature weight vector w :

$$\begin{array}{ll} \min & \sum_{j=1}^{m} w_j \Psi_j \\ \textit{subject to} & ||w||^2 \leq 1, \\ & ||w||_1 \leq l, \\ & w_j \geq 0 \quad \forall j \end{array} \tag{0.4}$$

Where Ψ_j is given by the following equation for k-means over j^{th} feature:

$$\Psi_j = \sum_{c=1}^{K} \frac{1}{n_c} \sum_{i,i' \in C_K} Sim(i, i', j) - \frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} Sim(i, i', j).$$

K is the number of clusters, n_c is the number of samples in cluster c and $Sim(i, i', j)$ is the similarity index of sample i and i' using only the selected feature j. Optimizing Eq (0.4) is done using iterative algorithm by holding w fixed and optimizing Eq (0.4) with respect to the clusters $\{C_1, \ldots, C_K\}$. In this step, we apply standard k-means clustering on *n-by-n* similarity matrix using j^{th} feature. Then, we hold the clusters fixed and optimize Eq (0.4)

with respect to w. w is set in this step to be

$$w = \frac{S(\Psi_+, \Delta)}{S(||\Psi_+, \Delta)||_2} \tag{0.5}$$

where Ψ_+ is the positive part of Ψ and $\Delta = 0$ when $||w||_1 \leq l$ and $\Delta > 0$ otherwise, so, $||w||_1 = l$. Algorithm (2) illustrates these steps of optimizing Eq (0.4).

Algorithm 2 Feature Selection for Sparse Clustering

Input:
D: dataset
l: number of selected features obtained from gab statistics-like approach
n: number of samples
Output:
the clusters and w
Initialize:
$w_1 = w_2, \ldots, w_m = \frac{1}{\sqrt{m}}$

1: **while** not converge **do**
2: Hold w fixed
3: Optimize Eq (0.4) with respect to $C1, \ldots, C_K$
4: Holding $C_1, \ldots, C_K$ fixed
5: Optimize Eq (0.4) with respect to w by applying Eq (0.5)
6: **end while**

Witten and Tibshirani in [71], also, proposed sparse hierarchical clustering based on lasso penalty similar to Algorithm (2). The hierarchical clustering involves *n-by-n* similarity matrix, which is optimized iteratively with w. Then, we perform hierarchical clustering on the constructed similarity matrix. For more about this algorithm we refer the reader to [71].

0.2.1.4 Localized Feature Selection Based on Scatter Separability (LFSBSS)

Li et al in [35] proposed a localized feature selection based on scatter separability (LFSBSS). This is motivated by the fact that the set of features that are relevant to one clustering result are not necessary the same set that is relevant to another clustering result. In other words, clustering dataset D using a set of feature F_1 may produce clusters $\{C_1, C_2, C_3\}$ while clustering using another set of features F_2 may lead to clusters $\{C_4, C_5\}$, where $F_1 \neq F_2$. This notion is also illustrated in Figure (3). Furthermore, each cluster in a clustering result may be associated to different set of relevant features. In document clustering, for instance, documents that belong to sport news are more likely to have different set of relevant terms such as: FIFA, Ball, and so on. While the set of documents that belong to technology news contains relevant terms such as: Apple, IBM, and so on. In this section we are will use the cluster set $C = \{(C_1, F_1), \ldots, (C_j, F_j), \ldots, (C_K, F_K)\}$ to refer to the clustering result where C_1 and F_1 are the first cluster and the set of selected features corresponds to the first cluster, respectively.

Li et al in [35] borrowed the notion of scatter separability from Dy and Brodley [18] and adopted as a localized feature selection. They defined the scatter separability as:

$$\Omega = tr(S_w^{-1} S_b)$$

where S_w^{-1} is the inverse of within-cluster separability and S_b is the between-cluster separability. If we need to evaluate Ω_i for cluster i, we should use the within that cluster separability $S_w^{(i)-1}$, instead. It was proven in [35] that Ω_i is monotonically increasing with

dimensions as long as the clustering assignments remain the same. To mitigate this problem, separability criteria must be normalized with respect to the dimensionality for feature selection. Moreover, since localized feature selection attempts to select different sets of relevant features for each cluster, the between-cluster separability needs to be appropriately normalized as well. This is performed using cross-projecting over individual clusters. We assume that the projected cluster is $\hat{C} = \{(\hat{C_1}, \hat{F_1}), \ldots, (\hat{C_K}, \hat{F_K})\}$. At each step of the projection, we replace the projected cluster $\hat{C_i}$ with the largest overlap and the original cluster C_j to generate the new clustering $C^* = \{(C_1, F_1), \ldots, (\hat{C_i}, \hat{F_i}), \ldots, (C_K, F_K)\}$. Finally, we cross-project into each other, which generates the normalized value, v that allows us to compare two different clusters with different subspaces. Larger v indicates larger greater separability between clusters. More details about the projection may be found in [35].

LFSBSS reduces the impact of overlapping and unassigned data by penalizing using what they called adjusted normalized value a. This value penalizes the cross-projection if the amount of unsigned or overlap have increased in the projected clustered compared to the original clusters.

LFSBSS adopts the sequential backward feature selection. This means, that the clusters are generated first using the whole feature space. Then, iteratively removing irrelevant or noisy feature based on a from each cluster individually. Algorithm (3) illustrates the steps of LFSBSS.

Algorithm 3 Localized Feature Selection Based on Scatter Separability (LFSBSS)

Input:
D: dataset
l: number of selected features
n: number of samples
K: number of clusters
Output:
the clusters and their corresponding features sets
Initialize:
initialize C using all feature space F

$F'_1 = F'_2 = \cdots = F'_K = F$
while not converged **do**
  for $c = 1$ to K **do**
    Evaluate a for C_c
    Choose feature f_i to be removed based on a
    F_c=$F_c - f_i$
    Generate a new cluster set C' based on F_c
    Compare Cluster in C' with clusters in C
    if *BetterClusterFound* **then**
      Replace the corresponding cluster in C
    end if
  end for
  if *Desired* **then**
    Process unassigned samples
  end if
end while

0.2.1.5 Multi-Cluster Feature Selection (MCFS)

Similar to the motivation illustrated in Figure (3), Cai et al in [11] proposed a multi-cluster feature selection (MCFS) method that is able to select the set of features that can cover all the possible clustering in the data. In MCFS, spectral analysis is used to measure the correlation between different features without label information needed. Using the top eigenvectors of graph Laplacian, spectral clustering can cluster data samples without utilizing label information. Thus, MCFS applied k-Nearest-Neighbors approach to construct the graph of the data samples, where k is a predetermined parameter. Next, the heat kernel weighting matrix W is computed as follows:

$$W_{ij} = e^{-\frac{||x_i - x_j||^2}{\sigma}}$$

where x_i and x_j are connected samples in the k-Nearest-Neighbors graph and σ is used predefined parameter. From W a degree matrix is computed as explained earlier in this chapter. Then, a graph Lapalcian matrix $L = \bar{D} - W$ is constructed. After that, MCFS solves the following eigen-problem:

$$Ly = \lambda \bar{D} y \tag{0.6}$$

Given $Y = [y_1, \ldots, y_K]$, the eigenvectors of Eq (0.6), we can find a relevant subset of features by minimizing the following objective function:

$$\begin{array}{ll} \min_{a_k} & ||y_k - X^T a_k||^2 \\ s.t. & ||a_k||_0 = l \end{array}$$

Where a_k is a $m-$dimensional vector and the $||a_k||_0$ is the number of non-zero elements in a_k. Then, K sparse coefficient vectors will be chosen to correspond to each cluster. For each feature f_j, the maximum value of a_k that correspond to f_j will be chosen. Finally, MCFS will choose the top $l-$features. MCFS shows improvement over other methods such as LS according to [11].

0.2.1.6 Feature Weighting *k*-means

k-means clustering is one of the most popular clustering techniques. It has been extensively used in data mining and machine learning problems. Large number of k-means variations were proposed to handle feature selection [9, 68, 30, 27, 47]. Most of these variations start by clustering the data into k clusters. Then, it assigns weight to each feature. The feature that minimizes the within-cluster distance and maximizes between-cluster distance is preferred, hence, gets higher weight. In [30], for example, an entropy weighting k-means (EWKM) was proposed for subspace clustering. It simultaneously minimizes the within-cluster dispersion and maximize the negative weight entropy in the clustering process. EWKM calculates the weight of each feature in each cluster by including the weight entropy in the objective function of k-means. The subset of features corresponding to each cluster are, then, selected based on that weight. Thus, EWKM allows subspace clustering where the set of selected features may differ from one cluster to another.

In addition, [47] proposed feature weighting k-means clustering using generalized Fisher ratio that minimizes the ratio of the average of within-cluster distortion over the average between-cluster distortion. In this algorithm, several candidate clusterings are generated and the one with the minimal Fisher ratio is determined to be the final cluster.

Similarly, [27] proposed another variation of feature weighting k-means (W-k-means) that measures the weight of each feature based on its variance of the within-cluster distance. Algorithm 4 illustrates the process of W-k-means. It, iteratively, minimizes Eq. (0.7) by fixing two parameters at each step and solve Ψ with respect to the third one. If there is no change in Ψ after the minimization, the algorithm is said to be converged.

$$\Psi(C, Z, \mathbf{w}) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{il} w_j^{\beta} d(x_{ij}, z_{lj}) \tag{0.7}$$

Where C is *n-by-k* partition matrix that contains binary values, $c_{il} = 1$ indicates that x_i belongs to cluster l. Z is the centroids and $d(\cdot, \cdot)$ is the distance matrix. $\mathbf{w}$ is the weight vector and β is the parameter of the attribute weight. Minimizing Ψ with respect to C and Z is straight forward. However, minimizing Ψ with respect to $\mathbf{w}$ depends of the value of β. We refer the reader to [27] for more about minimizing with respect to $\mathbf{w}$

Algorithm 4 Feature Weighting k-means (W-k-means)

Input:
D: dataset
n: number of samples
Ψ: the objective function Eq. (0.7)
Initialize:
C: apply *k-means* on D to obtain initial clusters
Z: randomly choose k centroids
$\mathbf{w}$: randomly initialize the weight of each feature so that $\sum_{i=1}^{m} w_i = 1$
t: 0
Output:
C: the clustering
Z: the centroids
$\mathbf{w}$: the features weights

1: **while** not stop **do**
2: Fix Z and $\mathbf{w}$ and solve Ψ with respect to C.
3: Stop when no changes occur on C.
4: Fix C and $\mathbf{w}$ and solve Ψ with respect to Z.
5: Stop when no changes occur on Z.
6: Fix C and Z and solve Ψ with respect to $\mathbf{w}$.
7: Stop when no changes occur on $\mathbf{w}$.
8: t=t+1
9: **end while**

0.2.2 Algorithms for Text Data

Document clustering aims to segregate documents into meaningful clusters that reflect the content of each document. For example, in the news wire, manually assigning one or more categories for each document requires exhaustive human labour, especially with the huge amount of text uploaded online daily. Thus, efficient clustering is essential. Another problem associated with document clustering is the huge number of terms. In a matrix representation, each term will be a feature and each document is an instance. In typical cases, the number of features will be close to the number of words in the dictionary. This imposes a great challenge for clustering methods where the efficiency will be greatly degraded. However, a huge number of these words are either stop words, irrelevant to the topic, or redundant. Thus, removing these unnecessary words may help significantly reducing dimensionality.

Feature selection does not only reduce computational time but, also, improves clustering results and provides better data interpretability [49]. In document clustering, the set of

selected words that are related to a particular cluster will be more informative than the whole set of words in the documents with respect to that cluster. Different feature selection methods have been used in document clustering recently. For example, term frequency, pruning infrequent terms, pruning highly frequent terms, entropy-based weighting and so on. Some of these methods and others will be explained in the following subsections.

0.2.2.1 Term Frequency (TF)

Term Frequency is one of the earliest and most simple, yet, effective term methods. It is dated back to 1957 in [43]. Thus, it is, indeed, a conventional term selection method. In a text corpus, the documents that belong to the same topic are more likely will use similar words. Therefore, the frequent terms will be a good indicator for a certain topic. We can say that a very frequent term that is normally distributed across different topics is not informative, hence, such term would be unselected. We call this technique: *pruning high highly frequent terms*. Similarly, very rare terms should be prone as well and that is called: *pruning infrequent terms*. Stop words most likely will be pruned due to their high frequency. Furthermore, words such as "*abecedarian*" will be ignored since they will not be very frequent.

TF for term f_i with respect to the whole corpus is given by:

$$TF(f_i) = \sum_{j \in D_{f_i}} tf_{ij} \tag{0.8}$$

Where D_{f_i} is the documents that contain the term f_i and tf_{ij} is the frequency of f_i in document j.

0.2.2.2 Inverse Document Frequency (IDF)

TF is effective term selection method. However, it is not effective in terms of term weighting, where all selected terms will be assigned the same weights. Also, we cannot link TF value to any document. In other words, we cannot distinguish between frequent words that appeared in a small set of documents, which could have discriminative power for this set of documents, and frequent words that appear in all or most of the documents in the corpus. In order to scale the term's weight, we use, instead, the inverse document frequency (IDF). IDF measures whether the term is frequent or rare across all documents.

$$idf(f_i) = \log \frac{|D|}{|D_{f_i}|} \tag{0.9}$$

Where $|D|$ is the total number of documents (i.e. sample size) and $|D_{f_i}|$ is the number of documents that contain the term f_i. The value of IDF will be high for rare terms and low for highly frequent ones.

0.2.2.3 Term Frequency-Inverse Document Frequency (TF-IDF)

We can now combine the above mentioned measures (i.e. TF and IDF) to produce weight for each term f_i in each document d_j. This measure is called TF-IDF. It is given by:

$$tf\text{-}idf(f_i, d_j) = tf_{ij} * idf_(f_i) \tag{0.10}$$

tf-idf assigns greater values to these terms that occur frequently in a small set of documents, thus, more discriminative power. This value gets lower when the term occur in more documents. While the lowest value is given to terms that occur in all documents. In document clustering, terms that have higher *tf-idf* have higher ability for better clustering.

0.2.2.4 Chi Square statistic

Chi Square χ^2 statistic has been widely used in supervised feature selection [72]. It measures the statistical dependency between the feature and the class. χ^2 with r different values and C classes is defined as:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{C}\frac{(n_{ij}-\mu_{ij})^2}{\mu_{ij}},$$

where n_{ij} is the number of samples (i.e. documents) with i^{th} feature value in the j^{th} class and $\mu_{ij} = \frac{n_{i*}n_{*j}}{n}$ and n is the total number of documents. This equation can be interpreted using the probability as:

$$\chi^2(f,c) = \frac{n(p(f,c)p(\neg f,\neg c) - p(f,\neg c)p(\neg f,c))^2}{p(f)p(\neg f)p(\neg c)p(c)} \tag{0.11}$$

Where $p(f,c)$ is the probability of class c contains the term f and $p(\neg f,\neg c)$ is the probability of not being in class c and not containing term f and so on. Thus, χ^2 cannot be directly applied in an unsupervised learning such as clustering due to the absence of class label. Y. Li et al in [36] propose a variation of χ^2 called $r\chi^2$ that overcome some drawbacks of the original χ^2 and embedded in an Expectation-Maximization (EM) algorithm to be used for text clustering problem. [36] found out that χ^2 cannot determine whether the dependency between the feature and the class is negative or positive, which leads to ignoring relevant features and select irrelevant features sometimes. Therefore, they proposed a relevance measure (R) that can be used in the original χ^2 to overcome this limitation. This new measure R follows:

$$R(f,c) = \frac{p(f,c)p(\neg f,\neg c) - p(f,\neg c)p(\neg f,c)}{p(f)p(c)} \tag{0.12}$$

R in Eq (0.12) will be equal 1 if there is no such dependency between the class and the feature and greater than on if there is a positive dependency while less than one if the dependency is negative.

From Eqs (0.11) and (0.12), [36] proposed a new variation of χ^2 that is able to distinguish positive and negative relevance:

$$r\chi^2(f) = \sum_{j=1}^{C} p(R(f,c_j))\chi^2(f,c_j) \tag{0.13}$$

where $p(R(f,c_j))$ is given by: $p(R(f,c_j)) = \frac{R(f,c_j)}{\sum_{j=1}^{C} R(f,c_j)}$. The larger the value of $r\chi^2$ is, the more relevant the feature f will be.

As we mentioned earlier, we cannot apply a supervised feature selection in an unsupervised learning directly. Therefore, [36] embedded their proposed method given in Eq (0.13) in clustering algorithm using EM approach. They used k-means as clustering algorithm and $r\chi^2$ as feature selection method as shown in Algorithm (5).

One advantage of this framework that it does not simply remove the unselected features, instead, it keeps them while reducing their weight to α, so, they can be reselected in coming iterations. Also, this approach outperforms other clustering techniques even with the existence of feature selection methods using *F-measure* and the *purity*. However, [36] did not investigate the convergence of Algorithm (5) which is a big concern for such an algorithm especially when we know that the selected features may change dramatically from iteration to another. In addition, the complexity of this algorithm is not discussed. In fact, the feature

Algorithm 5 $r\chi^2$ as a feature selector for clustering

Input:
D: dataset
k: number of clusters
α: predetermine parameter in the range of [0,1)
m: number of selected features
Output:
C the clusters
Initialize:
$C \leftarrow$Apply k-means on D to obtain initial clusters

1. **E-step**:
   - Apply $r\chi^2$ from Eq (0.13) using clusters C obtain from step (1) as class label
   - Make the weight of the top m relevant features to be 1 and α for the rest of the features.
   - Calculate the new k centroid for the new space.
2. **M-step**: Compute the new clusters using k-means
3. Repeat E-step and M-step until convergence

selection step in Algorithm (5) is not a feature selection, instead, it is a feature weighting. In other words, the number of features in each iteration remains the same. Thus, the complexity of k-means will not be reduced, which is against the goals of involving feature selection in clustering.

Similar approach is proposed for Relief algorithm by Dash and Ong in [14]. They called their method Relief-C. It is observed that if clustering has done using the whole feature space, Relief will fail miserably in presence of large number of irrelevant features. Thus, Relief-C starts by clustering using randomly selected features, exactly 2 features, and using clusters as a class label passed to Relief. After that, the feature weight will be updated. These two steps repeated until given criteria is met. We believe Relief-C may not perform well with huge dimensionality, say more than 1000, features, since the chance of finding the real clusters from two randomly chosen feature is very slim especially if we know that the percentage of relevant features is very small. In addition, both Relief-C and $r\chi^2$ are capable of handling generic data. We include $r\chi^2$ in the text data section since it was originally applied on text domain and it requires dataset to contain discrete values.

0.2.2.5 Frequent Term-Based Text Clustering

Frequent Term-Based Text Clustering (FTC) proposed in [8] provides a natural way to reduce dimensionality in text clustering. It follows the notion of frequent item set that forms the basis of association rule mining. In FTC, the set of documents that contains the same frequent term set will be a candidate cluster. Therefore, clusters may overlap since the document may contain different item sets. This kind of clustering can be either flat (FTC) or hierarchical (HFTC) clustering since we will have different cardinalities of item sets.

Algorithm (6) explains the FTC algorithm. First, a dataset D, predetermined minimum support min_{sup} value and an algorithm that find frequent item set should be available. The algorithm starts by finding the frequent item set with minimum support min_{sup}. Then, it

runs until the number of documents contributing in the selected term set $|cov(STS)|$ is equivalent to the number of documents in D. In each iteration, the algorithm calculates the entropy overlap EO for each set in the remaining term set RTS, where EO is given by:

$$EO_i = \sum_{D_j \in C_i} -\frac{1}{F_j} \cdot \ln(\frac{1}{F_j})$$

where D_j is the j^{th} document, C_i is the i^th cluster and F_j is the number of all frequent term set supported by document J. The less overlap assumed to be the better. EO equals 0 if all the documents in C_i support only on frequent item set (i.e. $F_j = 1$). This value increases with the increase of F_j. This method of overlap evaluation was found to produce better clustering quality than the standard one [8]. The best candidate set $BestSet$ will be the set with minimum amount of overlap. $BestSet$ will be selected and added to the STS and excluded from RTS. In addition, the set of documents that support the $BestSet$ are removed from the dataset since they have been already clustered, which lead to dramatically reduce the number of documents. They are also removed from the documents' list of RTS which lead to reduce the number of remaining term set. This greedy approach gives the computational advantage for this algorithm.

Algorithm 6 Frequent Term-Based Clustering (FTC)

Input:
D: dataset
min_{sup}: minimum support
$\Psi(\cdot,\cdot)$: frequent item set finder
n: number of documents
Output:
STS and $cov(STS)$
Initialize:
Selected Term Set $STS = \{\}$

```
Remaining Term Set RTS = Ψ(D, min_sup)
while |cov(STS)| ≠ n do
    for each set in RTS do
        EO_i = Calculate overlap for the set
    end for
    BestSet = RTS_i which is the set with min(EO)
    STS = STS ∪ BestSet
    RTS = RTS − BestSet
    D = D − cov(BestSet)
    cov(RTS) = cov(RTS) − cov(BestSet)
end while
```

Due to the monotonicity property of frequent item set, which means all (k-1)-items that are subset of a frequent (k)-items are also frequent, we can perform a hierarchical frequent term-based clustering (HFTC). HFTC is based on Algorithm (6). Instead of performing FTC on the whole frequent term sets, it is performed on single level of frequent term set, say k-term sets. Then, the obtained clusters are further partitioned using the next level of term set, (k+1)-term sets.

Both FTC and HFTC were imperially proven to be superior to other well-known clustering methods such as bisecting k-means and 9-secting k-means in efficiency and clustering quality. They also was able to provide better description and interpretation of the generated clusters by the selected term set.

0.2.2.6 Frequent Term Sequence

Similar to FTC, a clustering based on frequent term sequence (FTS) was proposed in [34]. Unlike FTC, the sequence of the terms matters in FTS. This means that the order of terms in the document is important. The frequent terms sequence, denoted as $\mathbf{f}$, is a set that contains the frequent terms $< f_1, f_2, \ldots, f_k >$. The sequence here means that f_2 must be after f_1, where it is not necessary to be immediately after it. There could be other non-frequent terms between them. This is true for any f_k and f_{k-1} terms. This definition of frequent terms sequence is more adaptable to the variation of human languages [34].

Similar to FTC, FTS starts by finding frequent term sets using an association rule mining algorithm. This frequent term set guarantees to contain the frequent term sequence but not vice versa. Hence, we do not need to search the whole term space for the frequent term sequence. We can only search in the frequent term set space, which is a dramatic dimension reduction. After that, FTS builds a generalized suffix tree (GST), which is a well-known data structure for sequence pattern matching, using the documents after removing the non-frequent terms. From the suffix nodes in GST, we obtain the cluster candidates. These cluster candidates may contain subtopics that may be eligible to be merged together to create more general topics. Therefore, a merging step takes place.

The authors of [34] chose to merge cluster candidates into more general topic clusters using $k - mismatch$ instead of the similarity. An example of using $k - mismatch$ concept is when we have $FS_i = \{feature, selection, clustering\}$ and $FS_j = \{feature, selection, classification\}$, they have one mismatch. Therefore, we can merge these two clusters if the tolerance parameter $k \geq 1$. In [34], FTS adopted Landau−Vishkin (LV) to test three types of mismatches: insertion, deletion, substitution. Insertion means that all we need to insert is k, or less, terms into the FS_j in order to match FS_i. Deletion, in contrast, means we need to delete, instead. While substitution means we need to substitute terms from FS_j by terms from FS_i.

These merged clusters are prone to overlap. Accordingly, more merging will be performed after measuring the amount of overlap using *Jaccard Index*.

$$J(C_i, C_j) = \frac{|C_i \cup C_j|}{|C_i \cap C_j|}$$

The larger $J(C_i, C_j)$ is, the more overlap would be. The merge takes place here when the overlap exceeds a user defined threshold, δ. However, the final number of clusters could be predefined too. In this case, this merging step would be repeated until the number of clusters meet the user's demand.

In Algorithm (7), the most time consuming is constructing GST, yet, it is still linear with the number of terms. In [34], the terms reduction after finding the frequent term sets is huge, where it exceeds 90% in all cases.

Similar to FTS, [34] proposed another frequent term set algorithm (FTMS) that is based on the synonymous set (synsets) of terms instead of the term itself. This is more adaptable with human language where we may use different terms to refer to the same meaning. This proposed algorithm used WordNet dictionary to retrieve the synsets of each word and replace it with the original term. Each two synsets that intersect in at least one term will be merged into one synset. Thus, the number of synsets will be further reduced. Finally, FTS will be applied to these documents that contain the synsets.

These two proposed algorithms (i.e. FTS and FTMS) were empirically proved to be superior to other algorithms, such as bisect k-means and hierarchical frequent item-based clustering. Yet, FTMS is more capable of capturing more precise clustering topics than FTS. This is due to using the terms synsets instead of just the word itself.

There are several clustering techniques based on frequent item sets besides the ones

Algorithm 7 Frequent Term Sequence (FTS)

Input:
D: dataset
min_{sup}: minimum support
$\Psi(\cdot,\cdot)$: frequent item set finder
n : number of documents
K: number of clusters
δ: overlap threshold
Output:
$Clusters$
Initialize:
$Clusters = \{\}$

Frequent 2-Term Set $FS = \Psi(D, min_{sup})$
$\hat{D}$ = Reduce D by removing all every term $f \notin FS$.
$G = GST(\hat{D})$
for each node j in G **do**
  if node j has Frequent term set FS_j with documents id ID_j **then**
    $C_j = \{FS_j, ID_j\}$
    $Clusters = Clusters \cup C_j$
  end if
end for
Use Landau−Vishkin to find k−mismatch
$Clusters$ ←Merge $Clusters$ according to Landau−Vishkin results
while number of $Clusters > K$ **do**
  $Clusters$ ← Combine overlapped $Clusters$ based on *Jaccard Index* and δ
end while

mentioned in this chapter [20, 61, 77, 6, 48]. However, they all follow the same notion of reducing dimensionality by finding the frequent term sets. They differ in the parameters or the underlying utilized methods. For example, the number of terms in the frequent set may be set to a specific number or undefined. Hence, the maximum number will be found. They also differ in the way the final clustered is smoothed or merged.

0.2.3 Algorithms for Streaming Data

Streaming Data are continuous and rapid data records. Formally defined as a set of multi-dimensional records $X_1, \ldots, X_n, \ldots$ that come in time stamps $T_1, \ldots, T_n, \ldots$. Each record X_i is m-dimensional. Usually these samples are with a huge dimensionality and arrive very fast. Therefore, they require scalable and efficient algorithms. Also, the underlying cluster structure is changing, so, we need to capture this change and keep selected feature sets up-to-date. In this chapter, we introduce what we believe to be required characteristics of a good algorithm that handles streaming data. These characteristics are:

- **Adaptivity:** the algorithm should be able to adjust features' weights or even reselect the set of features, so it is able to handle the data drift, a.k.a. dataset shift.
- **Single scan:** the algorithm should be able to cluster the incoming stream from one scan, since another scan is usually impossible or at least costly.

The following algorithms represent the literature of feature selection for data stream clustering. In fact, this area still needs great attention from the researchers due to the lack of work in this area.

0.2.3.1 Text Stream Clustering Based on Adaptive Feature Selection (TSC-AFS)

It is natural for clusters and data categories to evolve overtime. For example, the breaking news in that dominates the worlds' media today may change dramatically tomorrow. Therefore, using the same set of features to process and cluster data stream may lead to unsatisfactory learning results over time. Gong et al in [22] proposed to use cluster quality threshold γ to test the quality of newly arrived sample's clustering with respect to old clusters. TSC-AFS starts by applying feature selection on training data D. Then, it clusters the samples of D with respect to the selected features only.

After that, it starts to receive data streams and assign them to the closest cluster. TSC-AFS evaluates a validity index ρ for each cluster using Davies-Boulding D-B index. If γ is less than ρ, then, TSC-AFS should reselect the set of features. Otherwise, the algorithm keeps the current feature set and clusters based on them and accepts new streams to come. Otherwise, TSC-AFS will re-evaluate ρ while considering the new data stream being added the closest cluster. If the validity index cannot satisfy the threshold requirements, a tolerance parameter κ is set to allow this sample to be clustered and initialize the algorithm again. These parameters, κ and γ, are simply determined by the cross-validation approach.

This algorithm utilizes a word variance-based selection method as feature selection since the domain in [22] is text data stream. However, any other appropriate algorithm may be used. In addition, k-means was used as a clustering method. Algorithm (8) shows the pseudocode for TSC-AFS.

TSC-AFS, shown in Algorithm (8), is arguable in terms of performance and applicability in the current form. However, it has a nice underlying property that is the adaptivity with the drifting features. We believe that this approach needs more attention to improve the adaptivity step.

Algorithm 8 Text Stream Clustering Based on Adaptive Feature Selection (TSC-AFS)

Input:
γ: cluster quality threshold
κ: tolerance parameter
t : time stamp
K: number of clusters
$\Xi(\cdot)$: feature selection method
Output: $Clusters$
Initialize: $t = \rho = 0$

while We have more stream **do**
  $D \leftarrow$ Load new training set
  Select features $F_t = \Xi(D)$
  $\hat{D} \leftarrow$ remove unselected features from D
  Apply k-means clustering on $\hat{D}$
  $\mu \leftarrow$ the centroid of each cluster
  while $\rho < \kappa$ **do**
    $t = t + 1$
    $S_t \leftarrow$ new stream
    Evaluate the validity index ρ
    $Flag = true$
    while $Flage$ **do**
      if $\rho > \gamma$ **then**
        Assign S_t to the nearest μ
        $\rho \leftarrow$ Re-evaluate the validity index
      else
        Assign S_t to the nearest μ
      end if
    end while
  end while
  $Clusters \leftarrow \{\mu, F\}$
end while

0.2.3.2 High-dimensional Projected Stream Clustering (HPStream)

Aggarwal et al in [4] proposed a data streams clustering technique that involves feature selection step. The proposed method in [4] is called HPStream. Projected clustering [5] is a subset of data points P with a subset of dimensions F such that the points in P are closely clustered with respect to F.

Since data quality in data streams may decay, the stream clustering method should assign a greater level of importance to recent data points. Motivated from this belief, fading data structure was proposed in [4] to keep the clustering contemporary. Assuming, $X_1, \ldots, X_i, \ldots$ are multi-dimensional data samples arriving in time stamps $T_1, \ldots, T_i, \ldots$. Each data sample X_i contains m dimensions $(x_i^1, \ldots, x_i^m)$. The fading $f(t)$ function for each data point will be according to the arrival time t. The fading function is assumed to be monotonically exponentially decreasing with respect to the time t [4].

Fading function $f(t) = 2^{-\lambda t}$, where $\lambda = 0.5$ is the decay rate. After defining the fading function, [4] defines the fading cluster structure at time t as $\Omega(P,t) = (\Psi^2(P,t), \Psi(P,t), w(t))$, where $\Psi^2(P,t)$ and $\Psi(P,t)$ for each j^{th} dimension are given by $\sum_{i=1}^{n} f(t-T_i) \cdot (x_i^j)^2$ and $\sum_{i=1}^{n} f(t-T_i) \cdot (x_i^j)$ respectively. And $w(t)$ is the sum of all the weights of the data points at time t, $\sum_{i=1}^{n} f(t-T_i)$.

Algorithm (9) systematically describes the HPStream algorithm. HPStream is initialized off-line using k-means to cluster a portion of the data D_{tr} using full dimension. Then, the least spread dimensions within each cluster will be chosen to form the initial fading cluster structure Ω. These two steps will be repeated, using the selected features until convergence. After initialization, incoming data stream X is temporally added to each cluster with the corresponding selected features of that cluster for determination of new set of selected features. Final assignment of X will be to the closest cluster. After that, the limiting radius of each cluster will be calculated. Any data point that lies outside the limiting radii will form a cluster by itself. If the number of generated clusters exceeds the predetermined number of clusters K, the oldest cluster(s), so, that the number of clusters equals K.

There are few more things regarding HPStream worth mentioning. First, the data set should be normalized for meaningful comparison between different dimensions. Also, Manhattan Segmental Distance is used to find the distance along the projected dimensions. This is a normalized version of Manhattan Distance that can compute the distance of different dimensionality. In terms of the empirical evaluation of HPStream, [4] conducted several experiments on real-world and synthetic datasets. HPStream was able to out perform CluStream [3], which as clustering method for evolving data streams. Also, it was shown to be very stable in terms of processing speed and scalable in terms of dimensionality.

0.2.4 Algorithms for Linked Data

Linked data has become ubiquitous in real-world applications such as tweets in Twitter [1] (tweets linked through hyperlinks), social networks in Facebook [2](people connected by friendships) and biological networks (protein interaction networks). Typically linked data has the following three characteristics: (1) high-dimensional such as there are tens of thousands of terms for tweets; (2) linked, providing an important source of information beyond attributes, i.e., link information; (3) unlabeled due to the large-scale size and the expensive cost for labeling. Such properties pose challenges to clustering task and feature selection is an effective way to prepare high-dimensional data for effective and efficient clustering [38]. In this subsection, we first introduce the challenges and opportunities of linked data for traditional feature selection, and then present an embedded unsupervised feature selection

[1] http://www.twitter.com/
[2] https://www.facebook.com/

Algorithm 9 High-dimensional Projected Stream Clustering (HPStream)

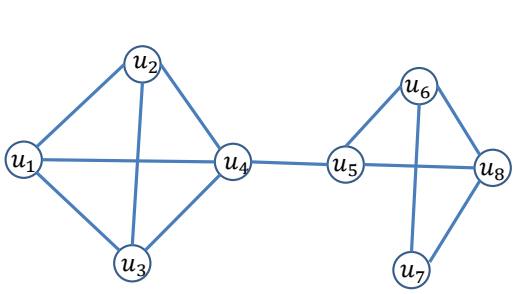
Input:
D_{tr}: training data set
K: number of clusters
l: number of selected features
Output: Ω
Initialize:
Perform clustering on D_{tr}
$F = \{F_1, \ldots, F_K\}$: sets of bit vectors
Ω : fading cluster structure

while $X \leftarrow$We have more incoming stream **do**
  for each cluster C_i **do**
    Add the new stream point X into C_i
    Update $\Omega(C_i, t)$
    for each dimension d **do**
      Compute the radii of $\Omega(C_i, t)$
    end for
    Pick the dimensions in $\Omega(C_i, t)$ with minimum radii
    Create selected features set F_i for C_i
  end for
  for each cluster C_i **do**
    for each selected feature F_i **do**
      $dis_i \leftarrow$ Average distance between X and the centroid of C_i
    end for
    $index = \text{argmin}_i\{\text{dis}_i\}$
    $s \leftarrow$ The radius of C_{index}
    if $dis_{index} > s$ **then**
      Add new cluster C_{K+1}
    else
      Add X to C_{index}
    end if
    Remove clusters with zero dimensions from Ω
    if Number of current clusters $> K$ **then**
      Delete the least recently added cluster from Ω
    end if
  end for
end while

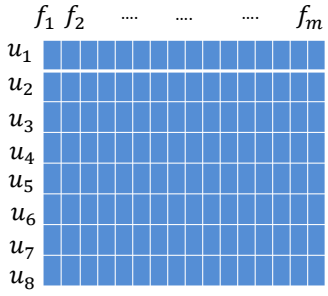
framework for linked data and finally follow with a discussion about future work of feature selection for linked data.

0.2.4.1 Challenges and Opportunities

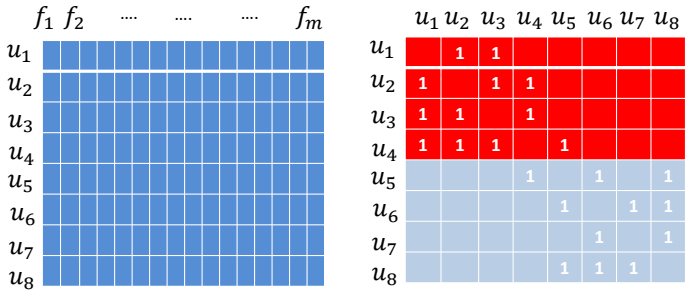
Linked instances are related to each other via different types of links (e.g., hyperlinks, friendships and interactions). Thus linked data is distinct from traditional attribute value data (or " flat" data). Figure 4 illustrates a typical example of linked data and its two representations. Figure 4(a) shows 8 linked instances (u_1 to u_8) while Figure 4(b) is a conventional representation of attribute-value data: rows are instances and columns are features. As mentioned above, except attributes, linked data provides an extra source in the form of links, represented as in Figure 4(c). These differences present both challenges and opportunities for traditional feature selection and machine learning [45, 57].



(a) Linked Data



(b) Attribute-Value Data Representation



(c) Linked Data Representation

FIGURE 4: A Simple Example of Linked Data

Linked data is patently not independent and identically distributed (i.i.d.), which is among the most enduring and deeply buried assumptions of traditional machine learning methods [29, 66]. Due to the absence of class labels that guide the search for relevant information, unsupervised feature selection is particularly difficult [11]. The linked property further exacerbates the difficulty of unsupervised feature selection for linked data. On the other hand, linked data provides more information than attribute value data and the availability of link information presents unprecedented opportunities to advance research. For example, linked data allows collective inference, inferring various interrelated values simultaneously [45] and enables relational clustering, finding a more accurate pattern [42].

Many linked data related learning tasks are proposed such as collective classification [45, 57], and relational clustering [41, 42], but the task of feature selection for linked

data is rarely touched due to its unique challenges for feature selection: (1) how to exploit relations among data instances; and (2) how to take advantage of these relations for feature selection. Until recently, feature selection for linked data attracts attention. In [63], a supervised feature selection algorithm, LinkedFS, is proposed for linked data in social media. Various relations(coPost, coFollowing, coFollowed and Following) are extracted following social correlation theories. LinkedFS significantly improves the performance of feature selection by incorporating these relations into feature selection. In [64], an unsupervised feature selection framework, LUFS, is developed for linked data [64]. More details about LUFS will presented in the following subsection.

0.2.4.2 LUFS: An Unsupervised Feature Selection Framework for Linked Data

LUFS is an unsupervised feature selection framework for linked data and in essence, LUFS investigates how to exploit and take advantage of link information of linked data for unsupervised feature selection. In general, the goal of feature selection is to select a subset of features to be consistent with some constraints [80]. For supervised learning, label information plays the role of constraint. Without label information, LUFS introduces the concept of pseudo-class label to guide unsupervised learning. Particularly, LUFS assumes that there is a mapping matrix $\mathbf{W} \in \mathbb{R}^{m\times c}$, assigning each data point with a pseudo-class label where m is the number of original features and c is the number of pseudo-class labels. The pseudo-class label indicator matrix is $\mathbf{Y} = \mathbf{W}^\top\mathbf{X} \in \mathbb{R}^{c\times n}$ where n is the number of data points. Each column of $\mathbf{Y}$ has only one nonzero entity, i.e., $\|\mathbf{Y}(:,i)\|_0 = 1$ where $\|\cdot\|_0$ is the vector zero norm, counting the number of nonzero elements in the vector. Then LUFS seeks pseudo-class label information by extracting constraints from both linked and attribute value data. The constraints from link information and attribute-value parts are obtained through social dimension regularization and spectral analysis, respectively.

Social Dimension Regularization: In [65], social dimension is introduced to improve the performance of relational learning. LUFS employs social dimensions to exploit the interdependency among linked data. It first adopts Modularity Maximization [50] to extract social dimension indicator matrix $\mathbf{H}$. Since social dimensions can be considered as a type of affiliations, according to Linear Discriminant Analysis, three matrices, i.e., within, between, and total social dimension scatter matrixes $\mathbf{S}_w$, $\mathbf{S}_b$, and $\mathbf{S}_t$, are defined as follows:

$$\begin{aligned}\mathbf{S}_w &= \mathbf{Y}\mathbf{Y}^\top - \mathbf{Y}\mathbf{F}\mathbf{F}^\top\mathbf{Y}^\top,\\ \mathbf{S}_b &= \mathbf{Y}\mathbf{F}\mathbf{F}^\top\mathbf{Y}^\top,\\ \mathbf{S}_t &= \mathbf{Y}\mathbf{Y}^\top,\end{aligned} \tag{0.14}$$

where $\mathbf{F} = \mathbf{H}(\mathbf{H}^\top\mathbf{H})^{-\frac{1}{2}}$ is the weighted social dimension indicator matrix.

Instances from different social dimensions are dissimilar while instances in the same social dimension are similar. Finally the constraint, social dimension regularization, from link information can be obtained via the following maximization problem,

$$\max_{\mathbf{W}} Tr\big((\mathbf{S}_t)^{-1}\mathbf{S}_b\big). \tag{0.15}$$

Spectral Analysis: To take advantage of information from attribute-value part, LUFS obtains the constraint from attribute-value part through spectral analysis [44] as,

$$\min \quad Tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top) \tag{0.16}$$

where $\mathbf{L}$ is a laplacian matrix.

Considering constraints from both link information and attribute-value part, LUFS, is

equivalent to solving the following optimization problem,

$$\begin{aligned} \min_{\mathbf{W},\mathbf{s}} \quad & Tr(\mathbf{YLY}^\top) - \alpha Tr\big((\mathbf{S}_t)^{-1}\mathbf{S}_b\big), \\ & s.t. \quad \|\mathbf{Y}(:,i)\|_0 = 1, \quad 1 \le i \le n. \end{aligned} \tag{0.17}$$

With the spectral relaxation for label indicator matrix [44], LUFS is eventually to solve the following optimization problem,

$$\begin{aligned} \min_{\mathbf{W}} \quad & f(\mathbf{W}) = Tr(\mathbf{W}^\top \mathbf{AW}) + \beta \|\mathbf{W}\|_{2,1}, \\ & s.t. \quad \mathbf{W}^\top \mathbf{BW} = \mathbf{I}_c, \end{aligned} \tag{0.18}$$

where $\mathbf{A} = \mathbf{XLX}^\top + \alpha\mathbf{X}(\mathbf{I}_n - \mathbf{FF}^\top)\mathbf{X}^\top$ is a symmetric and positive semidefinite matrix and $\mathbf{B} = \mathbf{XX}^\top + \lambda\mathbf{I}$ is a symmetric and positive matrix. Since the problem in Eq. (0.18) is convex, an optimal solution $\mathbf{W}$ can be guaranteed for LUFS [64]. The detailed algorithm for LUFS is shown in Algorithm 10.

Algorithm 10 LUFS

Input: $\{\mathbf{X}, \mathbf{R}, \alpha, \beta, \lambda, c, K, k\}$
Output: k most relevant features

1: Obtain the social dimension indicator matrix $\mathbf{H}$
2: Set $\mathbf{F} = \mathbf{H}(\mathbf{H}^\top\mathbf{H})^{-\frac{1}{2}}$
3: Construct $\mathbf{S}$ through RBF kernel
4: Set $\mathbf{L} = \mathbf{D} - \mathbf{S}$
5: Set $\mathbf{A} = \mathbf{XLX}^\top + \alpha\mathbf{X}(\mathbf{I}_n - \mathbf{FF}^\top)\mathbf{X}^\top$
6: Set $\mathbf{B} = \mathbf{XX}^\top + \lambda\mathbf{I}$
7: Set $t = 0$ and initialize $\mathbf{D}_0$ as an identity matrix
8: **while** Not convergent **do**
9: Set $\mathbf{C}_t = \mathbf{B}^{-1}(\mathbf{A} + \beta\mathbf{D}_t)$
10: Set $\mathbf{W}_t = [q_1, \ldots, q_c]$ where $q_1, \ldots, q_c$ are the eigenvectors of $\mathbf{C}_t$ corresponding to the first c smallest eigenvalues
11: Update the diagonal matrix $\mathbf{D}_{t+1}$, where the i-th diagonal element is $\frac{1}{2\|\mathbf{W}_t(i,:)\|_2}$;
12: Set $t = t + 1$
13: **end while**
14: Sort each feature according to $\|\mathbf{W}(i,:)\|_2$ in **descending** order and select the top-k ranked ones;

In Algorithm 10, social dimension extraction and weighted social dimension indicator construction are from line 1 to line 2. The iterative algorithm to optimize Eq. (0.18) is presented from line 8 to line 13.

0.2.4.3 Conclusion and Future Work for Linked Data

In this subsection, we first analyze the differences between linked data and attribute-view data, then present the challenges and opportunities posed by linked data for feature selection and finally introduce a recent proposed unsupervised feature selection framework, LUFS, for linked data in details. To the best of our knowledge, LUFS is the first to study feature selection for linked data in an unsupervised scenario and many works are needed to further exploit linked data for feature selection.

Since LUFS can be categorized as an embedded method, analogous to conventional data, how to develop filter, wrapper and hybrid models for linked data is an interesting

problem for further research. Furthermore, LUFS employs social dimension to capture link information, a further investigation into sophisticated methods of exploiting link will lead to novel approaches and help us develop a deeper understanding of how to improve the performance of feature selection. Finally, the availability of various link formation can lead to links with heterogeneous strengths. Therefore how to incorporate automatic link selection into feature selection is another promising direction for linked data.

0.3 Discussions and Challenges

Here are several challenges and concerns that we need to mention and discuss briefly in this chapter about feature selection for clustering.

0.3.1 The Chicken or the Egg Dilemma

In feature selection for clustering, there is a dilemma in choosing which one to start with or which one serves the demands of the other. Do we utilize feature selection to improve clustering quality? Or, do we use clustering as an indicator of relevant features? This, in fact, leads us to the chicken or the egg dilemma. Which one comes first, clustering or feature selection? If the answer to the latter is yes, that means feature selection is a goal in itself. However, this is not the case in feature selection literature. We usually use feature selection, as we mentioned earlier, to improve learning quality, reduce computational time and reduce require storage. Thus, the answer to the first question is, indeed, yes. We use feature selection to improve clustering quality. Accordingly, we should select the features that preserve cluster structure. However, some current methods initialize clusters using all features. Then, they apply supervised feature selection method using the initial clusters as class labels. We believe that such techniques will not achieve the goal of feature selection since the initial clusters may not be the real ones or even close. If these clusters are claimed to be real ones, why bother doing feature selection at all? Some other methods apply the same process but in an iterative manner using an EM-like approach. Although this might lead to better results, we still believe that there are several drawbacks associated with this approach. For example, if the dimensionality in the original space is huge, such an approach may take very long time to converge. Therefore, we prefer utilizing feature selection that does not depend on any clustering input to define the relevancy of the the feature. In other words, we may utilize the clustering method to guide the searching process but we do not use the clustering as a class label. This means, we apply the feature selection method on the whole dataset and then use the selected features to construct the clusters. If we are not satisfied with the clustering quality, we may use this as an indicator that this set of features is not satisfactory. This is exactly the *wrapper approach*.

0.3.2 Model Selection: K and l

Selecting the number of clusters K or the number of selected features l is an open problem. In real world problems, we have limited knowledge about each domain. Consequently, determining optimal K or l is almost impossible. Although this problem seems not to be a big issue with some approaches (such as frequent term set based feature selection methods) we still need to determine other sensitive parameters like the minimum support, which leads us to another problem. Choosing different l for the same problem usually results

in different equally good subsets of features, and hence, different clustering results. Similarly, choosing different K leads to merging totally different clusters into one or splitting one cluster into smaller ones. In other cases, this may lead to changing the set of selected features which results in losing potential clusters and/or constructing less important ones. Therefore, picking nearly optimal parameters is desirable for better clustering quality. Some work has been done to quantify such parameters. For instance, [12] used the notion of false nearest neighbor to find the number of selected features for clustering. On the other hand, [67] used gap statistics to estimate the number of clusters in a dataset. However, the effort in finding better parameters for clustering is still limited. We believe it is necessary to pay more attention to these two parameters for better clustering results.

0.3.3 Scalability

With the tremendous growth of dataset sizes, the scalability of current algorithms may be in jeopardy, especially with these domains that require online clustering. For example, streaming data or data that cannot be loaded into the memory require single data scan where the second pass is either unavailable or very expensive. Using feature selection methods for clustering may reduce the issue of scalability for clustering. However, some of the current methods that involve feature selection in the clustering process require to keep full dimensionality in the memory to observe any evolving in the cluster structure. If a change in data structure is observed, the clustering process should be restarted. Furthermore, other methods require an iterative process where each sample is visited more than once until convergence.

On the other hand, the scalability of feature selection algorithms is a big problem. Usually, they require a sufficient number of samples to obtain, statically, good enough results. It is very hard to observe feature relevance score without considering the density around each sample. Some methods try to overcome this issue by memorizing only samples that are important or a summary, say the mean of each cluster. In conclusion, we believe that the scalability of clustering and feature selection methods should be given more attention to keep pace with the growth and fast streaming of the data.

0.3.4 Stability

In supervised learning such as classification, stability of feature selection algorithms has gained increasing attention last few years. Stability of feature selection algorithms or selection stability is the sensitivity of the selection toward data perturbation [7]. An example of data perturbation would be new data sample(s). Therefore, with a small amount of perturbation introduced to the data, we do not expect dramatical change in the selected features. Otherwise, the algorithm is considered to be unstable. To the best to our knowledge, selection stability in unsupervised learning has not been studied yet.

Studying stability in supervised learning is much easier than the unsupervised. Due to the absence of class label in latter, we cannot maintain enough knowledge about the underlying clusters within the data. For this reason, we cannot be confident enough if the new sample(s) belong to any existing clusters or they form one or more new clusters. However, in supervised learning, we have limited number of classes. Thus, a sample that does not belong to any existing class would be considered an outlier and we do not need to modify our selected set to obey outliers. So, to address the issue of feature selection stability in clustering, we discuss it with respect to different situation:

Predefined Clusters:

In case we have enough domain knowledge to predefine the possible clusters, the stability issue will be identical to the stability in supervised learning. Therefore, we expect the feature

selection algorithm to be stable with the existence of the small amount of perturbation. However, if the topics do not change while the data itself change, we might consider unstable selection.

Undefined Clusters:

In contrast the first case, if we do not have enough knowledge about the clusters of the data, which is true in real-world situations, we generally can say that this case depends on the domain and the practitioner's desired. In streaming data, for example, we usually expect some kind of dataset shift or drift where the distribution and the clustering topics of the data may evolve over time. Therefore, we should have an adaptive feature selection process that adjusts the selected features in a away that can capture the evolving structure of the data. However, we might desire a stable selection for a period of time and unstable for another period. For example, in a news data clustering, we wish to have stable selection while we do not have a breaking news that takes the distribution to different topic.

On the other hand, subspace and projected clustering may not promote stable selection since they search for all possible sets of features that define clusters. In some cases, we might want to maintain stable selection on some clusters, while unstable on others. For instance, if we are performing projected clustering on news data, we may want to have stable selection on clusters that do not evolve rapidly while having unstable selection on rapidly evolving clusters.

Bibliography

[1] Feature selection for dna methylation based cancer classification. *Bioinformatics*, 17 Suppl 1:S157–S164, 2001.

[2] A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, Oct 2007.

[3] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[4] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment, 2004.

[5] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J.S. Park. Fast algorithms for projected clustering. *ACM SIGMOD Record*, 28(2):61–72, 1999.

[6] T.M. Akhriza, Y. Ma, and J. Li. Text clustering using frequent contextual termset. In *Information Management, Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on*, volume 1, pages 339–342. IEEE, 2011.

[7] Salem Alelyani, Lei Wang, and Huan Liu. The effect of the characteristics of the dataset on the selection stability. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, 2011.

[8] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM, 2002.

[9] C. Boutsidis, M.W. Mahoney, and P. Drineas. Unsupervised feature selection for the k-means clustering problem. *Advances in Neural Information Processing Systems*, 22:153–161, 2009.

[10] P.S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. pages 82–90. Morgan Kaufmann, 1998.

[11] D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2010.

[12] A.C. Carvalho, R.F. Mello, S. Alelyani, H. Liu, et al. Quantifying features using false nearest neighbors: An unsupervised approach. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 994–997. IEEE, 2011.

[13] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 74–81, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] M. Dash and Y.S. Ong. Relief-c: Efficient feature selection for clustering over noisy data. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 869–872. IEEE, 2011.

[15] Manoranjan Dash, Kiseok Choi, Peter Scheuermann, and Huan Liu. Feature selection for clustering - a filter solution. In *In Proceedings of the Second International Conference on Data Mining*, pages 115–122, 2002.

[16] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2 edition, 2001.

[17] Jennifer G. Dy and Carla E. Brodley. Feature subset selection and order identification for unsupervised learning. In *In Proc. 17th International Conf. on Machine Learning*, pages 247–254. Morgan Kaufmann, 2000.

[18] Jennifer G. Dy and Carla E. Brodley. Feature selection for unsupervised learning. *J. Mach. Learn. Res.*, 5:845–889, 2004.

[19] J.G. Dy. Unsupervised feature selection. *Computational Methods of Feature Selection*, pages 19–39, 2008.

[20] B.C.M. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, volume 30, pages 59–70, 2003.

[21] Nicola L. C. Talbot Gavin C. Cawley and Mark Girolami. Sparse multinomial logistic regression via bayesian l1 regularisation. In *NIPS*, 2006.

[22] L. Gong, J. Zeng, and S. Zhang. Text stream clustering algorithm based on adaptive feature selection. *Expert Systems with Applications*, 38(3):1393–1399, 2011.

[23] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[24] Mark A. Hall. Correlation-based feature selection for machine learning. Technical report, 1999.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[26] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *Advances in Neural Information Processing Systems*, 18:507, 2006.

[27] J.Z. Huang, M.K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):657–668, 2005.

[28] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:153–158, 1997.

[29] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, pages 259–266, 2002.

[30] L. Jing, M.K. Ng, and J.Z. Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *Knowledge and Data Engineering, IEEE Transactions on*, 19(8):1026–1041, 2007.

[31] Thorsten Joachims, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, and Lehrstuhl Viii. Text categorization with support vector machines: Learning with many relevant features, 1997.

[32] Y.S. Kim, W.N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6(6):531–556, 2002.

[33] Ron Kohavi and George H. John. Wrappers for feature subset selection, 1996.

[34] Y. Li, S.M. Chung, and J.D. Holt. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64(1):381–404, 2008.

[35] Y. Li, M. Dong, and J. Hua. Localized feature selection for clustering. *Pattern Recognition Letters*, 29(1):10–18, 2008.

[36] Y. Li, C. Luo, and S.M. Chung. Text clustering with feature selection by using statistical data. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):641–652, 2008.

[37] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic Publishers, 1998.

[38] H. Liu and H. Motoda, editors. *Computational Methods of Feature Selection*. Chapman and Hall/CRC Press, 2007.

[39] Huan Liu and Rudy Setiono. A probabilistic approach to feature selection - a filter solution. pages 319–327. Morgan Kaufmann.

[40] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491 – 502, April 2005.

[41] B. Long, Z.M. Zhang, X. Wu, and P.S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592. ACM, 2006.

[42] B. Long, Z.M. Zhang, and P.S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–479. ACM, 2007.

[43] H.P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[44] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[45] S.A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research*, 8:935–983, 2007.

[46] Pabitra Mitra, Student Member, C. A. Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:301–312, 2002.

[47] D.S. Modha and W.S. Spangler. Feature weighting in k-means clustering. *Machine learning*, 52(3):217–237, 2003.

[48] K. Morik, A. Kaspari, M. Wurst, and M. Skirzynski. Multi-objective frequent termset clustering. *Knowledge and Information Systems*, pages 1–24, 2012.

[49] MSK Mugunthadevi, M. Punitha, and M. Punithavalli. Survey on feature selection in document clustering. *International Journal*, 3, 2011.

[50] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):26113, 2004.

[51] Andrew Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 404–412. Morgan Kaufmann, 1998.

[52] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. In *Machine Learning*, pages 103–134, 1999.

[53] I.S. Oh, J.S. Lee, and B.R. Moon. Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1424–1437, 2004.

[54] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[55] V. Roth and T. Lange. Feature selection in clustering problems. *Advances in neural information processing systems*, 16, 2003.

[56] Yong Rui and Thomas S. Huang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10:39–62, 1999.

[57] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[58] Wojciech Siedlecki and Jack Sklansky. On automatic feature selection. pages 63–87, 1993.

[59] M. R. Sikonja and I. Kononenko. Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, 53:23–69, 2003.

[60] L. Song, A. Smola, A. Gretton, K. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *International Conference on Machine Learning*, 2007.

[61] C. Su, Q. Chen, X. Wang, and X. Meng. Text clustering approach based on maximal frequent term sets. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 1551–1556. IEEE, 2009.

[62] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 389–397. MORGAN KAUFMANN PUBLISHERS, INC., 1999.

[63] Jiliang Tang and Huan Liu. Feature selection with linked data in social media. In *SDM*, 2012.

[64] Jiliang Tang and Huan Liu. Unsupervised feature selection for linked social media data. In *KDD*, 2012.

[65] L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826. ACM, 2009.

[66] B. Taskar, P. Abbeel, M.F. Wong, and D. Koller. Label and link prediction in relational data. In *Proceedings of the IJCAI Workshop on Learning Statistical Models from Relational Data*. Citeseer, 2003.

[67] 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.

[68] C.Y. Tsai and C.C. Chiu. Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm. *Computational statistics & data analysis*, 52(10):4658–4672, 2008.

[69] J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003.

[70] Dietrich Wettschereck, David W. Aha, and Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

[71] D.M. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010.

[72] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.

[73] Zenglin Xu, Rong Jin, Jieping Ye, Michael R. Lyu, and Irwin King. Discriminative semi-supervised feature selection via manifold regularization. In *IJCAI' 09: Proceedings of the 21th International Joint Conference on Artificial Intelligence*, 2009.

[74] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. pages 412–420. Morgan Kaufmann Publishers, 1997.

[75] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML-03),*, pages 856–863, Washington, D.C., August 21-24, 2003 2003. Morgan Kaufmann.

[76] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research (JMLR)*, 5(Oct):1205–1224, 2004.

[77] W. Zhang, T. Yoshida, X. Tang, and Q. Wang. Text clustering using frequent itemsets. *Knowledge-Based Systems*, 23(5):379–388, 2010.

[78] Z. Zhao and H. Liu. *Spectral Feature Selection for Data Mining*. Chapman & Hall/Crc Data Mining and Knowledge Discovery. Taylor & Francis, 2011.

[79] Zheng Zhao and Huan Liu. Semi-supervised feature selection via spectral analysis. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2007.

[80] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1151–1157, New York, NY, USA, 2007. ACM.