Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Optimizing Clustering in High Dimensional Data: Preliminary vs. Integrated Feature Selection

Kiki Zinken (563979kz)

Supervisor:	C. Cavicchia
Second assessor:	H. Deng
Date final version:	29th June 2024

Abstract

This research examines the combined efficacy of clustering and feature selection, aiming to address the limitations created by high-dimensional data and thereby improve the clustering performance. Two approaches are compared: a two-step approach that performs feature selection before clustering and an integrated approach that embeds feature selection into the clustering process. In the two-step approach, feature selection is conducted using either the Laplacian score or the first order incremental search of the Max-Relevance and Min-Redundancy criteria (mRMR), followed by k-means or convex clustering, implemented using the Alternating Minimization Algorithm (AMA). In the integrated approach, sparse k-means and sparse convex clustering are applied, utilizing the Sparse AMA (S-AMA). Simulated data is used to systematically evaluate these methods across various data settings and noise levels. The results demonstrate that preliminary feature selection significantly improves clustering performance for k-means clustering. In contrast, the integrated approach encounters difficulties when the noise level increases. Conversely, for convex clustering, both AMA and S-AMA perform poorly, limiting conclusive findings on the impact of feature selection. These findings highlight the effectiveness of embedding feature selection into the k-means clustering process when considering high-dimensional data.

1 Introduction

The amount of high-dimensional data publicly available on the internet has greatly increased in recent years, making the extraction of meaningful patterns from this data a pressing issue. Identifying these patterns can be accomplished through clustering, a fundamental unsupervised learning method, categorizing similar data points into coherent clusters (Kaufman & Rousseeuw, 2009). Specifically, convex clustering (Hocking, Joulin, Bach & Vert, 2011; Lindsten, Ohlsson & Ljung, 2011; Pelckmans, De Brabanter, Suykens & De Moor, 2005) has shown its importance. It formulates the clustering as a convex optimization problem, ensuring stability and convergence towards a global optimum. Another well-known clustering method is k-means clustering (MacQueen, 1967), which iteratively partitions the dataset into k distinct subsets by minimizing the variance within each cluster.

The effectiveness of convex clustering and k-means clustering still falls short when handling high-dimensional data with numerous uninformative features. These features have the potential to conceal the true data structure, leading to suboptimal clustering outcomes. Feature selection has demonstrated efficacy in enhancing the performance of machine learning models, by identifying and retaining solely the most informative features (Kumar & Minz, 2014; Li et al., 2017; Cai, Luo, Wang & Yang, 2018), indicating that the addition of feature selection could benefit convex clustering.

Recently, B. Wang, Zhang, Sun and Fang (2018) introduced sparse convex clustering, which integrates feature selection into convex clustering, using the Sparse Alternating Minimization Algorithm (S-AMA), resulting in significant improvement for high-dimensional data. This is not the only work on simultaneously performing clustering and feature selection. Witten and Tibshirani (2010) introduced sparse k-means clustering, extending traditional k-means clustering by incorporating a feature selection mechanism that assigns weights to features, promoting sparsity to identify and retain only the most relevant features for clustering, again showing

superior performance for high-dimensional data.

This research examines the combined efficacy of enhancing clustering with feature selection, aiming to address the limitations posed by high-dimensional data and enhance the overall efficacy of the clustering process. This is investigated by comparing two approaches. First, an integrated approach that simultaneously performs feature selection and clustering, which is implemented using S-AMA and sparse k-means. Second, a two-step approach that performs feature selection before convex clustering. This approach is considered, since literature suggests that in case of high-dimensional data it can outperform embedded feature selection (Liu & Yu, 2005; Saeys, Inza & Larranaga, 2007; Bolón-Canedo, Sánchez-Maroño & Alonso-Betanzos, 2013). The objective is to determine the optimal framework by evaluating cluster quality, which has not yet been determined in previous work.

For the two-step approach, the original convex clustering method (Pelckmans et al., 2005; Hocking et al., 2011; Lindsten et al., 2011), implemented using AMA, and the traditional k-means clustering (MacQueen, 1967) are used for creating the groupings, while unsupervised feature selection is used to handle the uninformative features of the high-dimensional dataset, to make a fair comparison with the sparse clustering models. The Laplacian score is used as unsupervised feature selection method, which concerns the local structure of the data. He, Cai and Niyogi (2005) introduce a Laplacian score method for feature selection. The method evaluates the relevance of features based on their locality preserving power by assigning the Laplacian score to each feature. The importance of the features is then based on this score, where a lower score indicates a more informative feature.

Additionally, supervised feature selection is used in the two-step approach, to determine the effect on clustering performance when the labels are included in the feature selection process. A first-order incremental search of the minimum-redundancy-maximum-relevance (mRMR) criterion, as introduced by Peng, Long and Ding (2005), is considered for supervised feature selection. This method uses mutual information to determine a subset of features that maximizes the relevance of the features for determining the cluster, while simultaneously minimizing the redundancy of features in the subset, ensuring an optimal feature subset.

Simulated data is used in order to control the dimensionality and noise levels precisely. This controlled environment allows for a systematic comparison of the clustering methods across different scenarios. By manipulating these parameters, evaluating and identifying which methods perform best under specific conditions becomes possible, providing insights into their strengths and weaknesses.

The rest of the paper is structured in the following manner. First an overview of related work is provided in Section 2. Then a detailed description of the utilized datasets is provided in Section 3, followed by an explanation of the used models in Section 4. Finally, Sections 5 and 6 provide the results and conclusion, respectively.

2 Related Work

This study focuses on the clustering of high-dimensional data. Various clustering methods have been developed over the years, starting with some traditional clustering models, which utilize a greedy approach. Hierarchical clustering is one of the earliest traditional clustering

algorithms (Ward Jr, 1963). This organizes data into a hierarchy of clusters in a bottom-up or top-down method (Murtagh & Contreras, 2012) or a hybrid method (Vichi, Cavicchia & Groenen, 2022). Not long after k-means clustering was developed, which is considered to be one of the simplest and most popular clustering algorithm. The clustering is based on centroids, assigning the data to K clusters (MacQueen, 1967). Later the Gaussian Mixture Model (GMM) gained prominence in clustering, by modelling the distribution of the data as a mixture of several Gaussian distributions (Hastie, Tibshirani, Friedman & Friedman, 2009). GMM enables soft clustering, meaning that a data instance can belong to multiple clusters with different probabilities. These traditional clustering algorithms are susceptible to instabilities due to their non-convex optimization formulations.

Recently, convex clustering has emerged to address these instability issues (Hocking et al., 2011; Lindsten et al., 2011; Pelckmans et al., 2005) by formulating clustering as a convex optimization problem. The objective function is defined in such a way that convergence to a global optimum is ensured, leading to more stable and predictable solutions because of its convex optimization. Additionally, convex optimization techniques can be used for convex clustering, facilitating efficient and adaptable applications. These techniques are specifically useful for high-dimensional and noisy datasets as they create robust clusters. Alternating Direction Method of Multiplier (ADMM) is one of these convex optimization techniques. ADMM creates smaller and easier solvable sub-problems of the original problem (Glowinski & Marroco, 1975; Gabay & Mercier, 1976; Boyd, Parikh, Chu, Peleato & Eckstein, 2011). Another useful convex optimization technique is the Alternating Minimization Algorithm (AMA), alternating which part of the objective function is minimized (Tseng, 1991).

However, the performance of all these clustering methods are suboptimal for high-dimensional data with various uninformative features, since these features can hide the true data structure. Dealing with these uninformative features is mostly handled by feature selection, which identifies and retains solely the most informative features (Kumar & Minz, 2014; Li et al., 2017; Cai et al., 2018). Feature selection can be performed before clustering or incorporated into the clustering procedure, which is referred to as preliminary feature selection and integrated feature selection, respectively.

2.1 Preliminary Feature Selection

Several feature selection methods are defined in literature for performing feature selection before clustering. These can be split into supervised, unsupervised and semi-supervised models (Cai et al., 2018). Supervised models make use of labeled data to determine an optimal feature subset, while unsupervised solely relies on the local structure of the data without using labels. Semi-supervised models combine labeled and unlabeled data for determining the feature subset. Unsupervised feature selection is utilized in this study for the comparison of the preliminary and integrated feature selection, as in the integrated approach the labels are not used. Additionally, supervised feature selection for the two-step approach is used and compared with the unsupervised two-step approach.

2.1.1 Unsupervised Feature Selection

Unsupervised feature selection methods aim to improve clustering accuracy by finding a feature subset that remains the local structure of the data. Several methods have been created over time, varying in their way to select features. Dash and Liu (1999) introduced an unsupervised method to chose the feature subset using entropy to determine feature relevance. The entropy measures the randomness or uncertainty within the data, helping to find features that contain significant variability. Then the trace criterion is used to select the most important feature subset.

Another popular and easily applicable method is introduced by Mitra, Murthy and Pal (2002). In their method the maximum information compression index is used to determine the similarity between features, assessing the amount of information retention achieved by combining features. The aim is to retain features that are informative and not redundant.

The Laplacian score is another popular criteria for preliminary unsupervised feature selection. He et al. (2005) proposed a method where the Laplacian score it used to determine feature importance. The method is based on the principle that instances within the same cluster should be in close proximity to one another. This research focuses on the Laplacian score for unsupervised feature selection as it effectively preserves the local data structure by measuring the effect of each feature on the similarity relationship among neighboring instances. As this method ensures a feature subset that aids in forming well-defined, cohesive clusters, it is particularly suitable for clustering tasks.

2.1.2 Supervised Feature Selection

Supervised feature selection is based on the principle of evaluating the relevance or correlation between each feature and the target classification, aiming to find the optimal feature subset that maximizes the classification accuracy. A well known subset selection method is Correlation-based Feature Selection (CFS), introduced by (Hall, 2000), using heuristic approaches to select features with high correlation towards the target class and low inter-correlation between the selected features.

ReliefF (Kononenko, 1994) uses the Euclidean distance as correlation index and weights the features based on their ability to differentiate instances of different classes. This is an extension of Relief (Kira & Rendell, 1992), supporting multi-class problems.

Instead of considering the correlation of the features, information measures for relevance and redundancy can be used. A classical feature selection criterion for relevance and redundancy analysis using mutual information is Max-Relevance and Min-Redundancy (MRMR). This criterion can then be implemented using a first-order incremental search, creating the mRMR approach, introduced by Peng et al. (2005). This approach is specifically chosen for its ability to balance relevance and redundancy effectively, which is crucial in high-dimensional data where many features might be correlated or irrelevant.

Sometimes conditional mutual information is used, which considers the impact of selected features on the classification performance of candidate features, addressing the limitation of mutual information-based methods like mRMR, which only minimize feature-feature mutual

information. However, mRMR often results in superior performance due to its theoretical first-order optimality, resulting in the preference for mRMR.

2.2 Integrated Feature Selection

In other related work it is considered to simultaneously use clustering and feature selection. Some of these methods follow a model-free strategy, such as Witten and Tibshirani (2010), Sun, Wang and Fang (2012), and Y. Wang, Fang and Wang (2016). Conversely, others adopt a model-based approach, as demonstrated by Raftery and Dean (2006), Pan and Shen (2007), S. Wang and Zhu (2008), Xie, Pan and Shen (2008), and Guo, Levina, Michailidis and Zhu (2010). Sun et al. (2012) introduced sparse k-means clustering, incorporating feature selection into k-means clustering. It allows identification and focus on the most relevant features for clustering, thus improving performance and interpretability for high dimensional data.

As all of these methods use a non-convex optimization formulation, they encounter the same instabilities as the traditional clustering models mentioned above. However, these methods do provide very strong numerical performance. In order to benefit from the numerical performance of these methods, while also addressing their instability issues, B. Wang et al. (2018) created sparse convex clustering. This was then implemented using the Sparse ADMM (S-ADMM) and Sparse AMA (S-AMA) algorithms, where it was shown that S-AMA surpasses S-ADMM in computational time and performance. The S-AMA model also outperforms convex clustering methods as well as k-means clustering for high-dimensional data, resulting in the choice of S-AMA when using sparse convex clustering.

Due to the inclusion of k-means clustering in this study, the sparse k-means clustering (Sun et al., 2012) is considered to enable a comparison of preliminary and integrated feature selection for k-means.

3 Data Simulation

This research is conducted with simulated data, as in this manner the data can be controlled. The goal is to find out whether a preliminary or an integrated approach of combining clustering with feature selection performs better for high dimensional data with lots of noise. In the simulated data, the level of noise can be controlled, making it ideal for this study.

In order to simulate data several spherical settings are used, resulting in four simulated datasets. The simulated datasets consists of $n = 60$ observations with the number of features p being 150 or 500. The number of clusters K is set either to 2 or 4. The datasets are constructed in such a way that the first 20 features are informative, resulting in the other $p - 20$ features being uninformative.

The data instances x_{ij}, where $i \in \{1, \ldots, n\} \wedge j \in \{1, \ldots, p\}$, are generated in the following manner. For each i, a cluster label Z_i is sampled from the uniform distribution ranging from 1 to K. The informative features are generated from $MNV_p(\boldsymbol{\mu}_K(Z_i), \mathbf{I}_{20})$, where MNV_p denotes a p-dimensional multivariate normal distribution and $\boldsymbol{\mu}_K(Z_i)$ is depended on the number of clusters, as seen in (1) and (2). The μ in (1) and (2) regulates the distance between cluster centers. In this context, a large μ indicates well-separated clusters, whereas a small μ indicates

overlapped clusters.

$$\boldsymbol{\mu}_2(Z_i) = \mu \mathbf{1}_{20} I(Z_i = 1) - \mu \mathbf{1}_{20} I(Z_i = 2) \tag{1}$$

$$\begin{aligned} \boldsymbol{\mu}_4(Z_i) =& (\mu \mathbf{1}_{10}^T, -\mu \mathbf{1}_{10}^T)^T) I(Z_i = 1) + (-\mu \mathbf{1}_{10}^T, -\mu \mathbf{1}_{10}^T)^T) I(Z_i = 2) \\ &+ (-\mu \mathbf{1}_{10}^T, \mu \mathbf{1}_{10}^T)^T) I(Z_i = 3) + (\mu \mathbf{1}_{10}^T, \mu \mathbf{1}_{10}^T)^T) I(Z_i = 4) \end{aligned} \tag{2}$$

Lastly, The non-informative features, also known as noise features, are generated using the standard normal distribution $\mathcal{N}(0, \sigma)$. In this research various values for σ will be considered to determine the performance of the methods with different noise levels, starting with $\sigma = 1$ for a low noise level, followed by $\sigma = 1.5$ for a medium noise level and $\sigma = 2$ for a high noise level. When using the low noise level of $\sigma = 1$, the data simulation follows B. Wang et al. (2018).

In summary, four simulation settings are considered, where for each setting the σ values 1, 1.5 and 2 are used. Setting 1: $K = 2$, $n = 60$, $p = 150$ and $\mu = 0.6$, setting 2: $K = 2$, $n = 60$, $p = 500$ and $\mu = 0.7$, setting 3: $K = 4$, $n = 60$, $p = 150$ and $\mu = 0.9$ and setting 4: $K = 4$, $n = 60$, $p = 500$ and $\mu = 1.2$.

4 Methodology

The focus of this study lies on comparing two ways of incorporating feature selection into clustering to optimize the clustering performance for high-dimensional data. The first option is to use feature selection before clustering, referred to as the two-step approach. The feature selection methods are discussed in Section 4.1 and the clustering methods in Section 4.2 Then in Section 4.3 the second approach is discussed, namely the integrated approach, where feature selection is embedded into the clustering process, often referred to as sparse clustering.

4.1 Preliminary Feature Selection

The two step approach consists of the following steps. First, feature selection is employed to decrease the feature dimensionality. Second, clustering is employed to predict the target classification. This section focuses on the feature selection of the two step approach. The unsupervised feature selection is handled by the Laplacian Score, while the supervised feature selection is handled by mRMR, explained in Sections 4.1.2 and 4.1.1, respectively.

4.1.1 Laplacian Score

The Laplacian score (He et al., 2005) is used to handle unsupervised feature selection in the two step approach. The Laplacian score represent the locality preserving power of a feature, where a lower value results in the feature being in the feature subset. This is fundamentally based on Laplacian Eigenmaps (Belkin & Niyogi, 2001) and locality preserving projection (He & Niyogi, 2003). The Laplacian score of the j'th feature is denoted by ℓ_j and the i'th sample of the j'th feature by x_{ij}. First a nearest neighbor graph $\mathbf{G}$ is created with n nodes, where the i'th node

is denoted by $\mathbf{x}_i$. An edge is placed between nodes i_1 and i_2 if $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are "close", meaning $\mathbf{x}_{i_1}$ is among k nearest neighbors of $\mathbf{x}_{i_2}$ or $\mathbf{x}_{i_2}$ is among k nearest neighbors of $\mathbf{x}_{i_1}$.

Then the weight matrix S is created, modelling the local data structure by capturing the similarities between adjacent data points, where a higher value indicates a stronger proximity between the data points. Whenever nodes i_1 and i_2 are connected, s_{i_1,i_2} is calculated according to (3), where t is set to 0.1 as this is suitable constant for the simulated data in this research. If i_1 and i_2 are not connected, s_{i_1,i_2} is put to zero.

$$s_{i_1,i_2} = \exp\left(-\frac{\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|^2}{t}\right) \tag{3}$$

The Laplacian score for feature j can then be calculated following (4), where for feature j it holds that $\mathbf{x}_j = [x_{1j}, \ldots, x_{nj}]^T$, $\mathbf{D} = diag(\mathbf{S1}$ with $\mathbf{1}$ a vector of size n containing ones. Using this information the graph Laplacian $\mathbf{L}$ (Chung, 1997) can be calculated: $\mathbf{L} = \mathbf{D} - \mathbf{S}$.

$$\ell_j = \frac{\tilde{\mathbf{x}}_j^T L \tilde{\mathbf{x}}_j}{\tilde{\mathbf{x}}_j^T D \tilde{\mathbf{x}}_j}, \text{ where } \tilde{\mathbf{x}}_j = \mathbf{x}_j - \frac{\mathbf{x}_j^T D \mathbf{1}}{\mathbf{1}^T D \mathbf{1}} \mathbf{1} \tag{4}$$

After providing a Laplacian score for all features, the ultimate feature subset still needs to be determined. As a lower Laplacian score indicates a more important feature, the Laplacian scores are ordered from small to large. Then the number of features contained in the subset, denoted by f, is chosen using an iterative procedure, starting at $f = 5$ increasing till $f = p$ in steps of 5. At each step, a feature subset of size f is created, containing the m features with the lowest Laplacian score. Clustering is then performed using this subset, employing either k-means or AMA. The clustering performance is evaluated using the Adjusted Rand Index (ARI), introduced by (Hubert & Arabie, 1985), and the optimal number of features f is determined based on the highest ARI.

4.1.2 mRMR

The mRMR criterion for first-order incremental feature selection (Peng et al., 2005) is used to handle supervised feature selection in the two step approach. Feature selection methods aim to find a subset of features that are equipped to find the clustering optimally. When considering the input data $\mathbf{X} = \{\mathbf{x}_j, j = 1, \ldots, p\}$ with n instances and p features, this translates to finding the subspace $\mathbb{R}^m$ of feature space $\mathbb{R}^p$ that optimally characterizes the target classification $\mathbf{c}$, with f the number of selected features.

Often the optimal characterization is determined by selecting the features with the highest relevance to target classification $\mathbf{c}$, called maximal relevance (Max-Relevance). The relevance is usually characterized in terms of mutual information, measuring the dependency of variables. The mutual information of two random variables $\mathbf{x}$ and $\mathbf{y}$ is expressed via their probability density functions $p(\mathbf{x})$, $p(\mathbf{y})$ and $p(\mathbf{x}, \mathbf{y})$:

$$I(x, y) = \int\int p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right) dxdy \tag{5}$$

The m selected features $\mathbf{x}_j$ need to have the largest mutual information $I(\mathbf{x_j}, \mathbf{c})$, individually, as

this reflects the largest dependency on the target classification. The Max-Relevance is then calculated by searching the features that satisfy (6), where $D(\mathcal{S}, \mathbf{c})$ represents the joint dependency of the features in set $\mathcal{S}$ on target class $\mathbf{c}$.

$$\max D(\mathcal{S}, \mathbf{c}), \text{ where } D = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x}_j \in \mathcal{S}} I(\mathbf{x}_j, \mathbf{c}) \tag{6}$$

If two features are highly depended, the classification performance does not change much if one of them is removed. As the features chosen by Max-Relevance can have a rich redundancy, meaning that the dependency between these features is large, the minimal redundancy (Min-Redundancy) condition in 7 is added to select mutually exclusive features, where $R(\mathcal{S})$ represents the redundancy in feature set $\mathcal{S}$.

$$\min R(\mathcal{S}), \text{ where } R(\mathcal{S}) = \frac{1}{|\mathcal{S}|^2} \sum_{\mathbf{x}_j, \mathbf{x}_k \in \mathcal{S}} I(\mathbf{x}_j, \mathbf{x}_k) \tag{7}$$

The mRMR criterion is defined as $\Phi(D, R)$, shown in (8), by combining the Max-Relevance and Min-Redundancy constraints, described above. Following this maximization, ensures the optimization of D and R simultaneously.

$$\max \Phi(D, R), \text{ where } \Phi = D - R \tag{8}$$

In practice, incremental search methods can be employed to identify the features that are nearly optimal. When feature subset $\mathcal{S}_{f-1}$, containing $f-1$ features, is known, the task changes to selecting the f'th feature from the set $\mathbf{X} - \mathcal{S}_{f-1}$. This is achieved by selecting the feature that maximizes $\Phi(D, R)$, resulting in the following optimization:

$$\max_{\mathbf{x}_k \in \mathbf{X} - \mathcal{S}_{f-1}} \left[I(\mathbf{x}_k, \mathbf{c}) - \frac{1}{f-1} \sum_{\mathbf{x}_j \in \mathcal{S}_{f-1}} I(\mathbf{x}_k, \mathbf{x}_j) \right] \tag{9}$$

Determining the number of features f of the feature subset is handled in the following manner. A sequence with various options for f is made, starting at 5 increasing till p, the total number of features, in steps of 5. Then for all these possible f values the mRMR procedure is followed to determine the optimal feature subset. This subset is then used in the chosen clustering method, either k-means or AMA. The clustering performance is then determined by the ARI (Hubert & Arabie, 1985), where the highest index determines the number of features f.

4.2 Clustering Methods

The second step of the two-step approach is clustering. Clustering is either handled by k-means clustering or convex clustering, explained in Sections 4.2.1 and 4.2.2, respectively.

4.2.1 K-means Clustering

K-means clustering is a well known unsupervised learning algorithm, introduced by MacQueen (1967), partitioning the dataset into K clusters, where data points are assigned to clusters based on the nearest centroid. The objective of k-means is to minimize the within-cluster sum of squares (WCSS), calculated according to (10), where C_k denotes the set of data instances assigned to cluster k and μ_k the centroid of cluster k.

$$WCSS = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2, \text{ where } \boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i, \ \forall k = 1, \ldots, K \tag{10}$$

Specifically, k-means seeks to partition the observations into K clusters in such a manner that the WCSS is minimal. This results in the optimization problem (11), which is solved by an iterative algorithm. This algorithm is explained in detail in Appendix A.1.

$$\min_{\{C_1,\ldots,C_K\}} \sum_{k=1}^{K} \sum_{\mathbf{x_i} \in C_k} \|\mathbf{x_i} - \boldsymbol{\mu}_k\|^2 \tag{11}$$

4.2.2 Convex Clustering

Convex clustering is a recently proposed method (Pelckmans et al., 2005; Hocking et al., 2011; Lindsten et al., 2011), solving the minimization problem provided by (12). Here $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a data matrix with n instances and p features. This minimization aims to predict $\mathbf{A} \in \mathbb{R}^{n \times p}$.

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|_2^2 + \gamma \sum_{i_1 < i_2} \|\mathbf{a}_{i_1} - \mathbf{a}_{i_2}\|_q \tag{12}$$

Here $\mathbf{A}_i$ is the i'th row of $\mathbf{A}$, γ the tuning parameter and $\|.\|_q$ is the L_q-norm of a vector with $q \in \{1, 2, \infty\}$. The fused-lasso penalty, the second term, encourages identical rows in the solution $\hat{\mathbf{A}}$. Whenever two rows are identical, they are regarded as members of the same cluster. The number of unique rows in $\hat{\mathbf{A}}$ equals the number of estimated clusters, which is controlled by the tuning parameter γ. If $\gamma = 0$, there are no identical rows in $\hat{\mathbf{A}}$, resulting in every instance being a cluster by itself. When γ increases, $\hat{\mathbf{A}}$ contains some identical rows, demonstrating the fusion process. If γ is sufficiently large, all rows in $\hat{\mathbf{A}}$ are identical, implying that there is one single cluster for all instances. The solution $\hat{\mathbf{A}}$ depends on γ, providing a unique solution for each γ as the objective function in (12) is strictly convex.

A modification of convex clustering (12) is created by allowing an adaptive penalization through the addition of weights $w_{i_1,i_2} \geq 0$ to the fused-lasso penalty, as shown in (13).

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|_2^2 + \gamma \sum_{i_1 < i_2} w_{i_1,i_2} \|\mathbf{a}_{i_1} - \mathbf{a}_{i_2}\|_q \tag{13}$$

Convex clustering is implemented using AMA in the same manner as Chi and Lange (2015), explained in Appendix A.3. This algorithm determines the weights w_{i_1,i_2} by integrating the k-nearest-neighbors method with Gaussian kernel. In particular, the weight between instance pair (i_1, i_2) is calculated according to $w_{i_1,i_2} = \iota_{i_1,i_2}^k \exp -\phi \|\mathbf{x}_{i_1} - \mathbf{x}_{i_1}\|_2^2$, where ι_{i_1,i_2}^k equals 1 if

instance i_1 is among the k nearest neighbors of instance i_2 or conversely, and 0 otherwise. This weight selection performs effectively for a wide range of ϕ when k is small, resulting in the choice of fixing k at 5 and ϕ at 0.5.

The parameter γ is tuned using the Adjusted Rand Index (ARI) method, as outlined by B. Wang et al. (2018). Initially, a coarse grid search is conducted to explore different γ values. Following this, 50 bootstrap samples are utilized to compute the average ARI, thereby improving the robustness of the parameter tuning process. This approach aligns with stability selection techniques, which aim to find parameters that produce consistent clustering outcomes despite slight variations in the training data.

4.3 Sparse Clustering Methods

In the integrated approach, feature selection is handled during the clustering process, for which the integrated approaches of k-means clustering and convex clustering are utilized to make a fair comparison between the two-step approach and the integrated approach. This results in the implementation of sparse k-means clustering and sparse convex clustering, which are explained in Sections 4.3.1 and 4.3.2, respectively.

4.3.1 Sparse K-means Clustering

Witten and Tibshirani (2010) introduced sparse k-means clustering in order to improve clustering accuracy for high dimensional data. In order to formulate sparse k-means clustering, Witten and Tibshirani (2010) rewrote the WCSS of k-means to the formula represented in (14). Here n_k denotes the number of observations in cluster k and C_k represents set containing the indices of the observations in cluster k. The dissimilarity measure between two observations (i_1, i_2) along feature j is denoted by d_{i_1,i_2}, which is calculated using the squared Euclidean distance: $d_{i_1,i_2,j} = \|x_{i_1,j} - x_{i_2,j}\|^2$.

$$WCSS = \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i_1,i_2 \in C_k} \sum_{j=1}^{p} d_{i_1,i_2,j} \tag{14}$$

The between-cluster sum of squares (BCSS), introduced by Witten and Tibshirani (2010), can then reframe the minimization of WCSS to the maximization of BCSS, where the BCSS is calculated according to (15).

$$BCSS = \sum_{j=1}^{p} \left(\frac{1}{n} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} d_{i_1,i_2,j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i_1,i_2 \in C_k} d_{i_1,i_2,j} \right) \tag{15}$$

Sparse k-means clustering is created by the addition of weights w_j, corresponding to the weight for feature j, and tuning parameter s to the optimization problem, resulting in the following optimization:

$$\max_{C_1,\ldots,C_K,\mathbf{w}} \left\{ \sum_{j=1}^{p} w_j \left(\frac{1}{n} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} d_{i_1,i_2,j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i_1,i_2 \in C_k} d_{i_1,i_2,j} \right) \right\} \tag{16}$$
$$\text{subject to } \|\mathbf{w}\|^2 \leq 1,\ \|\mathbf{w}\|_1 \leq s,\ w_j \geq 0\ \forall j$$

The weights w_j can be interpreted as the contribution of feature j to the resulting sparse clustering, where a large value indicates a great contribution of feature j to the clustering solution, while $w_j = 0$ means that feature j is excluded from the clustering. Determining which features are excluded is done using the lasso penalty on $\mathbf{w}$: $\|\mathbf{w}\|_1 \leq s$. This ensures the sparsity, when the tuning parameter s is small. In this manner, some of the weights w_j equal zero, and will thus not influence the clustering outcome. The tuning parameter s should satisfy $1 \leq s \leq \sqrt{p}$.

The optimization is handled by an iterative algorithm, with the value of the tuning parameter s determined using a permutation approach. In the iterative algorithm, the feature weights $\mathbf{w}$ are determined, followed by the clustering. Appendix A.2 explains these procedures in detail.

4.3.2 Sparse Convex Clustering

B. Wang et al. (2018) introduces sparse convex clustering, starting with a reformulation of (13) in order to easily add the sparsity to the objective function. The data matrix $\mathbf{X}$ is denoted in feature-level as column vector: $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$, with $\mathbf{x}_j = (x_{1j}, \ldots, x_{nj})^T \quad \forall j = 1, \ldots, p$. The matrix $\mathbf{A}$ is also denoted in feature-level as column vector: $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_p)$. Without loss of generality, B. Wang et al. (2018) assumes that the feature vectors are centered, meaning $\sum_{i=1}^{n} x_{ij} = 0 \quad \forall j = 1, \ldots, p$. Now (13) can be rewritten to (17) using simple algebra as well as the introduction of $\mathcal{E} = \{l = (i_1, i_2) : 1 \leq i_1 < i_2 \leq n\}$.

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{j=1}^{p} \|\mathbf{x}_j - \mathbf{a}_j\|_2^2 + \gamma \sum_{l \in \mathcal{E}} w_l \|\mathbf{a}_{i_1} - \mathbf{a}_{i_2}\|_q \tag{17}$$

Let $\hat{\mathbf{A}} = (\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_n)^T = (\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_p)$ be the solution of (17) for a given γ. Just as for convex clustering the instance-level row estimates $\hat{\mathbf{a}}_i, i = 1, \ldots, n$ determine the cluster structure, meaning that two identical rows of $\hat{\mathbf{A}}$ belong to the same cluster. On the other hand, the instance-level column estimates $\hat{\mathbf{a}}_j, j = 1, \ldots, p$ determine the feature importance, meaning that feature j is not informative whenever $\hat{\mathbf{a}}_j$ has identical components. As the feature vectors are centered, feature j is not informative if and only if $\|\hat{\mathbf{a}}_j\|_2^2 = \sum_{i=1}^{n} \hat{a}_{ij}^2 = 0$.

Within high dimensional clustering, there is a desire to incorporate sparsity into convex clustering, resulting in a sparse solution $\hat{\mathbf{A}}$ with exact $\mathbf{0}$'s for some of its columns. To eliminate uninformative features and achieve a sparse solution, B. Wang et al. (2018) integrates an adaptive group-lasso penalty (Yuan & Lin, 2006; H. Wang & Leng, 2008) into objective function (17), which is then referred to as sparse convex clustering. Specifically, sparse convex clustering solves (18), where tuning parameters γ_1 and γ_2 control the cluster size and number of informative features, respectively. The group-lasso penalty contains the weight u_j, crucial for adaptively penalizing the features.

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{j=1}^{p} \|\mathbf{x}_j - \mathbf{a}_j\|_2^2 + \gamma_1 \sum_{l \in \mathcal{E}} w_l \|\mathbf{a}_{i_1} - \mathbf{a}_{i_2}\|_q + \gamma_2 \sum_{j=1}^{p} u_j \|\mathbf{a}_j\|_2 \tag{18}$$

Sparse convex clustering is implemented using S-AMA following the implementation of B. Wang et al. (2018), explained in Appendix A.4. This algorithm determines the weights w_{i_1,i_2} by integrating the k-nearest-neighbors method with Gaussian kernel in the same manner as in AMA (see Section 4.2.2). The weights u_j are set to $1/\|\hat{\mathbf{a}}_j^{(0)}\|_2$, with $\hat{\mathbf{a}}_j^{(0)}$ the estimate of $\mathbf{a}_j$

in (18) using $\gamma_2 = 0$. These weights were used by B. Wang et al. (2018) and Zou (2006) regarding the adaptive group-lasso penalty. This selection of weights penalizes informative features less, while penalizing uninformative features more. Consequently, it improves clustering accuracy and variable selection performance to its non-adaptive counterpart.

The parameters γ_1 and γ_2 are tuned using the Adjusted Rand Index (ARI) method, as outlined by B. Wang et al. (2018). Initially, a coarse grid search is conducted to explore different γ_1 values, while a fine grid search is conducted for γ_2. Following this, 50 bootstrap samples are utilized to compute the average ARI, thereby improving the robustness of the parameter tuning process. This approach aligns with stability selection techniques, which aim to find parameters that produce consistent clustering outcomes despite slight variations in the training data.

In accordance with B. Wang et al. (2018), the weights w_{i_1,i_2} and u_j are rescaled to have a total sum of $1/\sqrt{p}$ and $1/\sqrt{n}$, respectively. This adjustment guarantees that the optimal tuning parameters γ_1 and γ_2 fall within a relatively robust interval, independent of the feature dimension and sample size. This rescaling serves only for convenience and does not impact the final clustering path.

5 Numerical Results

In this section the empirical findings of this research are presented. The focus is on comparing clustering methods for high-dimensional data, particularly examining the impact of integrated and preliminary feature selection on clustering performance. To facilitate a comprehensive comparison, first clustering without feature selection is performed using k-means and convex clustering (handled by AMA). Subsequently, the clustering is enhanced with either preliminary or integrated feature selection. Preliminary feature selection is handled by the Laplacian score (unsupervised) or by mRMR (supervised), followed by either k-means clustering or AMA, making it a two-step approach. Integrated feature selection is handled using sparse k-means clustering and sparse convex clustering (handled by S-AMA).

As the focus is on determining which methods perform best for high-dimensional data with possibly lots of noise, various simulated datasets are considered. The following four simulation settings are considered, where for each setting the σ values 1, 1.5 and 2 are used for different noise levels. Setting 1: $K = 2$, $n = 60$, $p = 150$ and $\mu = 0.6$, setting 2: $K = 2$, $n = 60$, $p = 500$ and $\mu = 0.7$, setting 3: $K = 4$, $n = 60$, $p = 150$ and $\mu = 0.9$ and setting 4: $K = 4$, $n = 60$, $p = 500$ and $\mu = 1.2$. In order to have stable results, all methods are run on 200 simulated datasets for each setting, taking the average and standard deviation to discuss the performance.

To evaluate clustering performance, we use the Adjusted Rand Index (ARI), as introduced by Hubert and Arabie (1985). The ARI measures the similarity between two data clusters, adjusting for random chance. The ARI ranges from -1 to 1, where 1 indicates perfect agreement between clusters, 0 indicates the level of agreement expected by random chance, and negative values indicate less agreement than expected by random chance. The ARI is particularly useful for assessing cluster quality as it accounts for the possibility of random clustering matches.

In addition to ARI, the False Negative Rate (FNR) and False Positive Rate (FPR) are considered to provide a complete evaluation of the clustering methods. Low values of FNR and

FPR indicate better cluster separation and assignment accuracy, complementing the insights provided by ARI.

5.1 Exclusion of AMA and S-AMA

Throughout the simulations of the four settings, both AMA and S-AMA consistently resulted in an ARI of zero, indicating that the produced clusters are random. This means that either all data instances belong to separate clusters or all data instances belong to the same cluster. Various gamma grids were explored, collectively containing thousands of values, to optimize the performance of AMA and S-AMA. However, despite this thorough exploration, all configurations yielded an ARI of zero, confirming that these algorithms were unable to form meaningful clusters for the simulated high-dimensional datasets.

For AMA, this outcome was expected. It is well known and documented that convex clustering algorithms like AMA struggle with high-dimensional data due to their sensitivity to noise and the curse of dimensionality. Thus, the inability to effectively cluster high-dimensional data is a known limitation of AMA.

However, the poor performance of S-AMA was unexpected. S-AMA is designed to improve clustering performance for high-dimensional data by integrating sparsity, with an expectation of better cluster performance. S-AMA was implemented using the guidelines provided by B. Wang et al. (2018), as explained in Section 4.3.2. Using these guidelines, S-AMA consistently results in an ARI of zero, while B. Wang et al. (2018) indicates that it should provide an ARI between 0.8 and 1. This underperformance can be explained in two parts: parameter selection and sparsity weight calculation.

First, B. Wang et al. (2018) does not provide clear instructions on the selection of the grids for γ_1 and γ_2, making it difficult to find appropriate values. Their research does state that γ_1 can be found using a coarse grid and γ_2 using a fine grid. Many optional grids have been used in this study for both γ_1 and γ_2. For γ_1 various values have been used ranging from 0.1 to 100. To determine γ_2 linear and exponential ranges have been used, resulting in the exploration of γ_2 values ranging from 0.005 till $10,000$. However, all of these values result in an ARI of zero. As the tuning parameters γ_1 and γ_2 significantly impact the clustering performance, it is challenging to achieve optimal clustering without additional insights.

Second, the calculation of the sparsity weights $\mathbf{u}$ should result in a higher weight for uninformative features compared to informative ones, thereby ensuring a higher penalization for uninformative features. According to B. Wang et al. (2018), these weights should be calculated using $u_j = 1/\|\hat{\mathbf{a}}_j^{(0)}\|_2$, with $\hat{\mathbf{a}}_j^{(0)}$ being the estimate of $\mathbf{a}_j$ when $\gamma_2 = 0$. However, this approach does not necessarily guarantee the higher penalization of uninformative features, as $\mathbf{u}$ is solely determined by the initial estimate of $\mathbf{A}$ using $\gamma_2 = 0$, which may not accurately capture the true informativeness of features whenever γ_2 is not zero. Since the sparsity weights $\mathbf{u}$ play a crucial role in determining the importance of features, an inaccurate calculation can influence the clustering performance negatively by prioritizing uninformative features over informative features.

Furthermore, the GitHub page[1] of B. Wang et al. (2018) provides an example of implementing

[1]The GitHub page can be found using the following link: https://github.com/elong0527/scvxclustr

S-AMA. In this example, the values of γ_1 and γ_2 are provided instead of computed using the method described in their paper. Additionally, the weights $\mathbf{u}$ are not calculated using the described approach. Rather, a vector is created with value 0.5 for informative features and 1 for uninformative features, deviating from their own proposed calculation. Since researches typically depend on published methodologies to reproduce and validate results, these deviations undermine the ability of other researchers to replicate and extend the findings.

The identified discrepancies, along with the absence of a robust method to determine tuning parameters and sparsity weights, results in the underperformance of S-AMA. Since the sparsity weights deviate from the intended structure, the S-AMA procedure potentially mimics the behavior of the AMA procedure, undermining the benefits of integrating sparsity into the clustering process.

Given the consistent poor clustering performance of AMA and S-AMA for the high-dimensional data settings, these algorithms have been excluded from further analysis. Additionally, this results in the exclusion of the two-step approach using AMA, as there can not be a comparison of the integrated and preliminary approach. The subsequent results focus on k-means clustering and its variations with preliminary and integrated feature selection.

5.2 Clustering Performance K-means

This section discusses the clustering performance of various k-means methods. To ensure a fair comparison between preliminary and integrated feature selection, unsupervised feature selection is utilized. This analysis includes the clustering performance of k-means, sparse k-means and Laplacian k-means, as discussed in Section 5.2.1. Additionally, Section 5.2.2 examines the effects of using supervised feature selection compared to unsupervised feature selection in a two-step approach, where the supervised feature selection is handled by mRMR.

5.2.1 Preliminary vs. Integrated Unsupervised Feature Selection

As previously emphasized, this research focuses on determining the best way to enhance clustering with feature selection. In order to achieve this, two approaches are used: performing feature selection before clustering (preliminary feature selection) or performing feature selection during clustering (integrated feature selection). Specifically, this part of the study compares k-means clustering combined with the Laplacian score and sparse k-means clustering against traditional k-means clustering.

Figure 1 shows the average ARI for the methods over the four data simulation settings. For each data simulation the average ARI is shown for different noise levels: low noise level ($\sigma = 1$), medium noise level ($\sigma = 1.5$) and high noise level ($\sigma = 2$). Here it is important to remember that setting 1 and 2 consists of 2 clusters, while settings 3 and 4 consists of 4. For all these methods, K is preliminary put to either 2 or 4 based on the data simulation. Another key difference in the settings, is the number of features. For simulation 1 and 3 this is 150, while for simulation 2 and 4 this is 500.

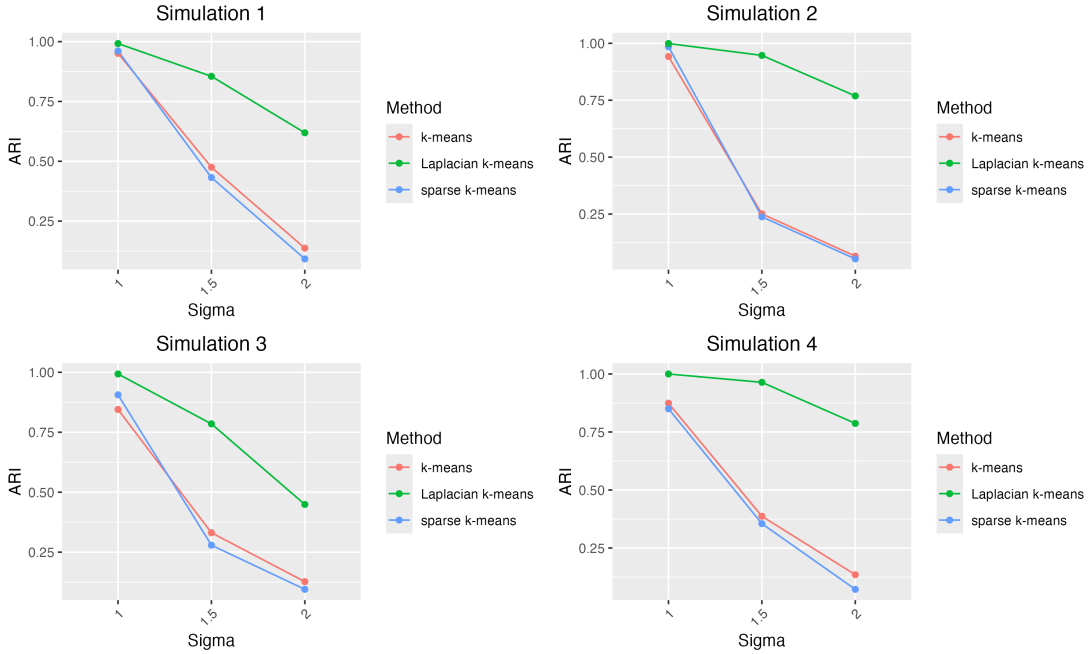

Figure 1: Average ARI of k-means, Laplacian k-means and sparse k-means for multiple simulation settings and noise levels.

When looking at the low level of noise ($\sigma = 1$), it is clear that all three methods work very well. For all simulation settings k-means combined with the Laplacian score provides the highest ARI. For simulation 1 and 2, the traditional k-means and sparse k-means provide very close ARI values compared to the Laplacian k-means, where the sparse k-means is a bit higher than the regular k-means. For simulations 3 and 4, we observe a slight decline in the ARI of traditional k-means and sparse k-means compared to simulations 1 and 2. This is expected, as clustering high-dimensional data with 4 clusters instead of 2 increases complexity and noise sensitivity. Additionally, the curse of dimensionality and challenges in feature relevance make it more difficult to accurately distinguish clusters, resulting in a lower ARI.

Whenever the noise level is increased, to either a medium ($\sigma = 1.5$) or a high ($\sigma = 2$) level, Laplacian k-means still provides good clusters, as the ARI remains quite high. A steeper decrease in ARI is shown whenever there are 150 features compared to 500 features, because Laplacian k-means, a two-step approach where the Laplacian score is used to determine the optimal subset of features, benefits from a larger feature pool. This allows it to identify and select the most relevant features more effectively, leading to higher ARI values even with increased dimensionality. The ARI decreases a bit more when the noise is increased, but Laplacian k-means still provides the best clustering performance of the three methods.

In contrast, when considering k-means and sparse k-means clustering for either the medium ($\sigma = 1.5$) or high ($\sigma = 2$) noise level, a rapid decrease in the ARI occurs. As the increased noise obscures the clear separation between clusters, it becomes harder to accurately assign data instances, resulting in a lower clustering accuracy. The graph even shows that sparse k-means results in a slightly smaller ARI compared to traditional k-means. Sparse k-means handles

high-dimensional data by selecting a subset of features, which can be done overly aggressive under noisy conditions. When this happens, relevant features for the clustering can be omitted, resulting in a poorer clustering performance compared to traditional k-means, which uses all features and could still capture some relevant information regardless of the noise.

When considering 2 clusters for k-means and sparse k-means, a rapid decrease is shown in ARI with an increase in number of features, due to the curse of dimensionality which amplifies noise sensitivity and dilutes the relevance of meaningful features. However, when considering 4 clusters, there is a steeper decrease in ARI for simulation 3 with 150 features compared to simulation 4 with 500 features, as the increased number of clusters in a moderately high-dimensional space intensifies the challenges of cluster separation and noise sensitivity more acutely than in an extremely high-dimensional space, where feature dilution somewhat stabilizes the clustering process.

Additional analysis of the FNR and FPR, included in Appendix B, shows that Laplacian k-means consistently results in lower FNR and FPR across different noise levels and dimensional settings. This indicates that Laplacian k-means not only results in a high overall clustering accuracy (as indicated by ARI) but also transcends in maintaining cluster integrity and separation, making less errors in data instance assignment to the true clusters.

Furthermore, Laplacian k-means shows lower variability in most cases compared to traditional k-means and sparse k-means clustering, as shown by the smaller standard deviation of the ARI, FNR and FPR, included in Appendix B. This indicates that Laplacian k-means has a more consistent performance, further supporting the robustness and reliability of Laplacian k-means in providing high-quality clusters.

Overall, these observations confirm that Laplacian k-means is the superior method when considering high-dimensional data with possible noise, as it consistently provides a higher ARI with lower variability while maintaining lower FNR and FPR. It effectively selects informative features across different levels of noise and dimensional settings, demonstrating its robustness in feature selection and clustering accuracy.

5.2.2 Preliminary Feature Selection - Supervised vs. Unsupervised

In addition to the comparison of preliminary and integrated feature selection, a comparison is made between supervised and unsupervised feature selection. This comparison can only be done for the two-step approach, as the integrated approach selects features during clustering, making it unsupervised. The unsupervised preliminary feature selection is again handled by the Laplacian score, while the supervised one is handled by mRMR. Both are then combined with k-means to compare clustering performance.

The clustering performance of these two methods is again compared using the ARI, shown in Figure 2 for the various data settings and noise levels. Remember that settings 1 and 3 contain 150 features, while settings 2 and 4 contain 500 features. The number of clusters varies per data simulation and is directly used as K in k-means clustering, which equals 2 for simulation 1 and 2 and 4 for simulation 3 and 4. There are three noise level options, namely low ($\sigma = 1$), medium ($\sigma = 1.5$) and high ($\sigma = 2$).

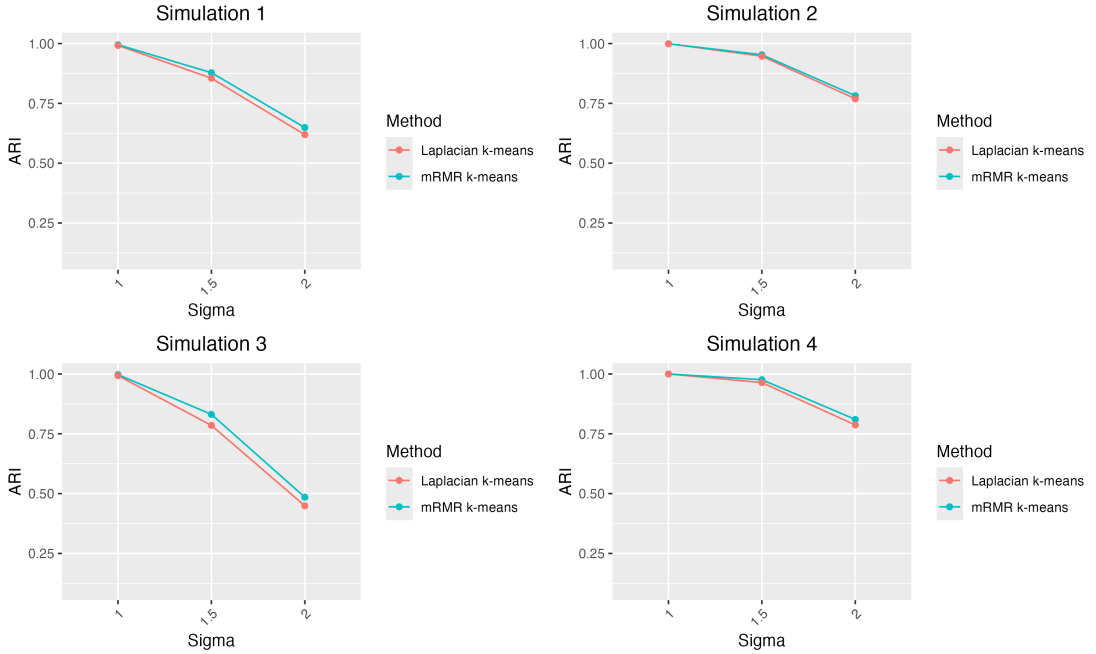

Figure 2: Average ARI of Laplacian k-means and mRMR k-means for multiple simulation settings and noise levels.

Surprisingly, across all data simulations and noise levels, Laplacian k-means and mRMR k-means show similar results. Since mRMR k-means uses the clustering labels to perform feature selection, it is expected to outperform Laplacian k-means, which does not use these labels. Figure 2 indicates that mRMR k-means performs better than Laplacian k-means, but the difference is minimal.

The similarity in results between these two methods can possibly be explained by the determination of the number of features retained after feature selection. This number is determined by running the algorithm across various options, selecting the one that results in the highest ARI. As both methods aim to maximize the ARI, the performance of these methods are naturally aligned, making the similar results less surprising.

Table 1 provides insights into the number of features selected by each method for various simulation settings and noise levels. For the low noise level, both methods select approximately 20 features, which is the number of informative features in the simulated data. As the noise increases, the number of selected features increases as well. This is expected, as higher noise can make some uninformative features seem relevant. Still the number of selected features remains close to the 20 informative features, indicating that both methods effectively select the number of features by maximizing the ARI.

Table 1: Average number of selected features of Laplacian k-means and mRMR k-means for multiple simulation settings and noise levels

		Low Noise		Medium Noise		High Noise	
	Methods	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**
Simulation 1	**Laplacian k-means**	20.6	16.9	32.6	24.5	34.3	25.4
	mRMR k-means	20.5	19.6	34.2	25.6	34.4	22.9
Simulation 2	**Laplacian k-means**	18.3	32.0	33.4	48.2	33.0	23.4
	mRMR k-means	13.9	6.0	35.7	38.1	36.9	33.0
Simulation 3	**Laplacian k-means**	22.7	11.4	28.5	12.7	31.7	17.8
	mRMR k-means	24.7	17.5	31.0	16.8	29.6	12.1
Simulation 4	**Laplacian k-means**	21.1	23.6	28.2	19.6	30.0	13.9
	mRMR k-means	19.1	4.8	29.3	18.7	31.9	17.5

The high standard deviation in the selection of features indicates significant variability across the 200 runs, suggesting that the chosen number of features can fluctuate considerably depending on the specific noise conditions and data structure. This means that the number of selected features should depend on the specific data at hand in order to generate optimal clusters.

Overall, these findings suggest that the benefits of supervised feature selection in the two-step approach are minimal compared to unsupervised preliminary feature selection. This implies that using clustering labels in mRMR does not significantly improve clustering compared to the unsupervised feature selection by the Laplacian score. However, it is also possible that the local data structure and the process of determining of the number of selected features, based on maximizing the ARI, are more critical for clustering performance than the type of feature selection used.

6 Conclusion

Throughout this study, the combined efficacy of feature selection and clustering to enhance clustering performance for high-dimensional data is explored. Two approaches are considered: preliminary feature selection, performed before clustering, and integrated feature selection, embedding feature selection in the clustering process. Preliminary feature selection uses either the Laplacian score (unsupervised) or mRMR (supervised), followed by clustering via k-means or convex clustering, implemented using AMA. Integrated feature selection is managed by sparse k-means and sparse convex clustering, implemented using S-AMA.

The findings indicate that for k-means clustering, both Laplacian k-means and mRMR k-means outperform traditional k-means and sparse k-means across various data settings and noise levels. The results of sparse k-means clustering show that the integrated method struggles with increased noise, often resulting in lower clustering performance than traditional k-means. However, the preliminary methods are particularly effective for high-dimensional data, maintaining strong clustering performance even with additional noise. The performance gap between supervised and unsupervised preliminary feature selection is minimal, suggesting that both approaches are effective.

Conversely, for convex clustering, a similar conclusion can not be drawn, as AMA and S-

AMA are excluded from the research due to their consistent poor clustering performance. This poor performance resulted from discrepancies and the lack of a robust method for determining tuning parameters and sparsity weights.

Future research should extend these methodologies to unlabeled data, requiring to replace ARI by alternative performance measures. Additionally, it would be beneficial to determine the number of clusters K in k-means clustering using a permutation approach rather than setting it manually, potentially increasing the robustness of the results.

References

Belkin, M. & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, *14*.

Bolón-Canedo, V., Sánchez-Maroño, N. & Alonso-Betanzos, A. (2013). A review of feature selection methods on synthetic data. *Knowledge and information systems*, *34*, 483–519.

Boyd, S., Parikh, N., Chu, E., Peleato, B. & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, *3*(1), 1–122.

Cai, J., Luo, J., Wang, S. & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79.

Chi, E. C. & Lange, K. (2015). Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, *24*(4), 994–1013.

Chung, F. R. (1997). *Spectral graph theory* (Vol. 92). American Mathematical Soc.

Dash, M. & Liu, H. (1999). Handling large unsupervised data via dimensionality reduction. In *1999 acm sigmod workshop on research issues in data mining and knowledge discovery.*

Gabay, D. & Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, *2*(1), 17–40.

Glowinski, R. & Marroco, A. (1975). Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, *9*(R2), 41–76.

Guo, J., Levina, E., Michailidis, G. & Zhu, J. (2010). Pairwise variable selection for high-dimensional model-based clustering. *Biometrics*, *66*(3), 793–804.

Hall, M. A. (2000). Correlation-based feature selection of discrete and numeric class machine learning.

Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

He, X., Cai, D. & Niyogi, P. (2005). Laplacian score for feature selection. *Advances in neural information processing systems*, *18*.

He, X. & Niyogi, P. (2003). Locality preserving projections. *Advances in neural information processing systems*, *16*.

Hocking, T. D., Joulin, A., Bach, F. & Vert, J.-P. (2011). Clusterpath: An algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning* (p. 1).

Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*, 193–218.

Kaufman, L. & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

Kira, K. & Rendell, L. A. (1992). A practical approach to feature selection. In *Machine learning proceedings 1992* (pp. 249–256). Elsevier.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning* (pp. 171–182).

Kumar, V. & Minz, S. (2014). Feature selection. *SmartCR*, *4*(3), 211–229.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J. & Liu, H. (2017). Feature selection: A data perspective. *ACM computing surveys (CSUR)*, *50*(6), 1–45.

Lindsten, F., Ohlsson, H. & Ljung, L. (2011). Clustering using sum-of-norms regularization: With application to particle filter output computation. In *2011 ieee statistical signal processing workshop (ssp)* (pp. 201–204).

Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, *17*(4), 491–502.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Mitra, P., Murthy, C. & Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, *24*(3), 301–312.

Murtagh, F. & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 86–97.

Pan, W. & Shen, X. (2007). Penalized model-based clustering with application to variable selection. *Journal of machine learning research*, *8*(5).

Pelckmans, K., De Brabanter, J., Suykens, J. A. & De Moor, B. (2005). Convex clustering shrinkage. In *Pascal workshop on statistics and optimization of clustering workshop.*

Peng, H., Long, F. & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, *27*(8), 1226–1238.

Raftery, A. E. & Dean, N. (2006). Variable selection for model-based clustering. *Journal of the American Statistical Association*, *101*(473), 168–178.

Saeys, Y., Inza, I. & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, *23*(19), 2507–2517.

Sun, W., Wang, J. & Fang, Y. (2012). Regularized k-means clustering of high-dimensional data and its asymptotic consistency.

Tibshirani, R., Walther, G. & Hastie, T. (2001). 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.

Tseng, P. (1991). Applications of a splitting algorithm to decomposition in convex programming

and variational inequalities. *SIAM Journal on Control and Optimization*, *29*(1), 119–138.

Vichi, M., Cavicchia, C. & Groenen, P. J. (2022). Hierarchical means clustering. *Journal of Classification*, *39*(3), 553–577.

Wang, B., Zhang, Y., Sun, W. W. & Fang, Y. (2018). Sparse convex clustering. *Journal of Computational and Graphical Statistics*, *27*(2), 393–403.

Wang, H. & Leng, C. (2008). A note on adaptive group lasso. *Computational statistics & data analysis*, *52*(12), 5277–5286.

Wang, S. & Zhu, J. (2008). Variable selection for model-based high-dimensional clustering and its application to microarray data. *Biometrics*, *64*(2), 440–448.

Wang, Y., Fang, Y. & Wang, J. (2016). Sparse optimal discriminant clustering. *Statistics and Computing*, *26*, 629–639.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*(301), 236–244.

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

Xie, B., Pan, W. & Shen, X. (2008). Variable selection in penalized model-based clustering via regularization on grouped parameters. *Biometrics*, *64*(3), 921–930.

Yuan, M. & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *68*(1), 49–67.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, *101*(476), 1418–1429.

A Algorithms

In this Appendix the algorithms used for clustering and sparse clustering are explained in detail. Starting with the k-means clustering algorithm in Appendix A.1, followed by the algorithm for sparse k-means clustering in Appendix A.2, which also includes a algorithm for finding the tuning parameter. Afterwards AMA, used for convex clustering, and S-AMA, used for sparse convex clustering, are explained in Appendices A.3 and A.4, respectively.

A.1 K-means

As previously mentioned, k-means clustering employs an iterative algorithm to partition a dataset into K clusters by assigning data instances to the nearest centroid. This Appendix focuses on explaining this iterative algorithm, resolving the following minimization:

$$\min_{\{C_1,\dots,C_K\}} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x_i} - \boldsymbol{\mu}_k\|^2 \text{ where } \boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i, \ \forall k = 1, \dots, K \tag{19}$$

In order to solve this minimization K centroids are randomly initialized: $\{\boldsymbol{\mu}_1^0, \dots, \boldsymbol{\mu}_K^0\}$. Then each data instance $\mathbf{x}_i$ is assigned to its nearest centroid following (20), creating K clusters: $\{C_1, \dots, C_K\}$.

$$C_k^m = \{\mathbf{x}_i : \|\mathbf{x}_i - \boldsymbol{\mu}_k^m\|^2 \leq \|\mathbf{x}_i - \boldsymbol{\mu}_k^m\|^2, \forall k \in 1, \dots, K\} \tag{20}$$

Next, the centroid $\boldsymbol{\mu}_k$ is updated to equal the mean of all data instances contained in cluster k according to (21), which is done for all K clusters.

$$\boldsymbol{\mu}_k^{m+1} = \frac{1}{|C_k^m|} \sum_{\mathbf{x}_i \in C_k^m} \mathbf{x}_i \tag{21}$$

The algorithm runs until convergence, meaning it stops when the assignment of data instances to clusters remains the same, or when the centroids change very little, below a predefined threshold. The complete algorithm can be defined as follows:

Algorithm 1 Implementing k-means clustering

1. Initialize $\{\boldsymbol{\mu}_1^0, \ldots, \boldsymbol{\mu}_K^0\}$.
2. For $i = 1, \ldots, n$, assign $\mathbf{x}_i$ to a cluster based on the nearest centroid $\boldsymbol{\mu}_k^m$:
   $C_k^m = \{\mathbf{x}_i : \|\mathbf{x}_i - \boldsymbol{\mu}_k^m\|^2 \leq \|\mathbf{x}_i - \boldsymbol{\mu}_k^m\|^2, \forall k \in 1, \ldots, K\}$
3. For $k = 1, \ldots, K$, do
   $\boldsymbol{\mu}_k^{m+1} = \frac{1}{|C_k^m|} \sum_{\mathbf{x}_i \in C_k^m} \mathbf{x}_i$
4. Repeat Steps 2-3 until convergence.

A.2 Sparse k-means

Implementing sparse k-means consists of two step. First, a permutation approach is used to determine the tuning parameter. Then, this tuning parameter is used in an iterative procedure to determine the feature weights and clusters. These steps are explained in Appendices A.2.1 and A.2.2.

A.2.1 Selection of the Tuning Parameter

The tuning parameter s is quite important for obtaining good clusters using sparse k-means clustering. The tuning parameter s determines the L_1 bound on the feature weights $\mathbf{w}$ in (16). Determining the value of s is a complicated procedure, as s can not be chosen in such a way that it maximizes the objective function in (16), since the objective increases with an increase in s. Instead, a permutation approach was introduced by Witten and Tibshirani (2010) to determine the tuning parameter.

First, B permuted datasets $(\mathbf{X}_1, \ldots, \mathbf{X}_B)$ are generated by independently permuting the instances of each feature to the original dataset $\mathbf{X}$, to assess the significance of different tuning parameters. Subsequently, a sequence of candidate tuning parameters $\mathcal{S}$ is introduced. For every $s_c \in \mathcal{S}$, the objective $O(s_c)$ obtained by performing sparse k-means with tuning parameter s_c on dataset $\mathbf{X}$ is calculated. Additionally, the objective is calculated for the permuted datasets $\{\mathbf{X}_b : b = 1, \ldots, B\}$, where the objective is denoted by $O_b(s_c)$.

The optimal tuning parameter is then determined through the gap statistic (Tibshirani, Walther & Hastie, 2001), which evaluates clustering performance on the original data compared to clusters generated from random data not containing subgroups. The gap statistic is calculated for every $s_c \in \mathcal{S}$ according to (22), where the highest gap statistics determines the value of the tuning parameter. This assessment remains robust as the features in the permuted datasets are

uncorrelated with each other, even when strong correlations exist among features in the original data $\mathbf{X}$.

$$Gap(s_c) = \log(O(s_c)) - \frac{1}{B}\sum_{b=1}^{B}\log(O_b(s_c)) \tag{22}$$

In this research B, the number of permuted datasets, is set to 50 to ensure robust determination the tuning parameter. The sequence $\mathcal{S}$ containing possible tuning parameters is based on the criteria that $1 \leq s \leq \sqrt{p}$, where p represents the number of features. This comprises 50 values, enabling a comprehensive exploration of potential tuning parameters.

A.2.2 Iterative Algorithm

After determining the tuning parameter s, an iterative algorithm is used to determine the clusters, summarized in Algorithm 2. In the optimization of sparse k-means, as shown in (16), a weight is assigned to each feature, based on its contribution to the increase in BCSS, iterating until convergence. First, the optimization is completed using fixed feature weights, meaning $w_1 = \ldots = w_p = \frac{1}{\sqrt{p}}$. This results it the optimization of (16) with respect to $C_1, \ldots, C_K$, following (23). This is done using the standard k-means algorithm on the dissimilarity matrix $(n \times n)$ with element $(i_1, i_2) = \sum_{j=1}^{p} w_j d_{i_1,i_2}$. The optimization is thus simplified to a clustering problem, using a weighted dissimilarity measure.

$$\min_{C_1,\ldots,C_K}\left\{\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i_1,i_2\in C_k}\sum_{j=1}^{p} w_j d_{i_1,i_2}\right\} \tag{23}$$

Second, the optimization is completed for fixed $C_1, \ldots, C_K$, assigning weights to the features based on the BCSS, prioritizing features with a larger BCSS. This means that (16) is optimized with respect to $\mathbf{w}$ holding $C_1, \ldots, C_K$ fixed. According to Witten and Tibshirani (2010) the feature weights $\mathbf{w}$ can then be calculated in the following manner:

$$\mathbf{w} = \frac{S(\mathbf{h}_+, \Delta)}{\|S(\mathbf{h}_+, \Delta)\|_2}, \text{ where } h_j = \frac{1}{n}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n} d_{i_1,i_2} - \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i_1,i_2\in C_k} d_{i_1,i_2} \tag{24}$$

Here, $\mathbf{h}_+$ denotes the positive part of $\mathbf{h}$ and S the soft-thresholding operator defined by $S(\mathbf{h}_+, \Delta) = sign(\mathbf{h}_+(|\mathbf{h}_+| - \Delta)_+$. The Δ value depends on whether $\|\mathbf{w}\|_1$ is smaller than the tuning parameter s or not. Whenever $\|\mathbf{w}\|_1 < s$, Δ is put to zero, otherwise $\Delta > 0$ is chosen to yield $\|\mathbf{w}\|_1 = s$.

Algorithm 2 Implementing sparse k-means clustering

1. Initialize $\mathbf{w}$ as $w_j = \frac{1}{\sqrt{p}}$ for $j = 1, \ldots, p$
2. Complete optimization for fixed $\mathbf{w}$ with respect to $C_1, \ldots, C_K$:
   $\min_{C_1,\ldots,C_K}\left\{\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i_1,i_2\in C_k}\sum_{j=1}^{p} w_j d_{i_1,i_2}\right\}$
3. Complete optimization for fixed $C_1, \ldots, C_K$ with respect to $\mathbf{w}$, resulting in:
   $\mathbf{w} = \frac{S(\mathbf{h}_+,\Delta)}{\|S(\mathbf{h}_+,\Delta)\|_2}$
4. Repeat Steps 2-3 until convergence.

This algorithm iterates until convergence. However, the convergence will generally not lead to the global optimum of (16), as k-means clustering is used whenever the feature weights are fixed, and k-means generally does not lead to the global optimum (Witten & Tibshirani, 2010).

A.3 AMA

Convex clustering is implemented using AMA, as introduced by Chi and Lange (2015). First, a recast of the convex clustering optimization in (13) is introduced by (25), where $\mathcal{E} = \{l = (i_1, i_2) : 1 \leq i_1 < i_2 \leq n\}$.

$$\begin{aligned} &\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|_2^2 + \gamma \sum_{l \in \mathcal{E}} w_l \|\mathbf{v}\| \\ &s.t.\ \mathbf{v}_l = \mathbf{a}_{i_1} - \mathbf{a}_{i_2} \end{aligned} \tag{25}$$

The minimization of the objective in (25) is equivalent to minimizing the Lagrangian function in (26). Here $\mathbf{V}$ denotes the matrix $(\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathcal{E}|})$, $\mathbf{\Lambda}$ denotes the matrix $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{|\mathcal{E}|})$ and v denotes a small constant.

$$\begin{aligned} \mathcal{L}_v(\mathbf{A}, \mathbf{V}, \mathbf{\Lambda}) = &\frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|_2^2 + \gamma_1 \sum_{l \in \mathcal{E}} w_l \|\mathbf{v}_l\|_q \\ &+ \sum_{l \in \mathcal{E}} \langle \boldsymbol{\lambda}_l, \mathbf{v}_l - \mathbf{a}_{i_1} + \mathbf{a}_{i_2} \rangle + \frac{v}{2} \sum_{l \in \mathcal{E}} \|\mathbf{v}_l - \mathbf{a}_{i_1} + \mathbf{a}_{i_2}\|_2^2 \end{aligned} \tag{26}$$

As the join minimization of the Lagrangian function over $\mathbf{A}$ and $\mathbf{V}$ is often challenging, AMA simplifies the minimization problem by minimizing the Lagrangian one block of variables at a time:

$$\begin{aligned} \mathbf{A}^{m+1} &= \arg\min_{\mathbf{A}} \mathcal{L}_v(\mathbf{A}, \mathbf{V}^m, \mathbf{\Lambda}^m) \\ \mathbf{V}^{m+1} &= \arg\min_{\mathbf{V}} \mathcal{L}_v(\mathbf{A}^{m+1}, \mathbf{V}, \mathbf{\Lambda}^m) \\ \boldsymbol{\lambda}_l^{m+1} &= \boldsymbol{\lambda}_l^m + v(\mathbf{v}_l^{m+1} - \mathbf{a}_{i_1}^{m+1} + \mathbf{a}_{i_2}^{m+1}),\ l \in \mathcal{E} \end{aligned} \tag{27}$$

These blocks of variables need to be updated throughout the process. AMA updates $\mathbf{A}$ by minimizing the ordinary Lagrangian ($v = 0$), resulting in:

$$\mathbf{A}^{m+1} = \arg\min_{\mathbf{A}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|_2^2 + \sum_{l} \langle \boldsymbol{\lambda}_l^m, \mathbf{v}_l - \mathbf{a}_{i_1} + \mathbf{a}_{i_2} \rangle \tag{28}$$

Since this minimization can be separated for each $\mathbf{a}_i$, this minimization can be simplified to: $\mathbf{a}_i^{m+1} = \mathbf{x}_i + \sum_{i_1 = i} \boldsymbol{\lambda}_l^m - \sum_{i_2 = i} \boldsymbol{\lambda}_l^m$, as shown by Chi and Lange (2015). Furthermore, an update of $\mathbf{V}$ is not necessary, as $\mathbf{A}$ is independent of $\mathbf{V}$.

The update of $\mathbf{\Lambda}$ is defined by Chi and Lange (2015) as $\boldsymbol{\lambda}_l^m = \mathcal{P}_{C_l}[\boldsymbol{\lambda}_l^{m-1} - v(\mathbf{a}_{i_1}^m - \mathbf{a}_{i_2}^m)]$, with C_l defined by $C_l = \{\boldsymbol{\lambda}_l : \|\boldsymbol{\lambda}_l\|_\dagger \leq \gamma_1 w_l\}$. They defined $\mathcal{P}_C(\mathbf{z})$ as the projection onto the set $C = \{\mathbf{y} : \|\mathbf{y}\|_\dagger\}$ with respect to the norm $\|.\|_\dagger$. Note that $\|.\|_\dagger$ is the dual norm of $\|.\|_q$, which is referred to as the fusion penalty.

These updates are iterated until convergence to determine a final solution for $\mathbf{A}$ and $\mathbf{\Lambda}$. The convergence of AMA is assured, provided that v remains reasonably small (Chi & Lange, 2015).

Algorithm 3 Implementing convex clustering using AMA

1. Initialize $\mathbf{\Lambda}^0$, $m = 1, 2, \ldots$.
2. For $i = 1, \ldots, n$, do
   $\mathbf{a}_i^m = \mathbf{x}_i + \sum_{i_1=i} \boldsymbol{\lambda}_l^{m-1} - \sum_{i_2=i} \boldsymbol{\lambda}_l^{m-1}$
3. For $l \in \mathcal{E}$, do
   $\boldsymbol{\lambda}_l^m = \mathcal{P}_{C_l}[\boldsymbol{\lambda}_l^{m-1} - v(\mathbf{a}_{i_1}^m - \mathbf{a}_{i_2}^m)]$, where $C_l = \{\boldsymbol{\lambda}_l : \|\boldsymbol{\lambda}_l\|_\dagger \leq \gamma_1 w_l\}$
4. Repeat Steps 2-3 until convergence.

A.4 S-AMA

In this part of the Appendix the efficient optimization approach S-AMA (B. Wang et al., 2018) is discussed, using a similar computational strategy as Chi and Lange (2015) for AMA. For the implementation of S-AMA, the objective function in (18) is rewritten to (29).

$$\begin{aligned} &\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{2} \sum_{j=1}^{p} \|\mathbf{x}_j - \mathbf{a}_j\|_2^2 + \gamma_1 \sum_{l \in \mathcal{E}} w_l \|\mathbf{v}_l\|_q + \gamma_2 \sum_{j=1}^{p} u_j \|\mathbf{a}_j\|_2 \\ &s.t.\ \mathbf{v}_l = \mathbf{a}_{i_1} - \mathbf{a}_{i_2} \end{aligned} \tag{29}$$

Equivalently, the Lagrangian function (30) can be minimized. Using $\mathbf{V}$ to denote the matrix $(\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathcal{E}|})$, $\mathbf{\Lambda}$ to denote the matrix $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{|\mathcal{E}|})$ and v as a small constant.

$$\begin{aligned} \mathcal{L}_v(\mathbf{A}, \mathbf{V}, \mathbf{\Lambda}) = &\frac{1}{2} \sum_{j=1}^{p} \|\mathbf{x}_j - \mathbf{a}_j\|_2^2 + \gamma_1 \sum_{l \in \mathcal{E}} w_l \|\mathbf{v}_l\|_q + \gamma_2 \sum_{j=1}^{p} u_j \|\mathbf{a}_j\|_2 \\ &+ \sum_{l \in \mathcal{E}} \langle \boldsymbol{\lambda}_l, \mathbf{v}_l - \mathbf{a}_{i_1} + \mathbf{a}_{i_2} \rangle + \frac{v}{2} \sum_{l \in \mathcal{E}} \|\mathbf{v}_l - \mathbf{a}_{i_1} + \mathbf{a}_{i_2}\|_2^2 \end{aligned} \tag{30}$$

Simultaneously handling feature-level and observation-level vectors in the new objective function presents additional challenges compared to the original algorithms introduced by Chi and Lange (2015). Still, splitting the minimization of the augmented Lagrangian problem by alternatively addressing one block of variables at a time works well:

$$\begin{aligned} \mathbf{A}^{m+1} &= \arg\min_{\mathbf{A}} \mathcal{L}_v(\mathbf{A}, \mathbf{V}^m, \mathbf{\Lambda}^m) \\ \mathbf{V}^{m+1} &= \arg\min_{\mathbf{V}} \mathcal{L}_v(\mathbf{A}^{m+1}, \mathbf{V}, \mathbf{\Lambda}^m) \\ \boldsymbol{\lambda}_l^{m+1} &= \boldsymbol{\lambda}_l^m + v(\mathbf{v}_l^{m+1} - \mathbf{a}_{i_1}^{m+1} + \mathbf{a}_{i_2}^{m+1}),\ l \in \mathcal{E} \end{aligned} \tag{31}$$

Next, we discuss the detailed updating implementations for $\mathbf{A}$, $\mathbf{V}$ and $\mathbf{\Lambda}$, summarized in Algorithm 4 (B. Wang et al., 2018). S-AMA solves $\mathbf{A}$ by treating $v = 0$, resulting in the change of $\mathcal{L}_v$ to $\mathcal{L}_0$ in (31). Updating $\mathbf{A}$ requires to solve the p group-lasso problems as denoted in (32) (B. Wang et al., 2018).

$$\min_{\mathbf{a}_j} \frac{1}{2} \|\mathbf{x}_j - \mathbf{a}_j\|_2^2 + \gamma_2 u_j \|\mathbf{a}_j\|_2 \quad \forall j = 1, \ldots, p. \tag{32}$$

The group lasso problem can be solved using the Karush-Kuhn-Tucker (KKT) conditions (Yuan & Lin, 2006), leading to the closed-form solution for (32): $\hat{\mathbf{a}}_j = \left(1 - \frac{\gamma_2 u_j}{\|\mathbf{z}_j\|_2}\right)_+ \mathbf{z}_j$, where

$\mathbf{z}_j = \mathbf{x}_j + \sum_{l\in\mathcal{E}} \lambda_{jl}(\mathbf{e}_{i_1} - \mathbf{e}_{i_1})^2$ and $(z)_+ = \max\{0, z\}$. The solution $\hat{\mathbf{a}}_j$ are still centered for every j. A detailed derivation of the closed form solution is provided by B. Wang et al. (2018). Note that the closed form solution of $\hat{\mathbf{A}}$ is independent of $\mathbf{V}$, indicating that an update of $\mathbf{V}$ is not necessary.

In order to update $\mathbf{\Lambda}$, $\mathcal{P}_C(\mathbf{z})$ is defined as the projection onto $C = \{\mathbf{y} : \|\mathbf{y}\|_\dagger\}$ with respect to the norm $\|.\|_\dagger$, where $\|.\|_\dagger$ is the dual norm of $\|.\|_q$, defining the fusion penalty. B. Wang et al. (2018) shows that the update of $\mathbf{\Lambda}$ can be reduced to $\boldsymbol{\lambda}_l^m = \mathcal{P}_{C_l}[\boldsymbol{\lambda}_l^{m-1} - v(\mathbf{a}_{i_1}^m - \mathbf{a}_{i_2}^m)]$ using $C_l = \{\boldsymbol{\lambda}_l : \|\boldsymbol{\lambda}_l\|_\dagger \leq \gamma_1 w_l\}$.

To achieve a definitive solution for $\mathbf{A}$ and $\mathbf{\Lambda}$, the updates must be iterated until convergence. B. Wang et al. (2018) investigated this aspect, demonstrating that the convergence of S-AMA is guaranteed as long as the positive constant v remains within reasonable bounds.

Algorithm 4 Implementing sparse convex clustering using S-AMA

1. Initialize $\mathbf{\Lambda}^0$, $m = 1, 2, \ldots$.
2. For $j = 1, \ldots, p$, do
   $\mathbf{z}_j^m = \mathbf{x}_j + \sum_{l\in\mathcal{E}} \lambda_{lj}^{m-1}(\mathbf{e}_{i_1} - \mathbf{e}_{i_2})$
   $\mathbf{a}_j^m = \left(1 - \frac{\gamma_2 u_i}{\|\mathbf{z}_i^m\|_2}\right)_+ \mathbf{z}_j^m$
   $\mathbf{a}_j^m = \mathbf{a}_j^m - \bar{\mathbf{a}}_j^m \mathbf{1}_n$, where $\bar{\mathbf{a}}_j^m = \mathbf{1}_n^T \mathbf{a}_j^m / n$
3. For $l \in \mathcal{E}$, do
   $\boldsymbol{\lambda}_l^m = \mathcal{P}_{C_l}[\boldsymbol{\lambda}_l^{m-1} - v(\mathbf{a}_{i_1}^m - \mathbf{a}_{i_2}^m)]$, where $C_l = \{\boldsymbol{\lambda}_l : \|\boldsymbol{\lambda}_l\|_\dagger \leq \gamma_1 w_l\}$
4. Repeat Steps 2-3 until convergence.

B Performance Metrics for K-means Clustering Methods

In this Appendix several performance metrics are shown for traditional k-means clustering, sparse k-means clustering and k-means clustering enhanced with either the Laplacian score or mRMR in a two step approach. The performance metrics are denoted for the four data settings as well as the three noise levels: low ($\sigma = 1$), medium ($\sigma = 1.5$) and high ($\sigma = 2$). The data settings are summarized as: setting 1: $K = 2$, $n = 60$, $p = 150$ and $\mu = 0.6$, setting 2: $K = 2$, $n = 60$, $p = 500$ and $\mu = 0.7$, setting 3: $K = 4$, $n = 60$, $p = 150$ and $\mu = 0.9$ and setting 4: $K = 4$, $n = 60$, $p = 500$ and $\mu = 1.2$.

Table 2 provides the average ARI for each method when running 200 trials over the different simulation sets and noise levels. The ARI should be high for good clustering. The highest ARI is shown in bold. The standard deviation of the ARI is also provided, which should be low. The lowest standard deviation is made bold. For both the mean and standard deviation, the second best value is denoted in underscore.

[2]The vectors $\mathbf{e}_{i_1}$ and $\mathbf{e}_{i_2}$ are n-dimensional unit vectors: all components are zero, expect the i_1 and i_2 component, respectively, equaling 1.

Table 2: Average ARI of k-means, Laplacian k-means, mRMR k-means and sparse k-means for the multiple simulation settings and noise levels.

		Low Noise		Medium Noise		High Noise	
	Methods	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**
Simulation 1	**K-means**	0.951	0.055	0.475	0.266	0.137	0.154
	Laplacian k-means	0.992	0.024	0.855	0.093	0.619	0.144
	mRMR k-means	**0.995**	**0.017**	**0.878**	**0.082**	**0.649**	**0.130**
	Sparse k-means	0.961	0.053	0.432	0.257	0.092	0.141
Simulation 2	**K-means**	0.942	0.096	0.251	0.235	0.065	0.110
	Laplacian k-means	**0.999**	0.009	0.947	0.058	0.769	0.129
	mRMR k-means	**0.999**	**0.007**	**0.953**	**0.056**	**0.782**	0.112
	Sparse k-means	0.985	0.033	0.238	0.254	0.053	**0.104**
Simulation 3	**K-means**	0.845	0.156	0.331	0.119	0.128	0.074
	Laplacian k-means	0.993	0.018	0.785	0.097	0.449	0.104
	mRMR k-means	**0.997**	**0.012**	**0.831**	**0.087**	**0.485**	0.104
	Sparse k-means	0.906	0.129	0.279	0.128	0.095	**0.070**
Simulation 4	**K-means**	0.874	0.178	0.387	0.121	0.135	0.072
	Laplacian k-means	**1.000**	0.005	0.964	0.037	0.787	0.094
	mRMR k-means	**1.000**	**0.000**	**0.976**	**0.031**	**0.810**	0.094
	Sparse k-means	0.850	0.236	0.355	0.234	0.072	**0.071**

Table 3 provides the average FNR for each method when running 200 trials over the different simulation sets and noise levels. The FNR should be low for good clustering. The lowest FNR is shown in bold. Additionally, the second best FNR is underlined. The standard deviation of the FNR is also provided, which should be low. The lowest standard deviation is made bold, while the second best is underlined.

Table 3: Average FNR of k-means, Laplacian k-means, mRMR k-means and sparse k-means for the multiple simulation settings and noise levels.

		Low Noise		Medium Noise		High Noise	
	Methods	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**
Simulation 1	**K-means**	0.024	0.027	0.248	0.125	0.408	0.075
	Laplacian k-means	0.004	0.011	0.069	0.044	0.181	0.068
	mRMR k-means	**0.002**	**0.008**	**0.058**	**0.039**	**0.168**	**0.063**
	Sparse k-means	0.019	0.025	0.271	0.123	0.434	0.070
Simulation 2	**K-means**	0.028	0.044	0.354	0.111	0.443	0.058
	Laplacian k-means	0.001	0.005	0.025	0.028	0.110	0.061
	mRMR k-means	**0.000**	**0.003**	**0.023**	**0.027**	**0.104**	**0.053**
	Sparse k-means	0.007	0.016	0.364	0.123	0.452	0.054
Simulation 3	**K-means**	0.091	0.084	0.453	0.085	0.600	0.058
	Laplacian k-means	0.005	0.012	0.146	0.066	0.374	0.073
	mRMR k-means	**0.002**	**0.008**	**0.116**	**0.059**	**0.350**	0.071
	Sparse k-means	0.064	0.088	0.494	0.094	0.629	**0.052**
Simulation 4	**K-means**	0.063	0.087	0.412	0.089	0.597	**0.054**
	Laplacian k-means	**0.000**	0.003	0.025	0.026	0.146	0.064
	mRMR k-means	**0.000**	**0.000**	**0.016**	**0.021**	**0.129**	0.064
	Sparse k-means	0.094	0.152	0.435	0.167	0.645	**0.054**

Table 4 provides the average FPR for each method when running 200 trials over the different simulation sets and noise levels. The FPR should be low for good clustering. Additionally, the standard deviation of the FPR is provided, which should also be low. The lowest FPR and standard deviation are shown in bold, while the second best values are underlined.

Table 4: Average FPR of k-means, Laplacian k-means, mRMR k-means and sparse k-means for the multiple simulation settings and noise levels.

		Low Noise		Medium Noise		High Noise	
	Methods	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**
Simulation 1	**K-means**	0.025	0.028	0.269	0.140	0.441	0.083
	Laplacian k-means	0.004	0.012	0.073	0.047	0.193	0.074
	mRMR k-means	**0.002**	**0.009**	**0.061**	**0.042**	**0.177**	**0.065**
	Sparse k-means	0.019	0.027	0.287	0.130	0.459	0.072
Simulation 2	**K-means**	0.029	0.050	0.383	0.124	0.476	0.063
	Laplacian k-means	0.001	0.005	0.026	0.029	0.117	0.066
	mRMR k-means	**0.000**	**0.003**	**0.023**	**0.028**	**0.111**	0.058
	Sparse k-means	0.007	0.016	0.386	0.129	0.480	**0.055**
Simulation 3	**K-means**	0.046	0.055	0.175	0.034	0.226	0.024
	Laplacian k-means	0.002	0.004	0.054	0.025	0.142	0.029
	mRMR k-means	**0.001**	**0.003**	**0.042**	**0.022**	**0.132**	0.029
	Sparse k-means	0.024	0.033	0.185	0.033	0.230	**0.021**
Simulation 4	**K-means**	0.044	0.066	0.162	0.035	0.222	0.024
	Laplacian k-means	**0.000**	0.001	0.009	0.009	0.053	0.024
	mRMR k-means	**0.000**	**0.000**	**0.006**	**0.008**	**0.047**	0.024
	Sparse k-means	0.043	0.070	0.170	0.062	0.236	**0.022**