University of California

Los Angeles

Solution Path Clustering with Minimax Concave Penalty and Its Applications to Noisy Big Data

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Yuliya Marchetti

2014

Abstract of the Dissertation

Solution Path Clustering with Minimax Concave Penalty and Its Applications to Noisy Big Data

by

Yuliya Marchetti

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2014

Professor Qing Zhou, Chair

Fast accumulation of large amounts of complex data has created a need for more sophisticated statistical methodologies to discover interesting patterns and better extract information from these data. The large scale of the data often results in challenging high-dimensional estimation problems where only a minority of the data shows specific grouping patterns. To address these emerging challenges, we develop a new clustering methodology that introduces the idea of a regularization path into unsupervised learning. A regularization path for a clustering problem is created by varying the degree of sparsity constraint that is imposed on the differences between objects via the minimax concave penalty with adaptive tuning parameters. Instead of providing a single solution represented by a cluster assignment for each object, the method produces a short sequence of solutions that determines not only the cluster assignment but also a corresponding number of clusters for each solution. The optimization of the penalized loss function is carried out through an MM algorithm with block coordinate descent. The advantages of this clustering algorithm compared to other existing methods are as follows: it does not require the input of the number of clusters; it is capable of simultaneously separating irrelevant or noisy observations that show no grouping pattern, which can greatly improve data interpretation; and it is a general methodology that can be applied to many clustering problems. We then develop an iterative subsampling approach to improve the computational efficiency of this clustering methodology. The proposed approach iterates between clustering a small subsample of the full data and sequentially assigning the other data points to attain orders of magnitude of computational savings. It preserves the ability to isolate noise, includes a solution selection mechanism that ultimately provides one clustering solution with an estimated number of clusters, and is shown to be able to extract small tight clusters from noisy data. The iterative subsampling approach's relatively minor losses in accuracy are demonstrated through simulation studies, and its ability to handle large datasets is illustrated through applications to gene expression datasets.

The dissertation of Yuliya Marchetti is approved.

Stefan Horvath

Jan de Leeuw

Ying Nian Wu

Qing Zhou, Committee Chair

University of California, Los Angeles

2014

To James

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Qing Zhou, for his support and guidance, for his patience and kindness. He transformed my graduate research experience to one of joy of learning and treated me without judgement and with respect. He shaped me as a researcher, as a scientist, and a thinking human being, and I am and will always be grateful that he shared his knowledge and wisdom with me.

I would also like to thank my committee members Jan de Leeuw, Ying Nian Wu and Steve Horvath for their guidance and helpful thought-provoking insights. Special appreciation goes to Amy Braverman, because without her encouragement and optimism I would not have entered the doctoral program. And another thank you to Jan and Rick Paik Schoenberg for backing me with the recommendation letters and a trust that I could be a successful doctoral student in the department.

I am also grateful to Glenda Jones, as she was always there to help. My heartfelt thanks go to my dear friends and department mates, Irina Kukuyeva, Linda Zanontian, Rakhee Patel, Jean Wang, Denise Ferrari and Hai Nguyen for their unconditional friendship, sharing and generosity to someone who was so far behind them and who would not have made it without their encouragement. I owe my intellectual well-being and sanity to Mi Kyung Kim, who is wise, intelligent and patient. I would also like to extend my thanks to Karen Chen and Chuck Hall for being friends that I can rely on day and night.

Lastly and most importantly, I would like to express immense appreciation and love to my husband, James. Without him, I simply would not be here. I owe him everything and more.

The research in this dissertation has been made possible by the generous financial support from the National Science Foundation Graduate Research Fellowship and UCLA Dissertation Year Fellowship programs. A version of Chapter 2 of this

dissertation has been published in the Electronic Journal of Statistics with Qing Zhou, and a version of Chapter 3 has been submitted for review in the Statistics and Its Interface journal, also co-authored with Qing Zhou.

Vita

1999	B.A./M.A. (Linguistics), Moscow State Linguistic University, Moscow, Russia, Department of Translation Theory, History, and Criticism.
1999–2007	Corporate Finance Professional, Corning Intellisense Corp, Chipwrights Design, Inc., Beckman Coulter, Inc., Helio LLC. Various roles in accounting and finance in the technology industry.
2004	M.B.A. (Finance), Babson College, Olin Graduate School of Business, Wellesley, MA.
2009	M.S. (Statistics), UCLA, Los Angeles, CA.
2009–2013	National Science Foundation Graduate Research Fellowship.
2009–2013	Graduate Student Researcher, Joint Institute for Regional Earth System Science and Engineering, UCLA, at Jet Propulsion Laboratory, Pasadena, CA.
2010	Graduate Student Researcher, Statistics Department, UCLA.
2012	C.Phil., Statistics Department, UCLA.
2013-2014	Dissertation Year Fellowship, UCLA.

Publications

Lambrigtsen B., Braverman A., Brown S., Ngyuen H., Yaglovskaya Y. Measuring the vertical structure of tropical convection with microwave sounders (2008) Poster presentation in: *American Geophysical Union, Fall Meeting.* Poster abstract CL#08-3211.

Esfandiari M., Nguyen H., Yaglovskaya Y., Gould R. Enhancing statistical literacy through short open-ended questions that involve context, data, and upper level thinking (2010). Contributed publication and presentation in: *ICOTS-8 Conference Proceedings.*

Marchetti Y., Zhou Q. Solution path clustering with adaptive concave penalty (2014). *Electronic Journal of Statistics*, 8(1), 1569–1603.

CHAPTER 1

Introduction

1.1 Overview and challenges of clustering

Cluster analysis allows us to group a collection of objects into subsets such that objects within a subset are similar to each other, while objects in different subsets are dissimilar from each other. Clustering is widely used in exploratory data analysis and has a great variety of applications ranging from biology and astrophysics to social sciences and psychology. Clustering is a first step in knowledge discovery where prior information is rarely available to a researcher. It is also instrumental for visualization of complex data and for partitioning a dataset into more homogeneous groups in which simpler models might be adequate. At the same time, clustering can enable discovery of unknown groups or associations, providing deeper insights into the data. For example, in biological research, clustering can help determine which genes are associated with particular cellular functions or phenotypes and can help isolate subclasses of diseases for targeted treatments.

There exists a great variety of clustering methods. A majority of them rely on minimizing some loss function, usually by an iterative procedure, like the well known k-means algorithm, while other methods recursively organize objects into trees, like the popular hierarchical clustering. Spectral clustering techniques are based on graph theory and matrix decomposition and are gaining popularity as being simple, accurate, and able to find non-convex clusters [Lux07, HTF09]. Along with k-means and its variants, there also exist mixture likelihood clustering approaches [MBP02, FR02, YFM01] that assume an underlying statistical model for the data and maximize the likelihood function with the EM algorithm or MCMC methods [BCR97, OR07]. Yet other methods take modified or combined approaches, for example, self-organizing maps [Koh90], dp-means [KM11], CLICK [SMS03], gene shaving [HTE00], pclust [WNM08], and support vector clustering [BHS01] to name just a few. Clustering literature is vast, and surveys of clustering methodologies are usually specialized. A recent comprehensive review of basic and new clustering methodologies is presented in [AR13], and more general developments and trends in clustering are covered in [Jai10].

Increasingly large and complex datasets, such as those in gene expression analysis or data mining, have created the need for new efficient approaches to clustering. Data now often contain large amounts of both noisy observations and irrelevant variables. Most existing clustering methods do not address the problems of identifying noise and selecting meaningful variables. Only recently, researchers have shown that simultaneously accommodating for the presence of noisy or irrelevant observations can immensely improve clustering results and provide a better interpretation of the patterns in the data [TW05b, TMZ06]. A number of new clustering algorithms have been proposed, the most popular of which include the resampling-based tight clustering method [TW05b], penalized weighted k-means [Tse07], and model-based clustering [FR02]. Other methods that take noise into account include adap_Cluster [DMM02], k-clips [MR09], DWCN [SSL10], trimmed k-means [GGM08] and other robust clustering algorithms [SEC13, FKG12]. Another challenging task in clustering in general is the specification of the number of clusters, which is required as an input for most of the existing methods. When the number of clusters is a required input, the obtained solution is prone to error, especially when a dataset is large and complex. There are a number of methods that suggest rules for choosing the number of clusters. For a recent overview, please refer to [FW12].

A different class of methodologies that has gained popularity for high-dimensional complex datasets is sparsity regularization techniques. Some examples of such penalization methods include the lasso [Tib96], the elastic net [ZH05], the group lasso [YL06], the fused lasso [TSR05], SCAD [FL01], SparseNet [MFH11], and MC+ [Zha10]. These methods are mostly used in linear and generalized linear models for identifying useful predictors among a large number of covariates. They introduce a penalty to a loss function to find sparse solutions in challenging problems. Efficient optimization methods exist that compute the entire regularization path or a particular solution for a penalized loss function, such as the least angle regression [EHJ04] and coordinate descent [FHH07, FHT10, WL08].

Penalized estimation has also been increasingly implemented for clustering problems. It has been utilized in clustering mostly for variable selection, such as in gene expression analysis [PS07, WZ08, XPS08, ZPS09, GLM10, WT10, SW12]. These methods assume a given number of clusters and select useful variables to partition objects, usually in high-dimensional cases and when the number of samples is smaller than the number of variables or dimensions. Very recently, several authors introduced penalized clustering methods that impose a penalty on the pairwise differences between cluster centers and make it possible to generate sets of solutions that do not require the specification of the number of clusters [PDS05, HJB11, LOL11, CL13, PSL13]. The partitioning of the data points into a certain number of clusters for such methods is achieved through varying the degree of penalization. Such emerging approaches are very promising for clustering and have potential to handle datasets with complex, intricate structures. Independently from the mentioned penalization methods, we develop a new regularization based approach to clustering and introduce the idea of a solution path.

1.2 Clustering and large data

There have been a large number of sophisticated and state-of-the-art clustering methods developed in statistics and machine learning over the last few decades, in response to the rapid emergence of more complex and richer datasets. The exponential growth in stored data has ushered in the era of “big data” and the interest to understand and exploit such information. The ever-increasing problem size and computational intensity of such efforts are further creating challenges for conventional data analysis methods. Large datasets are often noisy and consist of heterogeneous subsets. Consequently, clustering is an essential exploratory analysis to partition the data into smaller subsets and to filter out noisy data points so that simpler and robust models may be constructed for each cluster.

The challenges of clustering large datasets have been widely studied, and numerous methods have been proposed to efficiently handle such data. The majority of such methods, originating in machine learning and data mining applications, modify the popular k-means or EM algorithms in order to increase their computational speeds. Some examples of such modifications include the use of simple random subsampling or more sophisticated sampling schemes [GRS01, RD03, HJ06, BFR98, FRB98], the computation of some summary representations of the data [ZRL96, NH02, Pos01], and the parallelizing and distributing of the computation process [RRP07, EIM11]. A number of reviews of such techniques summarize and categorize the abundance of these methodologies [AR13, Jai10, SAW14].

More advanced clustering methodologies, on the other hand, make it possible, for example, to automatically estimate the number of clusters [FR02, Tse07], to isolate outliers and noisy data points [FR02, TW05b, Tse07], i.e. data points showing no grouping pattern, to perform dimension and variable selection [PS07, ZPS09], and to handle non-convex clusters [HJB11, CL13, PSL13]. Although not as fast as the k-means or the EM algorithm, these methods are powerful, and it is very useful to develop computational strategies to enable the application of these clustering methodologies to moderately large datasets that otherwise would be prohibitively slow or simply impossible to manipulate.

Perhaps the simplest approach to clustering a large dataset is based on random subsampling, which would be computationally efficient if the subsample is very small, say with size on the order of $\sqrt{n}$, where n is the sample size of a dataset. Subsampling for large datasets is not new and was first proposed in [KR90]. Banfield and Raftery [BR93] cluster a real dataset using model-based clustering (mclust) and apply discriminant analysis to classify the remaining data points. Fayyad and Smyth [FS96] suggest iteratively subsampling datasets for clustering and classifying the remaining data points until all of them belong to the clusters identified in the subsamples with sufficiently high probability. Later, Fraley and Raftery [FR02] elaborate on subsample clustering and discriminant analysis for large data and discuss a modification of the simple random subsampling with the goal of finding small, tight clusters. A number of other clustering methods were subsequently developed, following a similar idea [Mai01, WBF04, FRW05, KMC10, NDR14]. All of these methods are geared mainly towards computational efficiency, and several were also developed to find small clusters in large datasets [FS96, Mai01, FRW05, NDR14]. In this work we propose an approach to cluster big noisy data using a simple subsampling approach, performed iteratively on a dataset and coupled with a new noise classification and cluster selection procedures, which achieves all the aforementioned properties and benefits.

1.3 Outline of the dissertation

The remaining part of this dissertation is organized in three chapters. In Chapter 2 we discuss more extensively the regularization methods for clustering. We formulate, derive, and implement a new algorithm based on a quadratic loss function with a penalty imposed on the pairwise distances between the cluster centers. We show the performance of this method on simulated data and real data and then compare it with other popular sophisticated clustering methodologies that are able to recognize noisy data points. Chapter 3 is centered on extending the method developed in Chapter 2 to noisy big data. A heuristic acceleration of the algorithm is introduced that utilizes iterative subsampling with subsequent cluster assignment of the remaining data based on likelihood ratio evaluations. We are able to detect and isolate the noisy data points and provide the estimated number of clusters automatically with minimal user input. Lastly, we show the performance of the new iterative subsampling procedure on simulated data and large gene expression data, where we find some biologically meaningful gene clusters. The dissertation is concluded in Chapter 4 with a summary and future work discussion.

CHAPTER 2

Clustering with Concave Penalty

2.1 Introduction

In this chapter we propose a novel solution path clustering (SPC) method that can be applied to a wide range of data settings, including high dimensionality and the presence of noisy or irrelevant observations, which this method is able to isolate into singleton or very small clusters. Our algorithm minimizes a penalized quadratic loss function under the minimax concave penalty (MCP) [Zha10]. The regularization allows us to obtain sparse solutions and to construct a solution path with a decreasing number of clusters, which eliminates the need to specify the number of clusters as an input parameter. The method minimizes a non-convex objective function via the majorization-minimization (MM) algorithm [Lan04] coupled with block coordinate descent. We also develop adaptive data-driven strategies for selecting the penalty parameters along a solution path so that the SPC algorithm has in effect only one tuning parameter for initializing the path. Overall, SPC is a simple, easily implemented and relatively fast algorithm that has worked well in practice, although its convergence properties are to be established in future work.

Very recently, a number of authors introduced penalized clustering methods that are similar to our method in that they also impose a penalty on the pairwise differences between cluster centers and can generate solution paths that do not require the specification of the number of clusters. [PDS05], [HJB11], [LOL11] and [CL13] suggested minimizing an objective function where the penalty on the differences between the cluster centers is convex, and thus, the resulting algorithms are guaranteed to converge to a global minimizer. The potential severe bias in the cluster center estimates from these procedures are handled through penalty weights. The first three referenced papers focus primarily on the optimization of objective functions with convex penalties and do not discuss the choice of the penalty parameter, the detection of noise, performance on high-dimensional data, nor the impact of the penalty weights on the clustering results. [CL13] have improved on these convex clustering methods and have provided a general unified algorithm for solving such problems. They have also noted that the solution paths obtained from convex clustering can be unsatisfactory if the weights are not selected properly.

In addition, [PSL13] have proposed a penalized regression-based clustering method (PRclust) using a novel non-convex penalty on the pairwise differences in order to alleviate the possible bias of convex penalties. The authors have re-parametrized the objective function to ensure the convergence of the coordinate descent algorithm to a stationary point. PRclust, however, has not been shown to handle noisy and large high-dimensional datasets. In contrast to our adaptive selection of penalty parameters, [PSL13] mainly focus on determining the number of clusters by searching over a pre-specified grid of three penalty parameters resulting from the re-parametrization, which might not be efficient for large complex datasets.

The remainder of this chapter is organized as follows. Section 2.2 provides a general formulation of a clustering problem under a concave penalty. In Section 2.3 we develop our clustering algorithm with adaptive solution path construction. In Section 2.4 we use simulated data to illustrate and compare the SPC algorithm to several clustering methods. In Section 2.5 we briefly discuss solution selection for SPC. In Section 2.6 we apply SPC to a gene expression dataset from mouse embryonic stem cells to show the performance on bigger real data. Finally, Section 2.7 contains further discussion and future research directions.

2.2 Formulation

Let $Y = (y_{im})_{n\times p}$ be an observed data matrix, where $y_i = (y_{i1}, \ldots, y_{ip}) \in \mathbb{R}^p$ represents the ith object. Assuming that the underlying model for y_i, $i = 1, \ldots, n$, is multivariate Gaussian with a mean parameter $\theta_i \in \mathbb{R}^p$ and a constant diagonal covariance matrix $\sigma^2 I_p$, we propose to cluster the n objects into an unknown number of clusters K by minimizing a penalized ℓ_2 loss function. This is achieved by the use of sparsity regularization on the difference between pairwise mean parameters $d(\theta_i, \theta_j) = \|\theta_i - \theta_j\|_2$. Our goal is then to minimize

$$\ell(\theta) = \sum_{i=1}^{n} \|y_i - \theta_i\|_2^2 + \lambda \sum_{i<j} \rho\left(\|\theta_i - \theta_j\|_2\right), \tag{2.1}$$

over $\theta = (\theta_1, \ldots, \theta_n)$, where $\lambda > 0$ and $\rho(\cdot)$ is some penalty function. With a careful choice of $\rho(\cdot)$ we can achieve sparsity such that $\|\hat{\theta}_i - \hat{\theta}_j\|_2$ is arbitrarily small when λ is sufficiently large, where $(\hat{\theta}_1, \ldots, \hat{\theta}_n)$ is the minimizer of (2.1). An important advantage of this formulation, especially for situations when there is very little prior knowledge about the data, is that the number of clusters K does not need to be specified beforehand. This regularization also makes it possible to naturally separate noisy objects into singletons and to prevent them from erroneously merging into other clusters.

An appropriately chosen penalty should result in an estimator that satisfies the properties of unbiasedness, sparsity, and continuity [FL01]. To achieve these three properties and to specifically avoid excessive bias in the estimation of θ, which could lead to unsatisfactory cluster assignment, we propose to use the minimax concave penalty (MCP) developed by [Zha10],

$$\begin{aligned}\rho\left(t\right) &= \int_0^t \left(1-\frac{x}{\delta\lambda}\right)_+ dx \\ &= \left(t-\frac{t^2}{2\lambda\delta}\right) I(t<\lambda\delta) + \left(\frac{\lambda\delta}{2}\right) I(t\geq\lambda\delta), \quad (t\geq 0),\end{aligned} \tag{2.2}$$

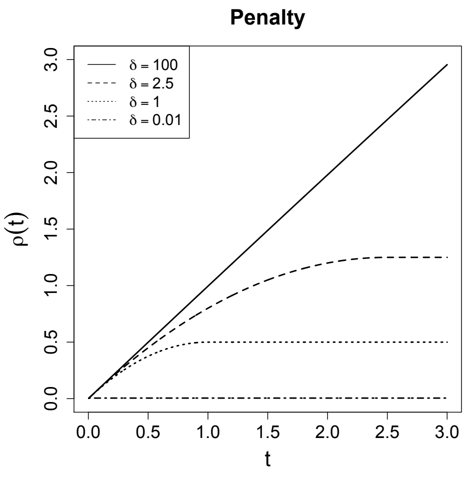
where $I(\cdot)$ is the indicator function. The MCP penalty $\rho(t)$ in (2.2) defines a family of penalty functions that are concave in $t\in[0,\infty)$, where $\lambda>0$ controls the amount of regularization and $\delta>0$ controls the degree of concavity. It has been noted that such non-convex penalties promote sparser models than the ℓ_1 penalty with the same or superior prediction accuracy in regression models [Zha10, MFH11]. In fact, MCP includes both the ℓ_1 penalty when $\delta\to\infty$ and the ℓ_0 penalty when $\delta\to 0+$, forming a continuum between the two extremes. MCP is a simple differentiable penalty function with only two parameters and is designed to minimize maximum concavity. Compared to other non-convex penalties such as SCAD [FL01] or the truncated Lasso penalty [PSL13], it includes the explicit concavity parameter δ in its formulation that is easily separated from the penalization rate. Increasing the concavity through this parameter allows us to effectively control the bias of θ using a data-driven approach. The minimax concave penalty is demonstrated in Figure 2.1.

We illustrate this regularization in the penalized loss function (2.1) with a special case $n=2$. Denote the sample mean of the two observations by $\bar{y}=\frac{1}{2}(y_1+y_2)$. The objective function in this special case is

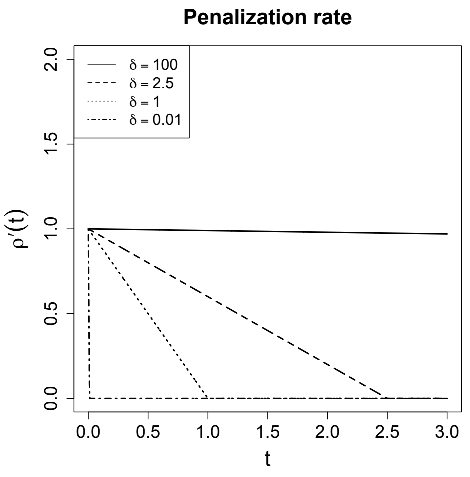
$$\ell(\theta_1,\theta_2)=\|y_1-\theta_1\|_2^2+\|y_2-\theta_2\|_2^2+\lambda\rho\left(\|\theta_1-\theta_2\|_2\right). \tag{2.3}$$

Let $\gamma=(\theta_2-\theta_1)\in\mathbb{R}^p$. Then, for any fixed γ, $\ell(\theta_1,\theta_2)$ is minimized at $(\theta_1,\theta_2)=(\bar{y}-\gamma/2,\bar{y}+\gamma/2)$, and thus, minimizing (2.3) reduces to

$$\begin{aligned}\min_{\gamma}\Big[\ell(\gamma) &= \|y_1-\bar{y}+\gamma/2\|_2^2+\|y_2-\bar{y}-\gamma/2\|_2^2+\lambda\rho\left(\|\gamma\|_2\right) \\ &= \frac{1}{2}\|\gamma-(y_2-y_1)\|_2^2+\lambda\rho\left(\|\gamma\|_2\right)\Big].\end{aligned} \tag{2.4}$$



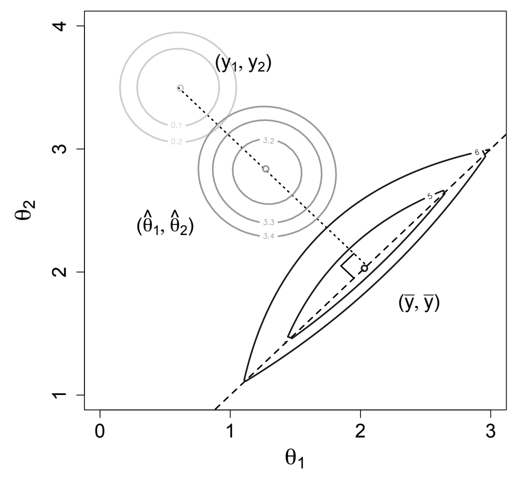
(a)



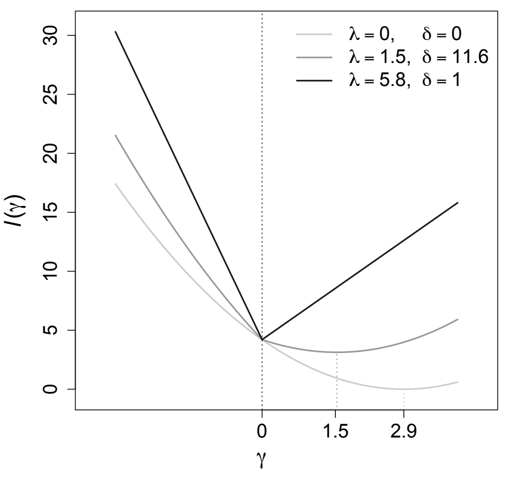
(b)

Figure 2.1: MCP in (a) and its derivative in (b) are plotted for different values of δ and $\lambda = 1$. It approaches the ℓ_1 penalty when δ is high (solid line).

Figure 2.2 plots $\ell(\theta_1, \theta_2)$ and the corresponding $\ell(\gamma)$ for different combinations of (λ, δ) for the MCP in (2.2), and demonstrates how the choice of these two tuning parameters affects the estimation of the cluster centers. The light gray contours in Figure 2.2a represent the unpenalized ℓ_2 loss (with $\delta \to 0$ or $\lambda \to 0$), which is minimized at (y_1, y_2) and gives two different clusters. The black contours in the same figure depict the objective function (2.3) when the value of $\delta = 1$ is low enough and the value of $\lambda = 5.8$ is big enough to produce an unbiased estimate of one cluster center $\hat{\theta}_1 = \hat{\theta}_2 = \bar{y}$. The penalty on $|\theta_1 - \theta_2|$ forces the minimizer to move from (y_1, y_2), when the loss function has no penalty, to $\hat{\theta}_1 = \hat{\theta}_2 = \bar{y}$, which lies on the dashed line $\theta_1 = \theta_2$. The dark gray contours plot the objective function for a larger value of $\delta = 11.6$ and a smaller value of $\lambda = 1.5$, which is minimized at $(\hat{\theta}_1, \hat{\theta}_2) = (1.3, 2.8)$. In this case, two clusters are obtained, and both centers are estimated with substantial bias. Therefore, a proper and data-driven choice of δ and λ is key to our method. This also shows that a penalty in the form $\|\theta_i - \theta_j\|_2$, which corresponds to $\delta \to \infty$, may not be appropriate for clustering.



(a)



(b)

Figure 2.2: Demonstration of the regularization with $n = 2$ and $p = 1$ for three sets of (λ, δ): black color for $(\lambda, \delta) = (5.8, 1)$, dark gray for $(\lambda, \delta) = (1.5, 11.6)$, and light gray for $(\lambda, \delta) = (0, 0)$. (a) The contours of $\ell_2(\theta_1, \theta_2)$ (2.3). The dark gray contours centered at $(\hat{\theta}_1, \hat{\theta}_2) = (1.3, 2.8)$ demonstrate the bias in the estimates of the cluster centers. (b) The penalized loss function $\ell(\gamma)$ (2.4) with minimizer indicated by a vertical dashed line.

Figure 2.2b plots the objective function $\ell(\gamma)$ with the same combinations of tuning parameters as those in Figure 2.2a. It can be seen from the figure that $\ell(\gamma)$ with $(\lambda, \delta) = (5.8, 1)$ is minimized when $\gamma = 0$, i.e. $\theta_1 = \theta_2 = \bar{y}$, and the unpenalized loss function in light gray color is minimized at $\gamma = (y_2 - y_1)$. The loss function with $(\lambda, \delta) = (1.5, 11.6)$ in dark gray color is minimized at $\gamma = 1.5$, between 0 and $(y_2 - y_1)$.

2.3 Solution path clustering

2.3.1 An MM algorithm

Minimization of a non-convex objective function is usually non-trivial. Motivated by the MM algorithm [De 94, Lan04] we propose to majorize the penalty term of (2.1) by a linear function [WL08]. We then minimize the majorizing surrogate function by cyclic block coordinate descent. We initialize the algorithm assuming all objects form singleton clusters and gradually merge the objects into a decreasing number of clusters for an appropriately chosen sequence of parameters (δ, λ), stopping when all the objects form one cluster. Correspondingly, once two objects are merged into a cluster, we do not consider splitting them in the later stages of the algorithm (Remark 3). Suppose that in the current solution the objects y_i are assigned to K clusters with centers $\mu_1, \ldots, \mu_K$, where $\mu_k \in \mathbb{R}^p$. Let $C_k = \{i : \theta_i = \mu_k\}$ represent the kth current cluster and $N_k = |C_k|$ denote the size of this cluster. We may then rewrite the objective function (2.1) as

$$\ell_K(\mu) = \sum_{k=1}^{K} \sum_{i \in C_k} \|y_i - \mu_k\|_2^2 + \lambda \sum_{k<\ell} N_k N_\ell \rho\left(\|\mu_k - \mu_\ell\|_2\right), \qquad (2.5)$$

where $\mu = (\mu_1, \ldots, \mu_K)$. With a proper choice of (λ, δ), minimizing $\ell_K(\mu)$ over μ will force some μ_k's to be very close to one another, effectively merging these clusters into a bigger cluster in the next solution.

To minimize $\ell_K(\mu)$, we employ a blockwise MM step to cycle through μ_k.

At each step, we fix $\mu_{[-k]} = (\mu_1, \ldots, \mu_{k-1}, \mu_{k+1}, \ldots, \mu_K)$ to its current value and majorize

$$\ell_K(\mu_k) \triangleq \sum_{i \in C_k} \|y_i - \mu_k\|_2^2 + \lambda N_k \sum_{\ell \neq k} N_\ell \rho\left(\|\mu_k - \mu_\ell\|_2\right). \tag{2.6}$$

Let $\mu_k^{(t)}$ be the value of μ_k before the current MM step, where t corresponds to the iteration number. By assumption $\mu_k^{(t)} \neq \mu_\ell$ for all ℓ, and we majorize $\rho(\|\mu_k - \mu_\ell\|_2)$ by

$$\begin{aligned}
&\rho\left(\|\mu_k - \mu_\ell\|_2\right) \\
&\leq \rho\left(\|\mu_k^{(t)} - \mu_\ell\|_2\right) + \rho'\left(\|\mu_k^{(t)} - \mu_\ell\|_2\right)\left(\|\mu_k - \mu_\ell\|_2 - \|\mu_k^{(t)} - \mu_\ell\|_2\right) \\
&\leq \rho\left(\|\mu_k^{(t)} - \mu_\ell\|_2\right) + \rho'\left(\|\mu_k^{(t)} - \mu_\ell\|_2\right)\left(\frac{\|\mu_k - \mu_\ell\|_2^2 - \|\mu_k^{(t)} - \mu_\ell\|_2^2}{2\|\mu_k^{(t)} - \mu_\ell\|_2}\right),
\end{aligned} \tag{2.7}$$

due to the concavity of the functions $\rho(x)$ and $\sqrt{x}$ for $x > 0$. When the majorization (2.7) is substituted into $\ell_K(\mu_k)$ (2.6) we obtain a quadratic surrogate function in μ_k for which the minimizer is

$$\mu_k^{(t+1)} = \frac{\bar{y}_k + \lambda \sum_{\ell \neq k} w_{k,\ell}^{(t)} \mu_\ell}{1 + \lambda \sum_{\ell \neq k} w_{k,\ell}^{(t)}}, \tag{2.8}$$

where $\bar{y}_k = \frac{1}{N_k} \sum_{i \in C_k} y_i$ and $w_{k,\ell}^{(t)}$ can be regarded as the weight for μ_ℓ:

$$w_{k,\ell}^{(t)} = \frac{N_\ell \rho'\left(\|\mu_k^{(t)} - \mu_\ell\|_2\right)}{2\|\mu_k^{(t)} - \mu_\ell\|_2} = \frac{N_\ell\left(1 - \|\mu_k^{(t)} - \mu_\ell\|_2 / \lambda\delta\right)_+}{2\|\mu_k^{(t)} - \mu_\ell\|_2}. \tag{2.9}$$

In effect, the derivative ρ' in (2.9) becomes the adaptive weight for the estimation of the cluster centers μ_k, which is similar to the convex clustering penalty weights in [HJB11] and [CL13]. The weight in (2.9), however, uses the distances between the cluster centers μ_k and varies with each iteration, whereas weights in [HJB11] and [CL13] are based on the distances between the data points, which do not change throughout the estimation procedure and solution path. It can be seen from (2.8) and (2.9) that:

- When $\|\mu_k^{(t)} - \mu_\ell\|_2 \geq \lambda\delta$ for all $\ell \neq k$, then $w_{k,\ell}^{(t)} = 0$, and thus, in the next iteration $\mu_k^{(t+1)} = \bar{y}_k$, the sample mean.

- When $\|\mu_k^{(t)} - \mu_\ell\|_2 \ll \lambda\delta$ so that $\lambda w_{k,\ell}^{(t)} \gg 1$ for a particular ℓ, then $\mu_k^{(t+1)} \approx \mu_\ell$ and the two clusters C_k and C_ℓ will merge.

A single iteration of the MM algorithm cycles through all K blocks as summarized in Algorithm 1. To account for data scaling we set $\xi = \frac{\epsilon}{\sqrt{p}} \sum_{m=1}^{p} \sigma_m$, where $\epsilon = 10^{-4}$ and σ_m is the standard deviation of the mth component of the data.

Algorithm 1 One iteration of the MM algorithm

```
1: for $k = 1, \ldots, K$ do
2:     majorization: compute weights $w_{k,\ell}^{(t)}$ as in (2.9) for all $\ell \neq k$
3:     minimization: update $\mu_k^{(t+1)}$ as in (2.8)
4:     if $\|\mu_k^{(t+1)} - \mu_\ell\|_2 < \xi$ for some $\ell$ then
5:         set $\mu_k^{(t+1)}$ and $\mu_\ell$ to their weighted mean
6:     end if
7: end for
```

Remark 1 (Relative sparsity)**.** It should be noted that the minimizer of (2.5) in general does not have exactly identical pairs of μ_k's for any finite λ, and thus only relative sparsity can be achieved so that some μ_k's become very close to one another. This is due to the penalized loss function (2.5) itself, not because of the algorithm implementation. However, such relative sparsity is sufficient for practical applications, with a simple thresholding step like on lines 4-5 of Algorithm 1. We perform the thresholding step for every iteration of the MM algorithm to decrease the overall running time of SPC as K may become smaller for the next iteration.

Remark 2 (Convergence)**.** Convergence for coordinate descent type algorithms has been established for the sum of a smooth function and a non-convex penalty under certain conditions that are in fact met by MCP [MFH11]. However, since the penalty term in (2.6) is non-separable, this result as well as that in [TY09] cannot be applied. Another difficulty is that the objective function (2.6) is non-differentiable when $\mu_k = \mu_\ell$, which does not meet the assumption for the fixed points of the MM algorithm to coincide with the set of the stationary points of the objective function [Lan95, LDY00]. Consequently, there is no theoretical guarantee that Algorithm 1 always converges to a stationary point. In our implementation, Algorithm 1 is repeated until it reaches the stopping criterion

$$\max_{1\leq k\leq K} \|\mu_k^{(t+1)} - \mu_k^{(t)}\|_2 < \xi \tag{2.10}$$

or until 50 iterations. In practice, we have observed that Algorithm 1 almost always terminates in fewer than 50 iterations. See Sections 2.4.2 and 2.4.3 for more discussion with numerical results.

Remark 3 (Cluster splitting)**.** In order to minimize (2.1), it is necessary that objects are allowed to be split or unfused from the clusters to which they were assigned. However, to save computation time, we have made a heuristic assumption that a cluster is never split so that μ_k's are gradually merged into a decreasing number of clusters. In order to exactly solve the problem in (2.1) it is necessary that objects are allowed to be split or unfused from the clusters they were assigned to. The splitting of clusters could be handled easily within the framework of our algorithm by an additional soft thresholding step. The details on cluster splitting by soft thresholding are provided in the Appendix.

2.3.2 Solution path construction

The penalty function in (2.2) has two parameters λ and δ, the former controlling the amount of regularization and the latter determining the degree of concavity of the function. We would like to create some simple data-driven rules for selecting several combinations of the penalty parameters to produce a solution path for any clustering problem. We start the algorithm assuming that each individual observation y_i forms its own singleton cluster. We then gradually enforce sparsity in the differences between cluster centers by using an increasing sequence of λ while reducing the bias, if necessary, through a decreasing sequence of δ. Since the penalty function (2.2) depends on the distances between μ_k's, the sequences of the tuning parameters can be guided by these distances, and the solution from the current combination of (δ, λ) can be used as a warm start for the next combination.

We define a decreasing sequence $\Delta = \{\delta_1, \dots, \delta_H\}$ and for each δ_h, $h = 1, \dots, H$, define an increasing sequence $\Lambda(\delta_h) = \{\lambda_1(\delta_h), \dots, \lambda_G(\delta_h)\}$. The sequence Δ is simply determined by

$$\delta_h = \delta_{h-1}\alpha, \tag{2.11}$$

for $h = 2, \dots, H$, where $\alpha \in (0, 1)$ is a constant. We discuss the choice of δ_1 later in this section when we talk about $\lambda_1(\delta_1)$. Each time the value of δ is decreased a new $\Lambda(\delta)$ is computed. The initial solution is then obtained with the lowest concavity. As mentioned previously, high values of δ decrease concavity of the penalty function and make it behave more like the ℓ_1 penalty, which could introduce considerable bias into the estimate of μ_k. Therefore, in order to determine whether the value of δ needs to be decreased, we define the bias-variance ratio (BVR) for each cluster C_k as

$$\mathrm{BVR}_k = \begin{cases} \frac{\|\mu_k - \bar{y}_k\|_2^2}{\sum_{i \in C_k} \|y_i - \bar{y}_k\|_2^2/(N_k - 1)}, & N_k > 1 \\ \\ \frac{\|\mu_k - y_i\|_2^2}{(r_k/2)^2}, & N_k = 1, C_k = \{i\}, \end{cases} \tag{2.12}$$

where $r_k = \min_{\ell \neq k} \|y_i - \mu_\ell\|_2$ is the distance between y_i and the nearest cluster center. If $\mathrm{BVR}_k > 1$ for any k, then we decrease δ as in (2.11). The idea behind the bias-variance ratio is that if a certain estimated cluster center μ_k moves beyond the range of the observations y_i in that cluster C_k, then the concavity is increased in order to reduce the bias that could lead to a bad solution.

We now address the choice of $\Lambda(\delta_h)$ determined by the lower and upper bounds of the sequence, $\lambda_1(\delta_h)$ and $\lambda_G(\delta_h)$. The values of λ in between are evenly spaced in log-scale. We state two lemmas for a simple case with $n = 2$ and then use these lemmas to motivate our choice of the lower and upper bounds for $\Lambda(\delta_h)$. The proofs for the lemmas are provided in the Appendix.

Lemma 1. *Assume $n = 2$ and that there are only two points (y_1, y_2) with distance $d = \|y_1 - y_2\|_2$. Let $\theta_i^{(t)}$ be the value of θ_i generated by the MM algorithm with $\theta_i^{(0)} = y_i$, $i = 1, 2$. Fix $\lambda\delta = \eta > 0$. For a given $\phi \in (0, 1)$, if $\eta > d$ and*

$$\lambda = \frac{2\phi\eta d}{(1-\phi)\,(\eta - d)}, \tag{2.13}$$

then $\|\theta_1^{(1)} - \theta_2^{(0)}\|_2 = (1 - \phi)\|\theta_1^{(0)} - \theta_2^{(0)}\|_2$. If $\eta \leq d$, then $\theta_i^{(t)} = y_i$ for all $t \geq 1$ and $i = 1, 2$.

Lemma 2. *Assume $n = 2$ and that there are only two points (y_1, y_2) with distance $d = \|y_1 - y_2\|_2$. For a given $\delta > 0$, if*

$$\lambda \geq \left(1 + \frac{1}{\delta}\right) d, \tag{2.14}$$

then the global minimizer of (2.3) *is given by $(\hat{\theta}_1, \hat{\theta}_2) = (\bar{y}, \bar{y})$.*

We use Lemma 1 to determine δ_1 and the initial lower bound $\lambda_1(\delta_1)$. From this lemma one sees that η serves as a threshold: if $d \geq \eta$, then $\theta_i^{(t)}$ will not change, and the points will not merge. Let $\eta_1 = \lambda_1(\delta_1)\delta_1$ and Q_β be the β-quantile of the nearest neighbor distances among y_i's. We choose $\eta_1 = Q_\omega$, where $\omega \in (0, 1)$ can be regarded as the approximate proportion of data points that may merge in the initial solution. On the other hand, it follows from Lemma 1 that $\phi \in (0, 1)$ can be considered the minimization step size. By default, we set $\phi = 0.5$. Then, to use (2.13) to determine $\lambda_1(\delta_1)$ we need to specify d. We may choose d as the distance between a pair of points such that $d < \eta_1$. This can be achieved by simply setting $d = Q_\tau$, where $\tau \in (0, \omega)$. Plugging these choices of parameters into (2.13), we obtain the initial lower bound for λ as

$$\lambda_1(\delta_1) = \frac{2\phi Q_\omega Q_\tau}{(1-\phi)\,(Q_\omega - Q_\tau)}. \tag{2.15}$$

The first value of the sequence Δ then follows directly from

$$\delta_1 = Q_\omega / \lambda_1(\delta_1). \tag{2.16}$$

Denote the maximum penalty by $z = \lambda \max_t \rho(t) = \frac{1}{2}\lambda^2\delta$. In order to achieve gradual merging of objects into fewer clusters, both the threshold $\eta = \lambda\delta$ and z should be non-decreasing. Suppose that δ_{h-1} is decreased to δ_h by the BVR criterion and $z_{h-1} = \frac{1}{2}[\lambda_{\tilde{G}}(\delta_{h-1})]^2\delta_{h-1}$, where $\lambda_{\tilde{G}}(\delta_{h-1})$ is the value of λ before decrease. The lower bound for $\Lambda(\delta_h)$ is then

$$\lambda_1(\delta_h) = \left(\frac{2z_{h-1}}{\delta_h}\right)^{1/2} = \left(\frac{[\lambda_{\tilde{G}}(\delta_{h-1})]^2\,\delta_{h-1}}{\delta_h}\right)^{1/2} = \alpha^{-1/2}\lambda_{\tilde{G}}(\delta_{h-1}). \tag{2.17}$$

Next, we directly apply Lemma 2 to define the upper bound $\lambda_G(\delta_h)$, $h = 1, \ldots, H$. Given a collection of objects, we can conservatively choose d in (2.14) to be the maximum distance among all pairs of objects to obtain

$$\lambda_G(\delta_h) = \left(1 + \frac{1}{\delta_h}\right) \max_{i,j} \|y_i - y_j\|_2. \tag{2.18}$$

For a collection of $n > 2$ objects, the upper bound in (2.18) becomes only an approximation for the value of λ such that all objects merge into a single cluster. This value of λ does not necessarily guarantee that the objects will merge, but in practice, we have not encountered a situation when this approximation did not work. In a case when the value of the upper bound of λ is not sufficiently large, one can simply decrease the value of δ, recalculate the new sequence of λ and run the algorithm until all data points form a single cluster.

In summary, the construction of the solution path involves the specification of four parameters, α in (2.11), and ω, ϕ and τ in (2.15), all ranging between 0 and 1. In effect, the parameters τ and ϕ control the step size of the MM iteration, and consequently the length of the solution path. Based on our experience and as demonstrated by the sensitivity analysis in Section 2.4.5, the solution path is not much affected by the choice of ϕ and τ, except when very small values are used ($\phi, \tau \leq 0.01$), which can slow down the progression of the solution path. We recommend setting τ to be slightly smaller than ω in order to avoid unnecessary detail in the solution path. In general, we recommend setting $\alpha = 0.9$, $\phi = 0.5$ and $\tau = 0.9\omega$, and we use these default values throughout the paper.

The only tuning parameter that needs to be specified by the user is ω for the calculation of $\lambda_1(\delta_1)$ in (2.15) and δ_1 in (2.16). Since it stands for the approximate proportion of the nearest neighbors that may merge initially, the nature of the dataset might help determine its value. For instance, if the dataset is very noisy and clusters are not tight, ω could be set to a low value, and if the dataset is well separated into clusters and there is little noise, a high value of ω could be used. If chosen too high in cases where the data is very noisy, ω could force the algorithm to skip the correct solution by initially merging too many noisy observations. In practice, we also found that ω should be small in high-dimensional settings. In this paper we use $\omega = 0.1$ for the high-dimensional examples when $n < p$ and $\omega = 0.5$ when $n > p$.

2.3.3 The full SPC algorithm

We now combine the MM algorithm with the construction of the solution path to describe the full SPC algorithm. It is initialized assuming that each observation forms its own singleton cluster and is run until all observations merge into one cluster using a sequence of (δ, λ). The solution obtained from a particular (δ, λ) is used as a warm start for the next solution. We do not require the input of the number of clusters K, and the algorithm typically yields a short path of 2–15 solutions.

For a particular $(\delta_h, \lambda_g(\delta_h))$, let $K(h,g)$ be the estimated number of clusters and $\hat{\mu}_k(h,g)$ for $k = 1, \ldots, K(h,g)$ be the estimated cluster centers. Using these notations, the full SPC algorithm is provided in Algorithm 2.

Algorithm 2 Solution path clustering

```
Inputs
required input: $Y = (y_{im})_{n \times p}$, $\omega \in (0,1)$
default input: $\tau = 0.9\omega$, $\phi = 0.5$, $\alpha = 0.9$, $G = \min(20, p)$
initialization: $h = 1$, $K = n$, $\mu_k = y_k$, $k = 1, \ldots, n$

1: repeat
2:     compute $\delta_h$, $\lambda_1(\delta_h)$, $\lambda_G(\delta_h)$ and construct $\Lambda(\delta_h)$ in logarithmic scale of size $G$
3:     for $g = 1, \ldots, G$ do
4:         run the MM algorithm (Algorithm 1) until $\max_k \|\mu_k^{(t+1)} - \mu_k^{(t)}\|_2 < \xi$ or $t > 50$ to obtain $K(g,h)$ and $\{\hat{\mu}_k(h,g), k = 1, \ldots, K(h,g)\}$
5:         for $k = 1, \ldots, K(h,g)$ do
6:             compute $\mathrm{BVR}_k$
7:             if $\mathrm{BVR}_k > 1$ then
8:                 $h \leftarrow h + 1$ and go to line 2
9:             end if
10:        end for
11:    end for
12:    $h \leftarrow h + 1$
13: until $K(h,g) = 1$
```

2.4 Simulation study

2.4.1 Competing methods and cluster quality assessment

We illustrate the performance of SPC on simulated data and compare it to the k-means++ algorithm [AV07], the convex clustering method of [CL13], and several popular clustering algorithms that were developed to account for the noise in data. For the latter purpose we selected model-based clustering (mclust) [FR02], tight clustering [TW05b], and penalized weighted k-means (PWK-means) [Tse07].

The majority of authors use the adjusted rand index (ARI) [HA85] to compare the clustering results across different methods when the true cluster assignment is known. ARI calculates the similarity of a clustering result to the underlying true clustering assignment. It yields a maximum value of 1 if the clustering solution is identical to the true structure and yields a value close to 0 if the clustering result is obtained from random partitioning. The exact definition of the ARI can be found in the Appendix. For the simulation study we also use ARI in order to be consistent with the prevailing method of cluster quality assessment.

Since most of the competing methods will detect noise, i.e. the data points that do not belong to any well-defined clusters, we calculate two ARI scores $(\mathrm{ARI}_c, \mathrm{ARI}_n)$ for each of them. Suppose the true partition and an estimated partition are $C = \{C_1, \ldots, C_R, C_{R+1}\}$ and $\hat{C} = \{\hat{C}_1, \ldots, \hat{C}_K, \hat{C}_{K+1}\}$, where C_r and $\hat{C}_k$ contain the indices of the data points assigned to true and estimated clusters for $r = 1, \ldots, R$ and $k = 1, \ldots, K$, respectively, and C_{R+1} and $\hat{C}_{K+1}$ are the indices assigned to noise. Denote the respective cluster labels as $v = \{v_1, \ldots, v_R, v_{R+1}\}$ and $\hat{u} = \{\hat{u}_1, \ldots, \hat{u}_K, \hat{u}_{K+1}\}$, where v_{R+1} and $\hat{u}_{K+1}$ indicate the labels for the true and estimated noise, respectively. See Table 2.1 for the full contingency table of the counts $n_{kr} = |\hat{C}_k \cap C_r|$. Our ARI scores are calculated based on parts of the counts in this table.

ARI_c accounts for data points that are identified as belonging to estimated

Table 2.1: Contingency table and notation for the calculation of $(\text{ARI}_c, \text{ARI}_n)$

Cluster	v_1	$\dots$	v_R	v_{R+1}	Sum
$\hat{u}_1$	n_{11}	$\dots$	n_{1R}	$n_{1(R+1)}$	$n_{1\bullet}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\hat{u}_K$	n_{K1}	$\dots$	n_{KR}	$n_{K(R+1)}$	$n_{K\bullet}$
$\hat{u}_{K+1}$	$n_{(K+1)1}$	$\dots$	$n_{(K+1)R}$	$n_{(K+1)(R+1)}$	$n_{(K+1)\bullet}$
Sum	$n_{\bullet 1}$	$\dots$	$n_{\bullet R}$	$n_{\bullet(R+1)}$	n

clusters $\hat{C}_k$, $k = 1, \dots, K$, and is calculated as in (2.24) in the Appendix using only the first K rows of Table 2.1, $\{n_{kr} : 1 \le k \le K,\ 1 \le r \le R+1\}$, with the corresponding row and column sums. In effect, ARI_c provides a quality assessment of the identified clusters $\hat{C}_k$, i.e. misclassification of clustered data and the amount of noise in the estimated clusters is reflected as lower values of ARI_c.

ARI_n indicates how sensitive a method is in identifying noise and whether any clustered data point is misclassified as noise. It is based on all the data points except the noise in the estimated clusters, which is accounted for in ARI_c. We collapse Table 2.1 to a 2×2 table with counts $n^*_{cc} = \sum_{k=1}^{K} \sum_{r=1}^{R} n_{kr}$, $n^*_{nn} = n_{(K+1)(R+1)}$, $n^*_{nc} = \sum_{r=1}^{R} n_{(K+1)r}$, and $n^*_{cn} = \sum_{k=1}^{K} n_{k(R+1)}$. We set $n^*_{cn} = 0$ since we account for these data points in ARI_c. The total number of data points to be considered for ARI_n is thus $n^* = n^*_{cc} + n^*_{nn} + n^*_{nc}$. Again, we use (2.24) in the Appendix to calculate ARI_n by plugging in n^*_{cc}, n^*_{nn}, n^*_{nc}, and $n^*_{cn} = 0$. In general, if n^*_{nn} is large and n^*_{nc} is small, ARI_n will be close to 1.

PWK-means, tight clustering, and k-means++, which is a classical k-means method combined with a randomized seeding procedure to select the starting centers, require as an input, an estimated number of clusters, and mclust requires an input of a range of the number of clusters. In most cases, we provided the comparison algorithms with ideal input parameters, which will likely result in optimal performance for these methods. In addition to the number of clusters, the competing methods, except k-means++, have other tuning parameters that we mention below.

PWK-means also requires the input of a penalty parameter λ since this method imposes a penalty on the number of noisy data points. We used the suggested prediction-based resampling method [TW05a] to find λ, calculated the prediction strength criterion for an increasing sequence of λ, and selected the value of λ corresponding to the highest prediction strength computed.

Tight clustering has several tuning parameters, but most of them are recommended to stay at their default values, which we follow for the simulated data. Along with the user-specified target number of clusters k_{target}, tight clustering relies on a starting number of clusters $k_0 > k_{\text{target}}$. The tight clustering algorithm is then applied to a decreasing sequence, decremented by 1, starting with k_0 and ending with k_{target}. The authors recommend $k_0 \geq k_{\text{target}} + 5$, however, a too large k_0 results in smaller clusters and many of the clustered data identified as noise. Conversely, a small k_0 can result in a smaller size of the estimated noise and more noise assigned to clusters. We set $k_0 = k_{\text{target}} + 5$, the smallest recommended value.

One additional tuning parameter of mclust is the reciprocal of the hypervolume V of the data region, and the authors note that the method is sensitive to this value. The default method of calculating the hypervolume is $V = \prod_{j=1}^{p}(\max_i\{y_{ij}\} - \min_i\{y_{ij}\})$, which we use for the simulated data. Another input into mclust is the estimated categorization of each data point as clustered data or noise. Once the categorization is provided, mclust applies hierarchical clustering to the identified clustered data to get a good initialization for the EM algorithm that generates a final clustering result. We have used the recommended Kth-nearest neighbor cleaning method [BR98] to obtain the initial categorization into clusters and noise. It is suggested that the value of K should be the size of the smallest cluster to be detected, but if K is selected too high, the noise identification might not perform well. We chose $K = 5$ for the nearest neighbor cleaning such that it is smaller than the average cluster size.

Finally, convex clustering requires the specification of three tuning parameters, which are the Gaussian kernel weight ϕ, the number of nearest neighbors k, and an increasing sequence of the regularization parameter γ for obtaining a solution path. We chose the parameter ϕ such that the penalty weights $w_{ij} = \iota^k_{\{i,j\}} \exp(-\phi\|y_i - y_j\|_2^2)$, where $\iota^k_{\{i,j\}}$ is 1 if y_j is among y_i's k-nearest neighbors and 0 otherwise, are on average around 0.25. This rule forced the parameter ϕ to be very close to 0 for our simulated data, but any larger values for ϕ resulted in unsatisfactory clustering. We also used $k = 5$ and an increasing sequence of $\gamma \in [0, 50]$ of size 20 at even intervals, following an example in the documentation for the R package `cvxclust`.

To compare the performance, we created four different clustering scenarios: 1) well-separated clusters, 2) overlapping clusters, 3) well-separated clusters with added noise, 4) overlapping clusters with added noise. All clusters in the examples in Sections 2.4.2 and 2.4.3 are spherical, with equal variance. The cluster centers and the noise points were generated from a uniform distribution on $[-5, 5]^p$. All noise was generated outside of the radius of the clusters, where the radius is the largest distance from the cluster center to the data points in that cluster. Overlapping clusters were generated such that 15–20% of the data points in a pair of clusters are located within the radiuses of both clusters. For each scenario, we simulated 20 datasets with $n > p$ and $n < p$. To demonstrate the ability of SPC to identify noise and for the purposes of comparison with other methods, we simply regard all estimated clusters of size $N_k \leq 3$ as noise. The same cut off is also used for defining noise from k-means++ and convex clustering results.

2.4.2 Results for $n > p$

The simulated datasets for both well-separated and overlapping scenarios when $n > p$ are of size $n = 400$, dimension $p = 20$, and with $K = 10$ clusters. For each of the two scenarios with noise, 200 uniformly distributed noise points were added to the clustered data. We chose $\omega = 0.5$ for all the scenarios assuming that about half of the nearest neighbors should merge. As the output from each dataset, we obtained a solution path of 7–12 solutions, each containing the number of clusters K with size $N_k > 3$, estimated cluster centers $\hat{\mu}_k$, and cluster assignments C_k.

The ARI_c and ARI_n scores for the comparison with other methods are presented in Figure 2.3. In addition to the true number of clusters $K = 10$, we supplied the competing methods with the number of clusters (of size $N_k > 3$) along the SPC solution path to demonstrate their performance when the number of clusters is misspecified. We report the ARI scores averaged over 20 datasets for $K = 10$ and for different ranges of K, e.g. $6 - 9$, since each of the SPC solution paths for the 20 datasets might contain a different number of clusters. It must be noted that $\mathrm{ARI}_n = 0$ when no noise is detected, which is misleading for the scenarios without noisy data. Thus, we use a special score, $\mathrm{S}_n = 1 - n^*_{nc}/n$, for these scenarios, such that if no noise is identified by a method, then $\mathrm{S}_n = 1$. As shown in Figure 2.3, SPC can clearly outperform k-means++, especially in the scenarios with noise, due to the fact that k-means++ is not designed to separate noisy observations. SPC performs similarly or slightly better in most scenarios compared with tight clustering and PWK-means. For $K = 10$, mclust outperforms all the other methods in all scenarios and separates the overlapping clusters and noise well. The pre-classification of the noise and hierarchical clustering of the remaining data provide mclust with an excellent initialization. The spherical nature of the clusters and uniformly generated noise also perfectly match the model assumptions of mclust. For $K = 10$ or a slightly smaller value, the performance of SPC is very comparable to that of mclust for noisy data scenarios, although

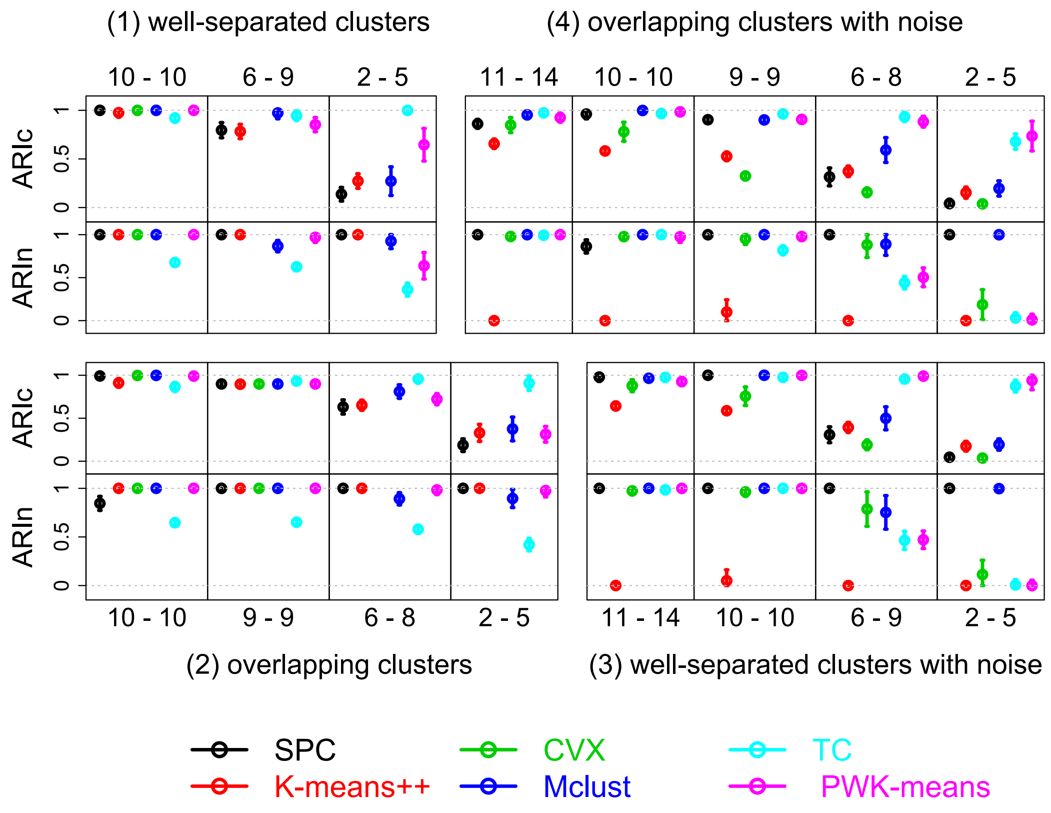


Figure 2.3: ARI_c and ARI_n for comparison methods for all four scenarios when $n > p$. Each scenario is represented by a panel consisting of a block of plots. Each colored point indicates the average ARI_c or ARI_n across 20 datasets and the error bar shows its 95% confidence interval. ARI scores are between 1 (best) and 0 (worst) as labeled on the y-axis. The range of numbers above or below each plot refers to the number of clusters of size $N_k > 3$ found by SPC. Different colors correspond to the different competing methods, as shown in the color legend with convex clustering and tight clustering abbreviated as CVX and TC, respectively. For scenarios (1) and (2) ARI_n reports $\text{S}_n = 1 - n^*_{nc}/n$.

our method makes much weaker data generation assumptions and does not use any specific initialization.

All the methods recover well-separated clusters with accuracy whether noise is present or not; however, tight clustering tends to considerably underperform when no noise is present due to the fact that it is targeted specifically for noisy data. When clusters are not well-separated, SPC tends to merge overlapping clusters but identifies noise very well (high ARI_n). With the progression toward greater sparsity (smaller K), SPC adds more and more noise into the clusters or creates bigger clusters from noise, which results in a low ARI_c score. On the other hand, the remaining noise is identified accurately, which is reflected in the high ARI_n scores. In contrast, when the number of clusters is misspecified and $K < 10$, tight clustering and PWK-means have the tendency to leave out clustered data as noise, reflected by low ARI_n scores, while mclust merges the closest clusters together. Convex clustering performs well with overlapping clusters; however, compared to SPC, it tends to produce less satisfactory results when any noise is present. In the scenarios with noise, aside from k-means++, convex clustering adds the most noise into the clusters.

Figure 2.4 shows an example of the cluster assignment for the overlapping scenario with noise for different methods. Figures 2.4a–2.4b show cluster assignments for two consecutive solutions with $K = 10$ and $K = 9$ for SPC. The solution for $K = 10$ leaves out a number of clustered data points as noise. The solution with $K = 9$ separates the noise perfectly, but merges two overlapping clusters. Figures 2.4c–2.4f show the results of competing methods with $K = 10$. One sees that mclust (Figure 2.4d) separates noise well and misclassifies only one noisy point, while tight clustering (Figure 2.4e) and PWK-means (Figure 2.4f) add noise to the clusters. Convex clustering (Figure 2.4c) has the most noise added to the clusters.

Finally, we provide some empirical evidence, which suggests that the SPC algorithm may reach the stopping criterion (2.10) quickly with the simulated data. Let T^* be the number of iterations before meeting the stopping criterion (2.10). Figure 2.5a shows the histogram of T^* for the combined four scenarios and confirms

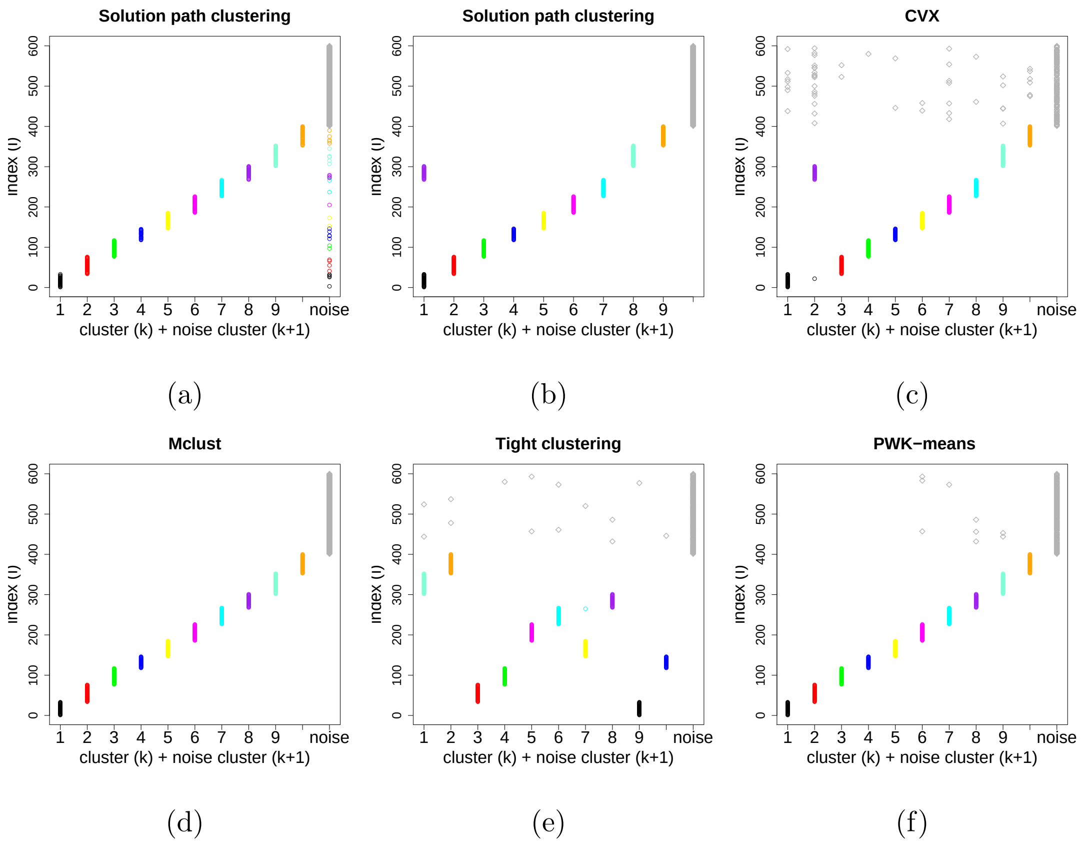
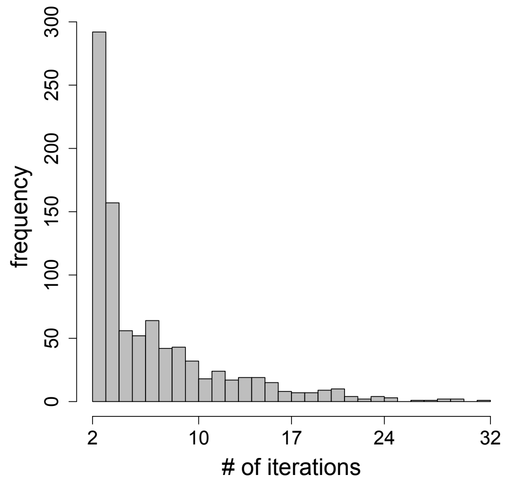
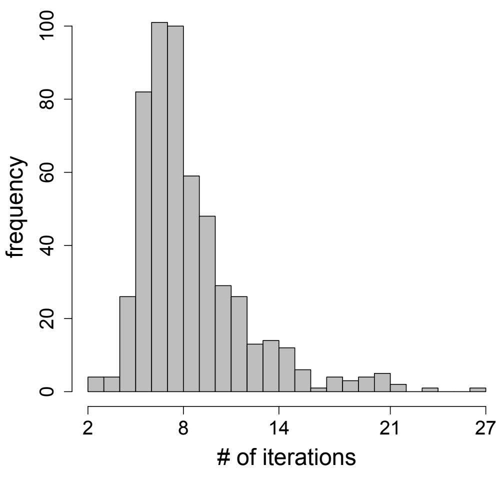


Figure 2.4: Cluster assignment plots for comparison methods. The y-axis displays the index i of each data point and the x-axis shows K estimated clusters and one additional cluster for the noise. The color indicates the true cluster assignment with true noise in gray, such that if a clustered data point is identified as noise it will appear as noise on the x-axis but will have a color associated with it. If noise is misclassified then it will appear in the clusters but in gray color. (a)–(b) Cluster assignments for solution path clustering for $K = 10$ and $K = 9$, corresponding to two consecutive solutions in a solution path. (c)–(f) Cluster assignments for $K = 10$ for convex clustering (CVX), mclust, tight clustering, and PWK-means.

that the SPC algorithm terminated within 35 iterations per solution for all the datasets. We note that T^* generally decreases along a solution path due to warm starts and the reduction in K.



(a) $n > p$



(b) $n < p$

Figure 2.5: Histograms of T^* for the combined four scenarios for the simulated data.

2.4.3 Results for $n < p$

For $n < p$ we simulated datasets with $n = 100$ clustered data points, $p = 200$, and $K = 10$. For each dataset, 50 uniformly distributed noise points were added to the clustered data. Similarly to the scenarios with $n > p$, we consider clusters of size $N_k \leq 3$ as noise so that the results can be compared across the methods.

With high-dimensional data we chose $\omega = 0.1$ in order to create a longer and a more detailed solution path. By setting a smaller ω, fewer observations merge into clusters in the initial stages so that the majority of data points are left as singletons or very small clusters. As the sparsity is increased, more observations form new clusters. Consequently, even though the total number of clusters (K_{total}) decreases, the number of clusters of size $N_k > 3$ (K_{clust}) increases along the solution path. As shown in the results for SPC in Figure 2.6, ARI_c is always high because almost no noisy points merge into clusters and there are few misclassified clustered data points. On the other hand, ARI_n increases as more clusters are formed so that

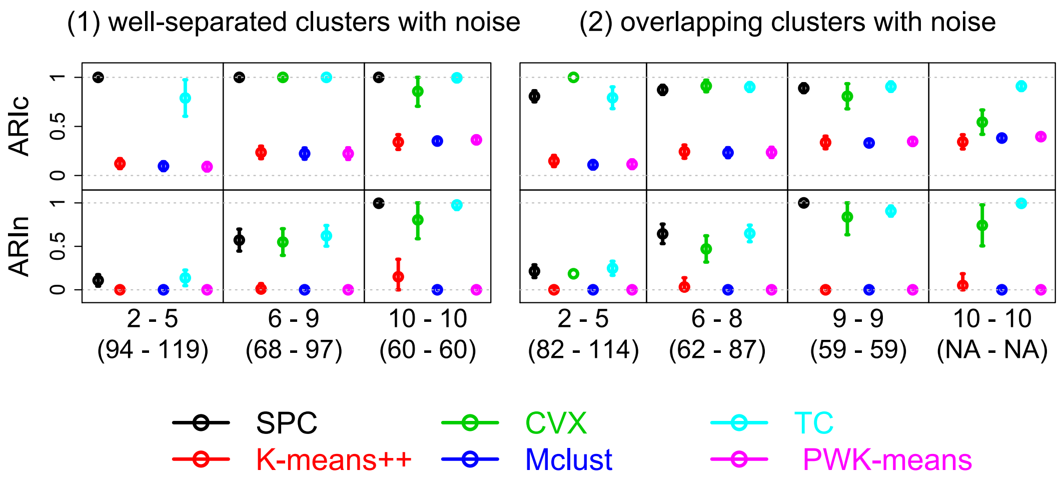


Figure 2.6: ARI_c and ARI_n for comparison methods for two scenarios with noise when $n < p$. The two scenarios are presented in two panels of plots in the same format as those in Figure 2.3. The x-axis indicates K_{clust} ranges and K_{total} ranges in brackets. NA on the x-axis in the last column in scenario (2) reflects that SPC did not obtain a solution path with $K = 10$ for this scenario.

fewer data points are regarded as noise. One sees that when K_{clust} is close to the true number of clusters ($K = 10$), satisfactory results are obtained with SPC in terms of both cluster assignment and noise detection.

When the number of dimensions p is higher than the number of observations n, SPC shows better performance than all the other methods in most scenarios. It performs at least as well as or slightly better than tight clustering. We could not obtain solutions for mclust due to error messages when the initial noise categorization was provided for the EM initialization, and thus, we used the solutions without the pre-specification of noise, which resulted in all the noise included in the clusters. This clearly demonstrates that mclust is sensitive to pre-specification of noise. K-means++ recognized clustered data quite well but grouped the noise with the clusters. The performance of mclust and k-means++ was very similar to that of PWK-means, which also did not recognize noise with penalty parameter λ selected by prediction-based resampling. This comparison shows that identification of noise in high-dimensional space is very challenging and that the uniform assumption for the noise may not be appropriate because the volume of the data range becomes huge when p is large.

Convex clustering showed slightly worse ARI scores in the $n < p$ scenarios, adding noise to the clusters as well as leaving out the clustered data as noise, which indicates that high dimensionality might create challenges for this method. The distances between observations in high-dimensional data become very large, forcing the parameter ϕ to be very close to 0 in order to produce reasonably large weights w_{ij}, which corresponds to uniform w_{ij} and introduces more bias into the estimation of the cluster centers, possibly leading to unsatisfactory solutions. The k-nearest-neighbor approach could mitigate this problem somewhat, but as mentioned in qualitative comparisons section in [CL13], can still lead to data points not being agglomerated correctly.

As in the previous section, we demonstrate in Figure 2.5b the empirical evidence that the SPC algorithm reached the stopping criterion (2.10) for the simulated data with $n < p$. From the figure we see that T^* is somewhere between 3 and 28 for each solution. In the high-dimensional case T^* is larger on average (about 8 iterations vs. about 4 for $n > p$).

Table 2.2 summarizes an example of the running time for all the methods except kmeans++, which is by far the fastest method. The run time for SPC is based on the longest solution path for each scenario, usually 10–14 solutions. The matching run times for mclust, tight clustering and PWK-means are calculated based on the unique cluster counts and those for convex clustering are based on its corresponding full solution path of length 20. If the number of clusters K_{clust} (of size $N_k > 3$) is duplicated in a solution path, these duplicates are not included in the run times of mclust, tight clustering and PWK-means. The running time for

Table 2.2: Summary of run times (in seconds) for SPC and comparison methods

	$n > p$					$n < p$				
	SPC	CVX	mclust	TC	PWK	SPC	CVX	mclust	TC	PWK
(1)	4.56	1.87	1.36	9.42	19.07	4.26	3.63	0.92	812.75	17.60
(2)	3.88	3.76	1.34	12.61	19.26	2.68	3.36	0.95	369.51	14.29
(3)	11.95	19.14	1.43	10.32	39.40	7.38	1.69	0.89	11.16	4.29
(4)	11.43	24.50	1.56	10.36	41.29	9.10	0.76	1.08	11.92	4.64

NOTE: (1) well-separated clusters, (2) overlapping clusters, (3) well-separated clusters with noise, (4) overlapping clusters with noise.

PWK-means includes penalty parameter search with prediction-based resampling. SPC is currently implemented in R, while tight clustering and PWK-means are written in C, mclust in Fortran, and convex clustering in R and Fortran. It seems that SPC's run times compare well to the other methods and that it has good potential in terms of speed, especially after implementation in a faster language.

2.4.4 Results for non-convex and non-spherical clusters

We now show the performance of SPC on non-convex and non-spherical simulated clusters. For the non-convex case we generated $n = 400$ observations in $p = 2$ dimensions grouped in $K = 4$ clusters, where data in each cluster were generated from a normal distribution with a high correlation. Three out of the four clusters have a negative or positive correlation of around 0.9 and the other one has a slightly lower correlation of 0.5. After the data points were simulated, non-convexity was introduced to each cluster as shown in Figure 2.7 (true model). The cluster size was varied to be $N_1 \approx 250$, $N_2 \approx 100$, $N_3 \approx 30$, and $N_4 \approx 20$. Finally, 20 noise data points were added similarly to the scenarios in the previous sections. We then applied all the methods except tight clustering, which did not

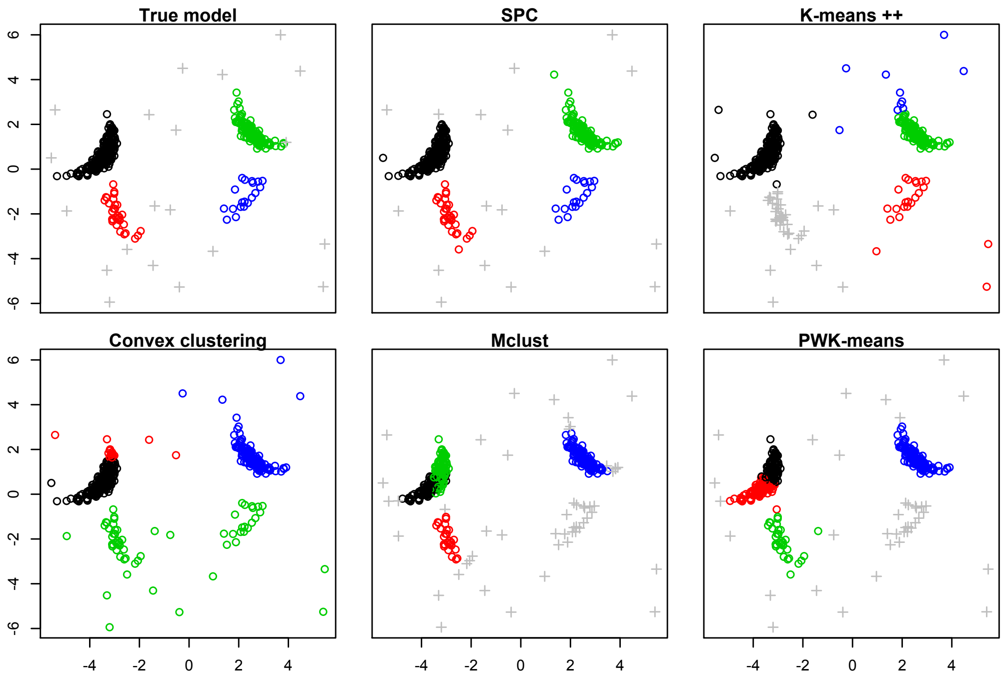


Figure 2.7: Cluster assignment results for non-convex clusters. Colored points indicate data in clusters and gray points indicate noise.

converge, to this simulated dataset.

It can be seen from Figure 2.7 that SPC can perform well in relatively complex settings when the clusters are non-convex, the noise or outliers are present and when the cluster sizes are different. All the comparison methods tend to split one of the two bigger clusters (black and green in the true model plot), while SPC correctly does not. Mclust and PWK-means perform well with noise; however, they do not recognize the smallest cluster with the lowest correlation (blue). Mclust also assigns the ends of the non-convex clusters to noise, showing its sensitivity to the assumption of normality. Convex clustering does not seem to be robust to outliers and noise, due to the design of its penalty, specifically the weights, even though it can obtain very good results for non-noisy non-convex data for exactly the same reason.

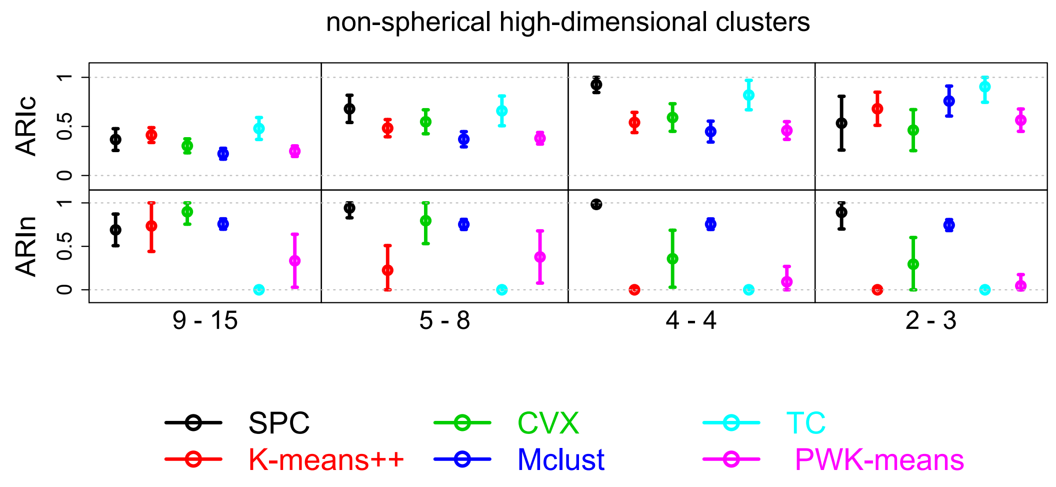


Figure 2.8: ARI$_c$ and ARI$_n$ for comparison methods for non-spherical simulated clusters with $p = 20$ and $K = 4$ true clusters, in the same format as the panels in Figure 2.3.

To further test the relative performance between SPC and the competing methods we applied them in higher dimensions. We generated $K = 4$ similarly correlated clusters in $p = 20$ dimensions with the same sizes and added 50 noise points. We did not, however, introduce any non-convexity into the clusters. The resulting ARI scores, averaged over 20 randomly generated datasets, are presented in Figure 2.8. SPC has the highest ARI$_c$ and ARI$_n$ scores for the true number of clusters $K = 4$ and also performs well when the number of clusters is misspecified. This confirms the usefulness of SPC for clustering high-dimensional non-spherical data.

2.4.5 Sensitivity to tuning parameters

In this section, we further comment on the sensitivity of the solution path to the tuning parameters ω, ϕ, τ in (2.15), and α in (2.11). As mentioned in the previous section, the parameter ω could have a considerable impact by skipping

Table 2.3: Solution path length and (ARI_c, ARI_n) scores for different tuning parameters ϕ and τ

τ	$\phi = 0.05$	$\phi = 0.1$	$\phi = 0.3$	$\phi = 0.5$	$\phi = 0.7$
0.05	6 (0.91, 0.99)	9 (0.90, 1.00)	10 (0.91, 1.00)	10 (0.91, 1.00)	11 (0.91, 0.99)
0.15	8 (0.90, 1.00)	13 (0.90, 1.00)	10 (0.91, 1.00)	11 (0.91, 0.99)	11 (0.91, 0.99)
0.25	10 (0.90, 1.00)	10 (0.90, 1.00)	10 (0.91, 1.00)	11 (0.91, 0.99)	11 (0.92, 0.99)
0.35	11 (0.90, 1.00)	10 (0.91, 1.00)	11 (0.91, 0.99)	11 (0.91, 1.00)	11 (0.91, 0.99)
0.45	10 (0.91, 0.99)	10 (0.91, 1.00)	11 (0.91, 1.00)	11 (0.92, 0.99)	11 (0.92, 0.99)

the best solution in the initial step if set too high. This can be avoided by setting this parameter to a low value at the expense of a more detailed solution path with a longer running time. We have found, however, that the rest of the tuning parameters do not impact the solution path in any major way. Setting a different value for α will automatically trigger a corresponding change in the value of λ, which might affect the speed of the algorithm. To further demonstrate the sensitivity to τ and ϕ, we have run SPC on 20 randomly generated datasets with different combinations of these parameters and averaged the length of the solution path with ARI_c and ARI_n for each combination. Table 2.3 is based on the overlapping case with noise for $n > p$ and illustrates that the cluster assignment and noise detection are not sensitive at all to these two parameters. The length of the solution path, except for the smallest values for both τ and ϕ, is also very stable. The greatest difference between the smaller and larger values of these tuning parameters was the speed of each solution due to the different minimization step size.

2.5 Solution selection

SPC does not require the specification of the number of clusters; however, it produces a solution path. Unlike the entire clustering path of hierarchical clustering or arbitrary regularization paths of convex clustering [CL13, HJB11] or PRclust [PSL13], a SPC solution path includes only a limited number of solutions obtained using an adaptive data-driven approach. This makes it easier for a user to explore different possible clustering assignments. However, it is still very useful in practice to be able to select the most appropriate member along the solution path, especially for large datasets.

Resampling-based methods such as the gap statistic [TWH01] or the clustering instability method in [FW12], are computationally intensive, while Bayesian information criterion and cross-validation have been shown to select models that are too complex compared to the true model in sparse linear regression. These problems are especially relevant for our method that aims at high-dimensional and large data. We therefore adopt the empirical approach in [FZ13] to demonstrate that solution selection for SPC is possible in a simple and fast way.

The approach of [FZ13] is based on the fact that as sparsity decreases, i.e. as the number of clusters increases, the unpenalized log-likelihood of the data will increase. The increase in the number of clusters, then, is justifiable only by a significant increase in the unpenalized log-likelihood. We should choose a solution after which an increase in the number of clusters will not correspond to a large increase in the unpenalized log-likelihood. To determine this, we sort the solution path according to an increasing number of clusters K_{total}, and for each solution s, $s = 1, \dots, S$, we calculate the difference ratio for two adjacent solutions:

$$dr^{(s,s+1)} = \frac{L(K^{(s+1)}) - L(K^{(s)})}{K^{(s+1)} - K^{(s)}}, \tag{2.19}$$

provided that $K^{(s+1)} - K^{(s)} \geq 1$, where

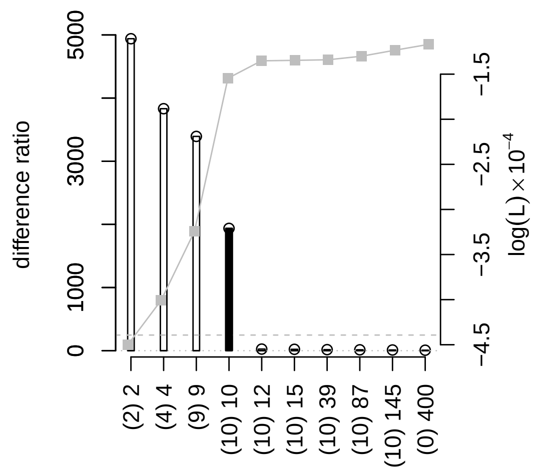
$$L(K^{(s)}) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K^{(s)}} \pi_k^{(s)} \phi(y_i; \mu_k^{(s)}, \Sigma_k^{(s)}) \right] \tag{2.20}$$

is the mixture Gaussian log-likelihood function and $K^{(s)}$ is the total number of estimated clusters K_{total} for that solution. The cluster centers $\mu_k^{(s)}$ and the cluster proportions $\pi_k^{(s)}$ are estimated given the cluster assignments for each solution s. We set the covariance $\Sigma_k^{(s)}$ to be the identity matrix, in line with the implicit assumption behind the use of ℓ_2 loss in (2.1) and (2.5). The likelihood for a singleton cluster in this case is well-defined and in effect amounts to $1/n$. We choose the solution indexed by

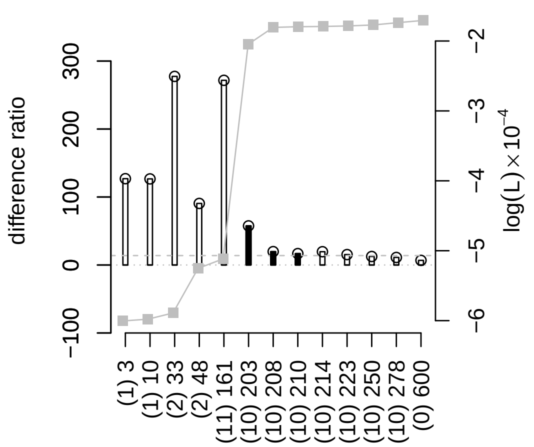
$$K^* = \max \left\{ K^{(s)} : dr^{(s,s+1)} \geq a \times \max \left(dr^{(1,2)}, \ldots, dr^{(S-1,S)} \right) \right\}, \tag{2.21}$$

with $a = 0.05$ as suggested by [FZ13]. We have plotted the difference ratio (2.19) and the unpenalized log-likelihood (2.20) for two simulated datasets in Figure 2.9. In Figure 2.9a the selected solution according to (2.21) is the solution with the true number of clusters $K^* = K_{\text{total}} = K_{\text{clust}} = 10$. An example from the scenario with noise in Figure 2.9b is more ambiguous and in this particular case we would barely choose a solution with $K_{\text{total}} = 214$ and $K_{\text{clust}} = 10$, which does not correspond to the highest ARI score and leaves 4 of the clustered points as noise. Generally, however, the selected solutions have relatively high ARI scores, as reported in Table 2.4. In this table, we demonstrate the averages for ARI_c and ARI_n scores for the selected solutions as well as the averages for the best ARI scores (in terms of the sum of the two ARI scores) along each solution path of the corresponding scenario. One sees that the ARI scores of a selected solution are close to the best along the solution path. The table also reports the average selected K_{clust} and its range, showing that the number of clusters given by a selected solution is close to the true one. All averages and ranges are taken over 20 simulated datasets.

It is generally hard to determine whether singleton or very small clusters are truly noise, and thus, solution selection is more challenging when noisy data points



(a)



(b)

Figure 2.9: The difference ratio (vertical bars) and the unpenalized log-likelihood (gray solid line) as a function of the number of clusters in a solution path. The dashed line represents the cutoff with $a = 0.05$ and the black color highlights the difference ratio for the solutions with the highest ARI scores. The number of clusters on the x-axis is represented by the total number of clusters K_{total} and the number of clusters of size $N_k > 3$ in brackets (K_{clust}). (a) An example from the $n > p$ scenario with well-separated clusters and no noise. (b) An example from the $n > p$ scenario with well-separated clusters and noise.

Table 2.4: Solution selection summary for $n > p$ scenarios

	ARI_c		ARI_n		K_{clust}	
	select	best	select	best	mean	range
(1)	1.000	1.000	1.000	1.000	10	10-10
(2)	0.899	0.935	1.000	0.986	9	9-9
(3)	0.986	1.000	0.979	1.000	10	10-13
(4)	0.940	0.918	0.900	0.987	10	9-10

NOTE: (1) well-separated clusters, (2) overlapping clusters, (3) well-separated clusters with noise, (4) overlapping clusters with noise. ARI_c and ARI_n are averaged over 20 datasets for each scenario for the selected solutions (select) and for the largest ARI scores in each solution path (best).

are present, which is reflected in selected average ARI_n being slightly lower than the best ARI_n scores in Table 2.4. For overlapping scenario with no noise, we tended to select the solutions with $K_{\text{clust}} = 9$ clusters, while the best ARI scores were given by solutions with $K_{\text{clust}} = 10$ (with a few clustered data points misclassified as noise). On the contrary, for the overlapping scenario with noise, most of the solutions selected were given by $K_{\text{clust}} = 10$, which resulted in a higher than best ARI_c but lower than best ARI_n. Altogether this method is very fast and appears to select good solutions for SPC, but more experiments are needed, especially for various possible assumptions on Σ_k.

2.6 Clustering gene expression data

To further test its performance in a real data environment we applied SPC to a gene expression dataset [ZCM07]. The full dataset consists of over 45,000 genes across 16 different experimental conditions that were created to study the regulation of these genes by Oct4, a transcription factor (TF) important for the self-renewal and maintenance of mouse embryonic stem cells. The 16 gene expression profiles include 3 generated from undifferentiated cells which are naturally high in Oct4 expression (conditions 1–3), 5 from early differentiated cells with high Oct4 expression (conditions 4–8), and 8 with low Oct4 expression (conditions 9–16). The study in [ZCM07] has identified and referenced 1,325 Oct4-high genes (Oct4+) and 1,440 Oct4-low genes (Oct4−) out of all the genes, which we regard as two true separated clusters for validation. We then added 3,000 extra randomly selected genes with various levels of coefficient of variation and randomly permuted the expression vector of each gene to obtain noise data points. Thus, we analyzed a final dataset with $n = 5,765$ observations (genes), $p = 16$ dimensions, and $K = 2$ distinct clusters of Oct4+ and Oct4− genes. Following common practice, we normalized the expression data of each gene to have zero mean and unit standard deviation.

We ran SPC with $\omega = 0.5$ and the resulting solution path is presented in Table 2.5. SPC has clearly identified the largest two clusters of Oct4+ and Oct4− genes, which are shown in Figure 2.10a–2.10b. It also suggested an additional smaller cluster of size $N_k = 43$ shown in Figure 2.10c. We do not address in this paper the problem of choosing a proper threshold for the size of a cluster. For demonstration purposes we consider only $K = 2$ largest clusters and regard the rest of the observations as noise, including the small cluster in Figure 2.10c. The left side of Table 2.5 shows the number of clusters of different size per each solution and the right side shows the numbers of Oct4+, Oct4−, and random genes in the two largest clusters (Cluster 1 and Cluster 2) and those in the rest of the genes (Other). The differences among the solutions 1–4 are only due to the random genes forming mostly very small clusters of size $N_k < 30$, and there are only 4 random genes assigned to the Oct4 clusters. It can be seen from the last column that a small number of Oct4 genes were left out as random, but most of

Table 2.5: Solution path for the gene expression data

			# of clusters of size			# of genes (Oct4+ / Oct4− / noise)		
	δ	λ	$[1\text{K}, n)$	$[30, 1\text{K})$	$(1, 30)$	Cluster 1	Cluster 2	Other
1	0.3	6.2	2	1^*	2954	1313/ 0 / 3	0/1392/ 1	12/48/2996
2	0.3	6.7	2	1^*	2868	1313/ 0 / 3	0/1392/ 1	12/48/2996
3	0.3	7.3	2	1^*	2617	1313/ 0 / 3	0/1392/ 1	12/48/2996
4	0.3	8.0	2	1^*	2000	1313/ 0 / 3	0/1392/ 1	12/48/2996
5	0.3	8.7	3	1	837	1315/ 0 / 4	0/1435/ 9	10/ 5 /2987
6	0.3	9.5	1	0	234	1325/1440/2658	0/ 0 / 8	0 / 0 / 334
7	0.3	10.4	1	0	80	1325/1440/2910	0/ 0 / 3	0 / 0 / 87
8	0.3	11.3	1	0	46	1325/1440/2954	0/ 0 / 1	0 / 0 / 45
9	0.3	12.4	1	0	4	1325/1440/2996	0/ 0 / 1	0 / 0 / 3
10	0.3	13.5	1	0	0	1325/1440/3000	0/ 0 / 0	0 / 0 / 0

NOTE: The asterisks ($*$) identify the small cluster of size $N_k = 43$ (Oct4−) in the solution path.

these genes (43 out of 48 Oct4− genes) formed the small Oct4− cluster of Figure 2.10c. In solution 5 the small Oct4− cluster is merged with the main Oct4− cluster and another large cluster of size > 1,000 is formed from many random genes. In solution 6 the majority of the genes merge into a single cluster and in the subsequent solutions the remaining random genes (outliers) are gradually added to this cluster. All the solutions in Table 2.5 were obtained in fewer than 41 iterations.

Table 2.6 compares the result of SPC with that of mclust, tight clustering, and PWK-means. For best results, we ran tight clustering with $K = 2$ and $k_0 = K + 1$ since any larger values of k_0 produced inferior results, with large amounts of Oct4 genes added to the random category. For mclust, we overrode the default value for the hypervolume V and applied the calculation from Section 2.4.1. For the

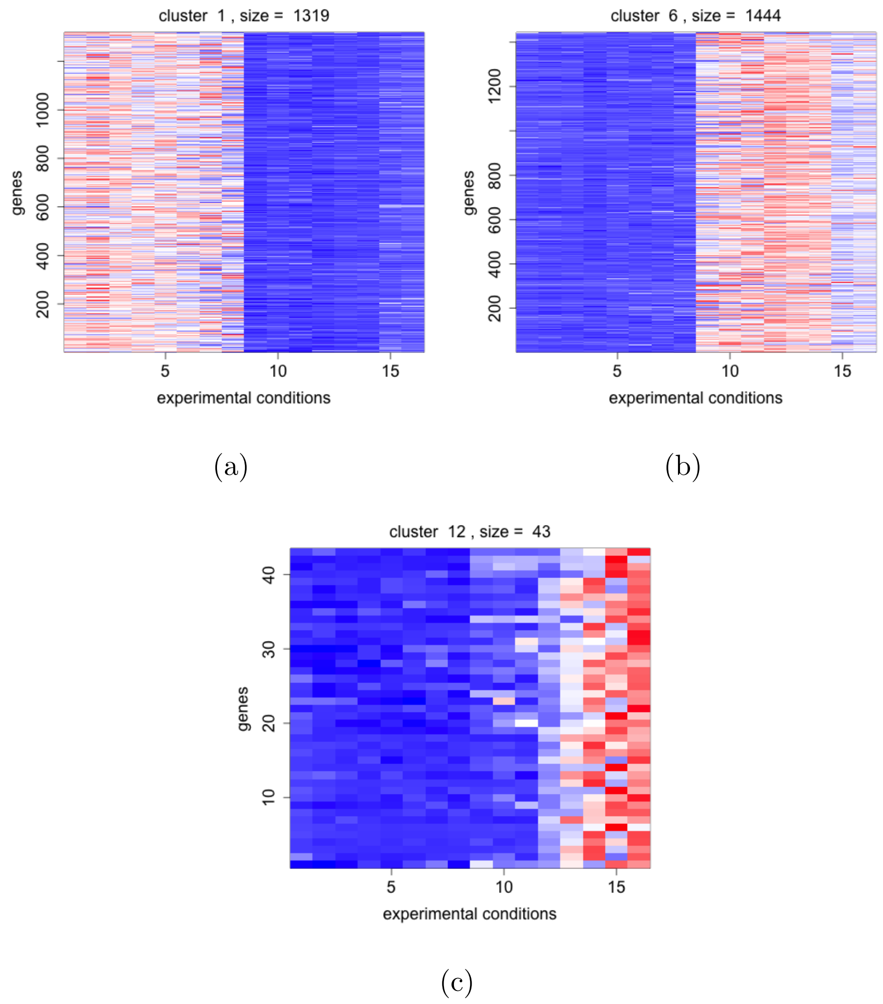


Figure 2.10: Heatmaps of the three largest clusters identified by SPC. Red color indicates high expression and blue color indicates low expression.

nearest neighbor cleaning $K = 5$ was used (the resulting partitioning into Oct4 clusters and random genes was not sensitive for $K < 40$, while if $K \geq 40$ was used, no noise would be identified in this data). Finally, for PWK-means we performed the penalty parameter λ search as with the simulated data. However, we had to start the sequence with a value greater than 1, otherwise the chosen value of λ would be less than 1 by prediction-based resampling and would result in most of the genes classified as random. Overall, we felt that all the other methods required more user fine tuning and rerunning for this dataset than SPC. It should be noted that the variation in the results of tight clustering across different runs was negligible, while the results of the other three methods were deterministic.

Table 2.6: Comparison of different clustering methods for the Oct4-sorted expression data

	# of genes (Oct4+ / Oct4− / noise)		
	Cluster 1	Cluster 2	Other
SPC	1315 / 0 / 3	0 / 1435 / 9	10 / 5 / 2987
mclust	1302 / 0 / 4	0 / 1412 / 5	23 / 28 / 2991
tight clust	1323 / 0 / 108	0 / 1402 / 64	2 / 38 / 2828
PWK	1250 / 0 / 11	0 / 1375 / 13	75 / 65 / 2976

It can be seen from Table 2.6 that none of the methods misclassified genes between the Oct4+ and Oct4− sets in this well-separated data; however, generally SPC added the fewest random genes into the Oct4 clusters and left out the fewest Oct4 genes as random. SPC (solution 5 in Table 2.5) performed better compared to tight clustering, which tended to add more random genes into Oct4 clusters, and PWK-means, which left out more Oct4 genes as random. SPC identified a larger number of Oct4 genes compared to mclust. Even when comparing mclust to solutions 1–4 of SPC, it can be seen that SPC's overestimation of the number of random genes is attributed to the small Oct4− cluster in Figure 2.10c that can arguably be classified as a separate cluster. Indeed, compared with those genes in the big Oct4− cluster in Figure 2.10b, the genes in Figure 2.10c have lower expression levels in conditions 9 to 12 and higher expression levels in conditions 13 to 16, corresponding to cells starting differentiation.

Gene ontology (GO) term enrichment analysis further confirms the specific biological roles of the genes in this newly identified small cluster. Using the full set of the 1,440 Oct4− genes as the background set, we found 64 enriched GO

Table 2.7: Top GO terms enriched in the small Oct4− cluster

GO term	% in cluster	% in full set	P-value
nervous system development	43.6	12.9	1.41×10^{-6}
multicellular organismal process	71.8	34.8	1.99×10^{-6}
cranial nerve development	10.3	0.3	2.27×10^{-6}
sequence-specific DNA binding	25.6	5.6	2.76×10^{-5}
sequence-specific DNA binding TF activity	28.2	6.8	2.86×10^{-5}
nucleic acid binding TF activity	28.2	6.9	3.16×10^{-5}

NOTE: Tabulated are the top three GO terms from the process (top) and the function (bottom) ontology.

terms from the process ontology with $P < 1.7 \times 10^{-4}$ and three from the function ontology with $P < 1 \times 10^{-4}$, both having a false discovery rate (FDR) of less than 0.005% as strong evidence that these genes are not clustered by chance. From Table 2.7 we see that this small cluster is highly enriched for genes involved in nerve development and organismal process and for TFs with sequence-specific DNA binding activity, compared to the corresponding proportions in the full Oct4− set. This analysis demonstrates a strong performance of SPC on a larger gene expression dataset and the possibility of discovering smaller, biologically meaningful clusters by further examining the solution path. Moreover, this shows an example where a detailed solution path is useful even when strong prior knowledge for the number of clusters is available. With the expected number of clusters $K = 2$, all the other methods failed to discover this small cluster. The results of mclust, tight clustering and PWK-means identified solely the two big clusters and only differed in how many random genes were added to the clusters and how many Oct4 genes were left out as random.

2.7 Discussion

SPC is a new general clustering method that provides a small set of solutions, each including a cluster assignment and a number of clusters, in a wide variety of situations including high-dimensional settings when model-based clustering, for example, may have difficulty in providing a solution. SPC can be applied to datasets of different complexities, whether they are small with only a few outliers or large with a high proportion of irrelevant observations. This method searches for the best solution given a certain degree of sparsity penalty, which is determined adaptively by the data. The irrelevant observations are simultaneously identified as singletons or very small clusters. SPC does not require extensive fine tuning and has only one required input tuning parameter, which could have some impact on the resulting set of solutions. However, even this parameter can be set at the lowest possible value, with the only drawback of a longer solution path and running time. Most importantly, SPC does not require the knowledge of the number of clusters and provides its own estimate, depending on the amount of sparsity imposed.

Solution path clustering is effective in separating noise from clustered data; however, it tends to merge overlapping clusters due to its dependence on distances between cluster centers. In the later stages of a solution path, when sparsity is higher, it also tends to add more noise into clusters as the larger distances, usually characterized by noisy observations or outliers, receive more penalization. Overall, SPC has provided satisfactory clustering results across all the simulated data scenarios and the gene expression data.

Penalized estimation methods have been widely applied in linear model analysis and are emerging in clustering. The idea of a solution path or surface along with model selection methodology has been successfully utilized for fast search of sparse models. We extended the regularization path or surface and sparsity approach to unsupervised learning, where the gradual increase of sparsity is guided by the data and the bias of the cluster centers is explicitly adjusted to avoid bad solutions. All these features make SPC different from the majority of the existing penalization frameworks, which either encourage sparsity in cluster means or use a convex penalty for sparsity.

There are several questions that need to be addressed by future research. First, we have not discussed what can be considered a cluster and what can be considered noise. If a cluster consists of just one observation, it could safely be assumed that this singleton cluster is an outlier; however, a decision needs to be made, based on the data, with respect to clusters of size $N_k > 1$. This is easier to determine with smaller datasets, where the cut off $N_k > 1$ is usually sufficient; however, it will not be clear with larger and more complex data that might need higher cluster size cutoff values. We would like to address this issue in the future in a more principled manner. In this chapter, we also only cursorily discuss the methods for selecting the best candidate solution from a solution path and we do not address the asymptotic properties of the cluster center estimates. These and other algorithmic improvements are interesting topics for future work.

2.8 Appendix

2.8.1 Soft thresholding

In order to split an existing cluster, we cycle through every θ_i for $i = 1, \ldots, n$ while fixing $\theta_{[-i]} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n)$ to its current value and minimize

$$\ell(\theta_i) = \|y_i - \theta_i\|_2^2 + \lambda \sum_{k=1}^{K} N_k \rho\left(\|\theta_i - \mu_k\|_2\right), \tag{2.22}$$

over θ_i. Suppose that $\theta_i^{(t)} = \mu_k$ for some k, i.e. the object y_i has been assigned to the kth existing cluster. We cycle through each component of $\theta_i = (\theta_{i1}, \ldots, \theta_{ip})$ and write the difference in each component as $\beta_m = \theta_{im} - \mu_{km}$, $m = 1, \ldots, p$. Let

$$\tilde{\theta}_{im}^{(t+1)} = \frac{y_{im} + \lambda \sum_{\ell \neq k} w_{i,\ell}^{(t)} \mu_{\ell m}}{1 + \lambda \sum_{\ell \neq k} w_{i,\ell}^{(t)}}, \tag{2.23}$$

and

$$\gamma = \frac{\lambda N_k}{1 + \sum_{\ell \neq k} \frac{\lambda N_\ell}{2\|\theta_i^{(t)} - \mu_\ell\|_2}}.$$

Minimization of $\ell(\theta_i)$ in (2.22) reduces to soft thresholding of $\hat{\beta}_m = \tilde{\theta}_{im}^{(t+1)} - \mu_{km}$. Let

$$\beta_m^{(t+1)} = \begin{cases} \hat{\beta}_m - \gamma, & \hat{\beta}_m > 0 \text{ and } \gamma < |\hat{\beta}_m| \\ \hat{\beta}_m + \gamma, & \hat{\beta}_m < 0 \text{ and } \gamma < |\hat{\beta}_m| \\ 0, & \gamma \geq |\hat{\beta}_m|. \end{cases}$$

If $\beta_m^{(t+1)} = 0$ for all m, then $\theta_i^{(t+1)} = \theta_i^{(t)} = \mu_k$ and the corresponding y_i will stay in its current cluster k. If $\beta_m^{(t+1)} \neq 0$ for any m, then the corresponding y_i will be separated or unfused from its current cluster. In this case, $\theta_i^{(t+1)} \neq \mu_k$ and the update for the remainder of the coordinates $\left(\theta_{i(m+1)}, \ldots, \theta_{i(p)}\right)$ will be performed jointly as for $\theta_{im}^{(t+1)}$ (2.23). Algorithm 1 could be modified accordingly to incorporate this split step.

2.8.2 Proof of Lemma 1

Fix $\theta_2^{(t)}$ and one minimization step of the MM algorithm as in (2.8) gives

$$\theta_1^{(t+1)} - \theta_2^{(t)} = \frac{y_1 + \lambda w_{1,2}^{(t)} \theta_2^{(t)}}{1 + \lambda w_{1,2}^{(t)}} - \theta_2^{(t)}.$$

Equating this to $(1-\phi)(\theta_1^{(t)} - \theta_2^{(t)})$ and simply re-arranging the terms, we get

$$\lambda = \frac{2\eta \|(\theta_1^{(t)} - y_1) + \phi(\theta_2^{(t)} - \theta_1^{(t)})\|_2}{\left(\eta - \|\theta_1^{(t)} - \theta_2^{(t)}\|_2\right)_+ (1-\phi)}.$$

For the initial step of the MM algorithm $\theta_1^{(0)} = y_1$ and $\theta_2^{(0)} = y_2$, and thus

$$\lambda = \frac{2\phi\eta \|y_1 - y_2\|_2}{(1-\phi)\left(\eta - \|y_1 - y_2\|_2\right)},$$

if $\eta > d = \|y_1 - y_2\|_2$. If $\eta \le d$, then $w_{1,2}^{(t-1)} = 0$ and thus $\theta_i^{(t)} = y_i$ for all $t \ge 1$.

2.8.3 Proof of Lemma 2

Let $\gamma = \theta_2 - \theta_1$. Recall that $\ell(\theta_1, \theta_2)$ is minimized at $(\bar{y} - \gamma/2, \bar{y} + \gamma/2)$ for any γ and thus, minimizing (2.3) reduces to minimizing $\ell(\gamma)$ defined in (2.4). First assume that $\gamma = c(y_2 - y_1)$ for $c \in \mathbb{R}$ and define

$$f(c) \overset{\Delta}{=} \ell(c(y_2 - y_1)) = \frac{1}{2} d^2 (c-1)^2 + \lambda \rho(d \cdot |c|),$$

where $d = \|y_2 - y_1\|_2$. It is easy to see that $f(c) > f(0)$ for all $c < 0$ and $f(c) > f(1)$ for all $c > 1$. Therefore, f must be minimized globally at $c \in [0, 1]$. Then a sufficient condition for $f(c)$ to be minimized at $c = 0$, which gives $\gamma = 0$, is

$$\frac{\partial f(c)}{\partial c} = d^2(c-1) + \left(\lambda - \frac{dc}{\delta}\right)_+ d > 0,$$

for $0 < c \le 1$. If (2.14) holds, the above inequality is equivalent to

$$\lambda > \left[(1-c) + \frac{c}{\delta}\right] d,$$

which is obviously implied by (2.14), noting that $c \in (0, 1]$. Lastly, for any nonzero γ' that is not collinear with $y_2 - y_1$, there is a $\gamma = c(y_2 - y_1)$ such that $\|\gamma\|_2 = \|\gamma'\|_2$ and $\|\gamma - (y_2 - y_1)\|_2 < \|\gamma' - (y_2 - y_1)\|_2$ by triangle inequality, and thus $\ell(\gamma) < \ell(\gamma')$. Therefore, the global minimizer of (2.3) is $(\hat{\theta}_1, \hat{\theta}_2) = (\bar{y}, \bar{y})$ if (2.14) holds.

2.8.4 Adjusted Rand Index

Let $h(k) = \binom{k}{2}$. Given a contingency table $(n_{ij})_{I \times J}$ with entries n_{ij}, row sums $n_{i\bullet} = \sum_j n_{ij}$, column sums $n_{\bullet j} = \sum_i n_{ij}$, and a total sum of entries $n = \sum_{i,j} n_{ij}$, the ARI is defined by

$$\mathrm{ARI} = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} h(n_{ij}) - \left[\sum_{i=1}^{I} h(n_{i\bullet}) \sum_{j=1}^{J} h(n_{\bullet j})\right] / h(n)}{\frac{1}{2}\left[\sum_{i=1}^{I} h(n_{i\bullet}) + \sum_{j=1}^{J} h(n_{\bullet j})\right] - \left[\sum_{i=1}^{I} h(n_{i\bullet}) \sum_{j=1}^{J} h(n_{\bullet j})\right] / h(n)}. \tag{2.24}$$

CHAPTER 3

Iterative Subsampling in Solution Path Clustering of Noisy Big Data

3.1 Introduction

In this chapter we propose a new algorithm, primarily motivated by the need to accelerate the SPC method developed in Chapter 2. SPC produces a small set of clustering solutions, called a solution path, that includes not only cluster assignments but also an estimated decreasing number of clusters along the path. SPC is based on sparsity regularization and introduces a concave penalty to a quadratic loss function, which is minimized with penalty parameters chosen by a data driven approach. The regularization on the pairwise distances between the cluster centers effectively makes it possible to achieve clustering and at the same time eliminates the need to specify the number of clusters as an input parameter. SPC is able to find compact clusters and detect noise as singleton or small clusters, due to the fact that initial clustering solutions are obtained by penalizing relatively small pairwise distances between the cluster centers. SPC, however, has time complexity $O(n^2)$ as it depends on the calculation of the pairwise distances between data points. To achieve considerable computational savings we introduce an iterative approach that enables applications of SPC to large datasets. For the new algorithm, we combine SPC, performed on a small subsample of the data, with subsequent assignment of the remaining data points based on likelihood ratio evaluation. We then iterate between the clustering and sequential assignment steps until no more valid clusters are found. The iterative subsampling SPC (ISSPC) provides orders of magnitude computational savings compared to the original SPC algorithm with relatively little loss in accuracy of the resulting clustering partition. It maintains the effectiveness of SPC to separate the noise from the clusters, eliminating the need for any prior filtering of the data. It can also be successful in locating small tight clusters in large datasets. Moreover, the new iterative subsampling approach utilizes SPC's solution path to obtain one final clustering solution with an estimated number of clusters, and in effect, performs fast and efficient solution selection.

First, we summarize the novel contributions of our work in comparison to existing subsampling-based clustering methods. First, while all of the existing methods assume that the number of clusters is given or require that some initial estimate of the number of clusters is provided by the user [FS96, Mai01, NDR14, BR93, FR02, WBF04, FRW05], ISSPC determines the number of clusters on its own through significance tests on a solution path. This improves usability and at the same time reduces user bias. Second, both the clustering step and the assignment step in our approach are designed under the assumption that the full dataset may contain a certain proportion of noisy data points. In each iteration, only a portion of the data points are partitioned into clusters, and the remaining data points will be considered in the subsequent iterations until no more clusters are identified. On the contrary, all but one of the methods reviewed above partition the entire dataset into clusters generated from the subsample. The performance on noisy data has been demonstrated solely in [WBF04] with data containing only 5% of noise, while we show results for varying noise proportions up to 90%. Third, as most clustering algorithms including SPC have time complexity of $O(n^2)$, the subsample size we consider here is $O(\sqrt{n})$, which is much smaller than the subsample sizes used in the previous work [BR93, FR02, WBF04, FRW05, KMC10, NDR14]. This is important in the context of big data applications and inherently large datasets, for which only algorithms with $O(n)$ operations would be computationally feasible. We study systematically the performance loss and computational savings by varying the subsample size as $a\sqrt{n}$ for $a \in [1, 10]$.

SPC algorithm is initialized assuming that all data points form singleton clusters. It then gradually builds up the sparsity in $\|\theta_i - \theta_j\|_2$ and merges the cluster centers by penalizing increasingly larger distances between them. As an output, SPC provides a path of clustering solutions with a decreasing number of clusters, each solution consisting of cluster assignments for all y_i and a number of clusters. Index the solutions on a path of size S by $s = 1, \ldots, S$ and denote the cluster assignment by $A_s = A_s(Y)$ and the number of clusters by $\hat{K}^s$. The number of total clusters $\hat{K}^s$ consists of clusters of any size, including singleton clusters, and $\hat{K}^s \geq \hat{K}^{s+1}$ for all s. The solution path $\mathcal{A}(Y) = \{A_1, \ldots, A_S\}$ is generated using an adaptive data-driven approach by automatically selecting a combination of the penalty parameters $(\lambda, \delta)_s$ for each solution. The choice of the combination of $(\lambda, \delta)_s$ is based on the properties of the MCP and is guided by the pairwise distances between center parameters θ_i's.

The noisy data in each solution A_s can be identified as singleton or very small clusters, i.e. clusters of size $N_k \leq n_0$. Our default choice is $n_0 = 3$; however, a much larger cutoff cluster size can be selected based on some knowledge about the data. Another special characteristic of SPC is that it can find tight clusters in the initial solutions, similar to hierarchical clustering, when a clustering tree is cut at smaller distances or at a relatively large number of clusters. Finally, SPC is very easy to use as it effectively has only one tuning parameter that determines the approximate proportion of nearest neighbors to be merged for the initial solution. These characteristics of SPC coupled with the small solution path allow us to develop an iterative subsampling algorithm that can handle large noisy data and can select a single clustering solution.

Lastly, integrating SPC, a regularization clustering method, with subsampling raises a number of new challenges, such as the choice of tuning parameters in the concave penalty and model selection along the regularization path. This work provides practical solutions to these problems with satisfactory performance on both simulated and real data.

The remainder of this chapter is organized as follows. In Section 3.2.1 we describe the sequential assignment based on likelihood ratio calculation. The full algorithm is presented in Section 3.2.2. Section 3.2.3 discusses some practical considerations for choosing tuning parameters. We demonstrate the performance of ISSPC on simulated data in Section 3.3 and on gene expression data in Section 3.4. The chapter is then concluded with a brief discussion.

3.2 Methods

3.2.1 Sequential cluster assignment

In this section we describe the sequential cluster assignment procedure that can be applied to noisy data and that is thereafter combined with SPC for the full ISSPC algorithm. The sequential cluster assignment can be directly connected to classification, specifically to discriminant analysis as it is based on evaluating the likelihood ratios for new data points to determine their cluster memberships. We assume that a dataset can generally be separated into data points showing a grouping pattern and data points without any grouping pattern, i.e. noise. We introduce a background model M_0 under which a noise data point follows $\mathcal{N}_p(\mu_0, \Sigma_0)$, while clustered data points are modeled by a mixture Gaussian distribution $M_\mathcal{C}$. In short, we apply SPC on a subsample to estimate parameters for these models and then classify the remaining data points to obtain their cluster memberships. In this sense, the role of the subsample is similar to that of training data and the remaining data similar to test data. So we may also call the two subsets training and test data, respectively.

Suppose that we have drawn without replacement a subsample $\mathscr{D}$ (training data) from Y and denote the remaining data by $\mathscr{T}$ (test data). Let $D, T \subset \{1, \ldots, n\}$ be the indices for data points in $\mathscr{D}$ and $\mathscr{T}$. The SPC method is applied to $\mathscr{D}$ to obtain a clustering path $\mathcal{A}(\mathscr{D})$ from which a cluster assignment is selected. Write the selected clustering assignment as $C^t = \{\mathcal{C}, C_0\}$, where $\mathcal{C} = \{C_1, \ldots, C_K\}$ denotes the clusters and C_0 contains the indices of the data points identified as noise. Assuming that y_i follows a mixture Gaussian distribution $M_{\mathcal{C}}$, its likelihood is

$$\begin{aligned} & L\left(y_i | M_{\mathcal{C}}\right) \qquad (3.1) \\ =& \sum_{k=1}^{K} \frac{\pi_k}{\left|2\pi\Sigma_k\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(y_i - \mu_k\right)^T \Sigma_k^{-1}\left(y_i - \mu_k\right)\right\} \\ \triangleq& \sum_{k=1}^{K} L_k(y_i | M_k), \end{aligned}$$

with μ_k denoting the mean, $\Sigma_k = \mathrm{diag}(\sigma_{k1}^2, \ldots, \sigma_{kp}^2)$ – a diagonal covariance matrix, and the mixture proportions $\sum_{k=1}^{K} \pi_k = 1$. Let $N_k = |C_k|$ be the size of a cluster. Based on the clustering assignment $\mathcal{C}$ of the training data, we estimate μ_k by the cluster sample mean $\bar{y}_k = \frac{1}{N_k}\sum_{i \in C_k} y_i$ and Σ_k by cluster sample variances $S_k = \mathrm{diag}(s_{k1}^2, \ldots, s_{kp}^2)$, where

$$s_{km}^2 = \frac{1}{N_k - 1}\sum_{i \in C_k}(y_{im} - \bar{y}_{km})^2, \qquad (3.2)$$

and the mixture proportions $\hat{\pi}_k = N_k / \sum_{j=1}^{K} N_j$. On the other hand, the background model M_0 is estimated by the overall mean and variance of all the data points Y, i.e., $\hat{\mu}_0 = \bar{y} = \frac{1}{n}\sum_i y_i$ and $\hat{\Sigma}_0 = S_0 = \mathrm{diag}(s_{01}^2, \ldots, s_{0p}^2)$ with

$$s_{0m}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_{im} - \hat{\mu}_{0m})^2. \qquad (3.3)$$

To simplify notation, we use $\hat{M}_0$ and $\hat{M}_{\mathcal{C}}$ ($\hat{M}_k$) to denote the two models with the estimated parameters.

We then classify the test data $\mathscr{T}$ sequentially based on the estimated models $\hat{M}_0$ and $\hat{M}_{\mathcal{C}}$. We follow a simple decision rule that assigns a test data point y_i, $i \in T$, to the more likely model. The likelihood ratio for each y_i, $i \in T$, is

$$\Lambda(y_i) = \frac{L(y_i|\hat{M}_{\mathcal{C}})}{L(y_i|\hat{M}_0)} = \frac{\sum_{k=1}^{K} L_k(y_i|\hat{M}_k)}{L(y_i|\hat{M}_0)}. \tag{3.4}$$

Let $G(y_i) \in \{0, \ldots, K\}$ be the cluster membership for y_i. Then the assignment rule is

$$G(y_i) = \begin{cases} \underset{1\leq k\leq K}{\operatorname{argmax}} L_k(y_i|\hat{M}_k) & \text{if} \quad \Lambda(y_i) \geq c \\ 0 & \text{if} \quad \Lambda(y_i) < c \end{cases}, \tag{3.5}$$

where by default $c = 1$. The threshold c can also take values other than 1 depending on the application; however, we assume $c = 1$ for all the simulation and real data studies. Note that y_i is identified as noise if $G(y_i) = 0$. Once y_i is assigned to a cluster C_k, the estimated parameters $\hat{\pi}_k$, $\bar{y}_k$ and S_k are updated for the calculation of (3.4) for the next test data point.

The above assignment rule can be regarded as a mixture discriminant analysis. Mixture discriminant analysis [HT96] is a generalization of linear and quadratic discriminant analysis, which was further generalized and extended to the models with varying covariance matrices [FR02]. In our case, we have a mixture discriminant model with two classes, where one of the classes, the cluster model $M_{\mathcal{C}}$, is a Gaussian mixture model with K components.

3.2.2 Iterative subsampling

We now show how sequential cluster assignment can be combined with SPC to produce accurate clustering solutions for large datasets that otherwise would be computationally expensive or prohibitive. Given a large dataset Y, we choose the subsample size $\nu = |\mathscr{D}| = a\sqrt{n}$ where $a \geq 1$ is a small scalar, so that the computational complexity of both the clustering and the sequential assignment steps is $O(n)$. However, such a small subsample will most probably not be able to capture all the clustering structure of the full data Y. Thus, we introduce a recursion between the clustering and sequential assignment steps. Let Y_0 be the data points identified as noise by either the clustering or the sequential steps after the current iteration. That is, the index set of data points in Y_0 is $C_0 \cup \{i \in T : G(y_i) = 0\}$. SPC is then repeated for a random sample of the same size ν taken from Y_0, followed by another sequential assignment step. This recursion will be repeated until no more clusters are found. Since SPC does not require the number of clusters as an input parameter and provides a short solution path, it is necessary to select one clustering solution for the sequential step in the combined algorithm. Instead of relying on a sophisticated model selection procedure, we consider any solution on the path as a set of potential clusters $\mathcal{C}$. We test whether these potential clusters are significantly different from the null model M_0, which is characterized by a large variance since it is estimated with all the data points, including noise. If some clusters in $\mathcal{C}$ are not significantly different from the null, then the data points in these clusters are reassigned to C_0. This quality check becomes important for later stages in the clustering-assignment recursion since the clustering is assumed to be performed on an increasing amount of noise. It also becomes a stopping criterion for the recursion: the recursion is terminated when no more clusters are found to be significantly different from the null.

Now we describe the detailed testing procedure. For a particular solution, we compare each estimated covariance matrix S_k, $k = 1, \ldots, K$, to the estimated null covariance matrix S_0. Since we have assumed that both Σ_k and Σ_0 are diagonal, a set of p hypothesis tests can be performed for each cluster k:

$$H_{km}^0 : \sigma_{km}^2 = \sigma_{0m}^2 \quad \text{vs} \quad H_{km}^1 : \sigma_{km}^2 < \sigma_{0m}^2,\ m = 1, \ldots, p,$$

where we reject H_{km}^0 if s_{km}^2 (3.2) is sufficiently large relative to s_{0m}^2 (3.3). Since by design $n \gg \nu > N_k$ for all k, the variance of s_{0m}^2 is negligible compared to s_{km}^2

and thus may be treated as a constant. Then the test statistic

$$F_{km} = \frac{(N_k - 1)s_{km}^2}{s_{0m}^2}, \quad k = 1, \dots, K \text{ and } m = 1, \dots, p, \tag{3.6}$$

follows a χ^2-distribution with $N_k - 1$ degrees of freedom if H_{km}^0 is true. Next, we sort the p-values P_{km} of the statistic F_{km} in ascending order, $P_{k(m)} \leq P_{k(m+1)}$, and apply the Benjamini-Hochberg procedure to find

$$m_k^* = \max\left\{ m : P_{k(m)} \leq \frac{m}{p}\beta, m = 1, \dots, p \right\} \tag{3.7}$$

for each k, where β is the cutoff of the false discovery rate (FDR). We keep cluster C_k if $m_k^* \geq \eta$ with $\eta \in (1, p)$, i.e. if we find sufficiently many dimensions with significantly smaller s_{km}^2 compared to the corresponding s_{0m}^2. Otherwise, we discard cluster C_k and reassign its members to C_0. The procedure of controlling the FDR for each cluster allows us to account for the relevancy of a subset of dimensions or features in each cluster, which becomes more critical for high-dimensional data. In addition, this procedure will generally reject very small clusters with moderately small variances, and thus, it can also be seen as a way of controlling for a minimum cluster size. Finally, if all the clusters are discarded after the SPC step for a subsample, then the clustering-assignment recursion is terminated.

The hypothesis testing coupled with FDR control effectively becomes a practical mechanism for an automatic determination of the number of clusters in a dataset. In fact, we do not need to use any sophisticated solution selection methodology in order to pick a solution from $\mathcal{A}(\mathscr{D})$, but we would simply select the first solution with the largest number of clusters of size $N_k > n_0$, followed by the above testing procedure to remove loose clusters. This gives us a higher chance of discovering tight clusters in presence of noise. The effectiveness of this approach will be demonstrated in Sections 3.3 and 3.4.

The full ISSPC method is outlined in Algorithm 3, in which b is the iteration number in the clustering-assignment recursion. The collection of clusters generated in the bth iteration is denoted by $\mathcal{C}^{(b)}$, while the noise data points are denoted

Algorithm 3 Iterative Subsampling SPC (ISSPC)

```
**Inputs**
input: $Y = (y_{im})_{n \times p}$, $\omega^{(1)} \in (0,1)$, $\nu$, $\eta$
default input: $\omega^{(b)} = 0.1$ for $b \geq 2$, $\beta = 0.01$
initialization: $b = 1$, $Y_0^{(1)} = Y$, $\mathcal{C} = \emptyset$

1: repeat
2:     $\mathcal{C}^{(b)} = \emptyset$
3:     draw a random sample $\mathscr{D}^{(b)}$ of size $\nu$ from $Y_0^{(b)}$
4:     $\mathscr{T}^{(b)} = Y_0^{(b)} \setminus \mathscr{D}^{(b)}$
5:     run SPC Algorithm 2 to obtain a solution path $\mathcal{A}(\mathscr{D}^{(b)})$
6:     choose $A_t = \{C_0, C_1, \ldots, C_K\} \in \mathcal{A}(\mathscr{D}^{(b)})$ such that $K$ is maximized.
7:     for $k = 1, \ldots, K$ do
8:         compute $m_k^*$ as in (3.7)
9:         if $m_k^* \geq \eta$, $\mathcal{C}^{(b)} \leftarrow \mathcal{C}^{(b)} \cup \{C_k\}$
10:        else $C_0 \leftarrow C_0 \cup C_k$
11:    end for
12:    if $K = 0$ or $m_k^* < \eta$ for all $k$ then
13:        break
14:    end if
15:    for all $y_i \in \mathscr{T}^{(b)}$ do
16:        compute $G(y_i)$ as in (3.5)
17:        assign $y_i$ to $C_0$ or the clusters in $\mathcal{C}^{(b)}$ accordingly
18:    end for
19:    $Y_0^{(b+1)} = C_0$, $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}^{(b)}$, $b \leftarrow b + 1$
20: until $|Y_0^{(b)}| < \nu$
```

by $Y_0^{(b)}$. The set $\mathcal{C}$ consists of all the clusters found along the recursion. Note that $\omega \in (0,1)$ is an input parameter for the SPC algorithm.

3.2.3 The choice of the tuning parameters

Along with the tuning parameter ω of the original SPC, the ISSPC has additional user-defined parameters. These include β, the FDR cutoff, and η, the number of dimensions with FDR $< \beta$ that determines whether a cluster should be kept or removed. The choice of these parameters is relatively straightforward, especially for β, which does not have a big impact on the procedure as long as its value is reasonably low, e.g. $\beta \in (0.1, 0.01)$. Throughout the simulated data and real data examples we set $\beta = 0.01$, which is commonly used in practice for hypothesis testing. High values of β can result in the acceptance of "false" clusters, i.e. clusters consisting mostly or entirely of noise, and the subsequent erroneous assignment of clustered data into these clusters along with noise. The parameter η can be set based on user knowledge of dimension relevance, but we generally recommend to choose $\eta \in (0.1p, 0.5p)$. If $\eta \approx p$, many relevant clusters could be omitted, and if $\eta \approx 1$, too many "false" clusters can be accepted since low variances can occur by chance in just a few dimensions. It is clear that the parameter η can also affect the number of clusters, with smaller values resulting in more clusters.

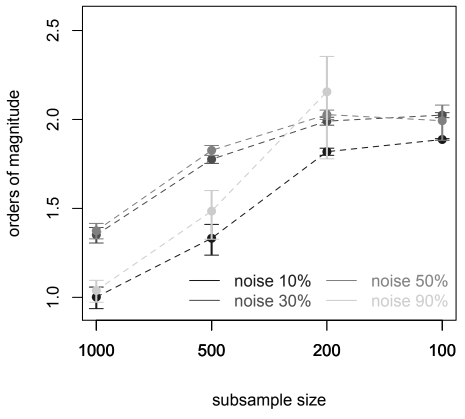
It is important to point out that SPC itself and the sequential assignment rely on the assumption that the clusters are generated by a mixture Gaussian distribution. If this assumption is violated, the clustering result might be unsatisfactory, especially considering the fact that the subsample sizes for clustering are very small. A typical example of such an adverse scenario is the presence of several outliers in the SPC clusters, which can trigger the sequential assignment step to incorporate noise or misclassify clustered data. To overcome such problems we may calculate (3.4) by using robust estimates of μ_k and Σ_k, such as the trimmed mean and variance. Trimming 5% to 25% of data points is usually an acceptable amount. If α% of data points are removed in trimming, then $(\alpha/2)$% of the smallest and $(\alpha/2)$% of the largest values in each dimension are discarded.

We perform trimming on clusters with a relatively substantial size; in particular we set this size to be $N_k > 10n_0$ to avoid trimming very small clusters that can lead to unstable estimates of means and variances. Generally, more clusters with a smaller size will be discovered with increasing amounts of trimming. This is due to the fact that fewer test data points will be added to the existing trimmed clusters, and these omitted data will most probably be clustered in the next iteration of the subsample clustering. Since the same amount of trimming is done across all of the clusters with a larger size, it is possible that, while very tight patterns can be extracted, some larger clusters could be split as a result.

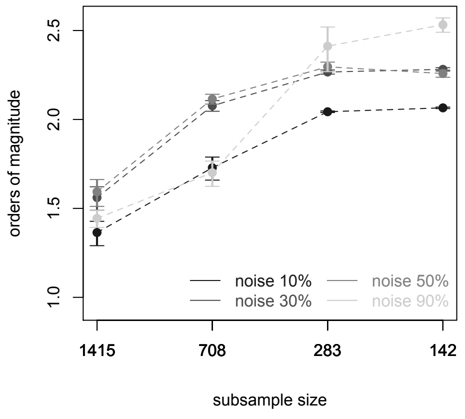
To conclude, we would like to comment on the choice of the original SPC tuning parameter ω. This parameter can be determined by the user for the first iteration of Algorithm 3; however, we generally recommend to set $\omega = 0.1$ or a comparably small value for $b \geq 2$ iterations of the clustering and sequential steps. Low values of ω will ensure that smaller clusters with fewer outliers are discovered in later iterations and the procedure does not terminate prematurely.

3.3 Simulation studies

In this section we demonstrate the performance of ISSPC on simulated data. We generate large datasets and cluster these data with three methods: the original SPC algorithm, ISSPC and the mclust method in the R package `mclust` [FR02]. We chose mclust based on its ability to produce high quality results, to separate noisy data points and to estimate the number of clusters, as well as its competitive speed as shown in Chapter 2. We compare the accuracy and the speed of both SPC approaches and mclust. The accuracy is evaluated based on the adjusted rand index (ARI) developed in [HA85], which compares an estimated cluster assignment to the true cluster assignment. We calculate two different ARI scores ($\text{ARI}_c, \text{ARI}_n$) as stated in Section 2.4.1 to gauge the performance with respect to the clustered



(a) $n = 10,000$



(b) $n = 20,000$

Figure 3.1: The orders of magnitude of computational savings of the ISSPC algorithm compared to the original SPC algorithm. The bars indicate 95% confidence intervals. The subsample sizes are $a\sqrt{n}$ for $a = 10, 5, 2, 1$ from left to right.

data and noise. ARI$_c$ assesses whether data points are misclassified, whether identified clusters are merged or split, and whether the estimated clusters contain noise. ARI$_n$ evaluates whether any clustered data point is misclassified as noise.

We performed the comparison on four different data sizes $n = 10,000$, $n = 20,000$, $n = 50,000$, and $n = 100,000$ and $p = 20$ dimensions, and generated each dataset with four different proportions of noise – $0.1n$, $0.3n$, $0.5n$, and $0.9n$. The clustered data in each of a total of 16 datasets were generated from a Gaussian mixture model with $K = 10$ clusters of about equal size and variance. The noise was generated from a uniform distribution on $[-5, 5]^p$ outside the radiuses of the clusters, where the radius of a cluster is defined as the largest distance from the cluster center to the data points in that cluster. The ISSPC algorithm was run 20 times independently on each dataset and the ARI scores and the computation speed are reported as an average of the 20 runs, with each run using different subsamples. We chose $\eta = 10$ since all the dimensions were relevant for all the simulated clusters. We were not able to apply the original SPC algorithm to the datasets of size $n > 20,000$ due to operating memory limitations in holding large distance matrices, and so we present the comparison results only with $n = 10,000$ and $n = 20,000$. The results from $n = 50,000$ and $n = 100,000$ are presented to demonstrate the accuracy, stability and potential time savings of the ISSPC approach.

Mclust, when applied to data with noise, has in effect three inputs that include the categorization of data into noise and clusters, a range of the number of clusters for model selection via Bayesian Information Criterion (BIC), and the reciprocal hyper volume V of the data region. For more details on the tuning parameters of mclust, please refer to [FR02] and Section 2.4.1. The recommended Kth nearest neighbor cleaning method was applied for the categorization [BR98] with $K = 10$ and the range of the number of clusters was set to $(5, 15)$. We could not use the default value for V, which appeared to be too small and forced all the noise into the clusters. Instead we estimated a larger V using the approximate volume of the convex hull of the data, which was computed with the `geometry` package in R based on six dimensions and extrapolated roughly to 20 dimensions.

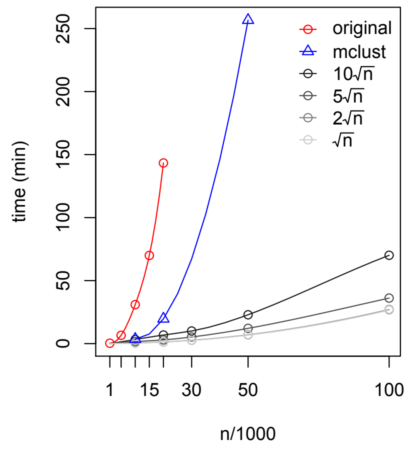
3.3.1 Computation time

First, we show computational savings of the new approach compared to the original algorithm for $n = 10,000$ in Figure 3.1a and for $n = 20,000$ in Figure 3.1b. The curves show the average orders of magnitude of computational savings compared to the original SPC algorithm. The different gray scales in the plots represent the different proportions of noise in the datasets. It can be seen from Figure 3.1 that computational improvements of this procedure are considerable and can amount to somewhere between 1 to 2 orders of magnitude (10 to 100 times faster) for $n = 10,000$ and even larger, 1.5 to 2.5 orders of magnitude, for $n = 20,000$. For the dataset with 90% of noise and of size $n = 10,000$, ISSPC with a subsample size of $\sqrt{n} = 100$ did not find any clusters, and thus the computation time for this case is not reported in Figure 3.1a.

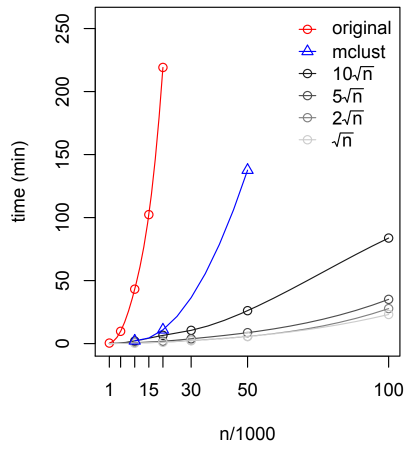
Figure 3.2 shows the comparison of the absolute run times for all the compared clustering methods across all the simulated datasets. The run time for mclust includes the Kth nearest neighbor procedure that estimates the noisy data points. It can be seen from the figure that the run time of mclust depends on the amount of noise and is much shorter when this amount is large due to the fact that mclust is effectively applied only to the identified clustered data. It should be noted that mclust could not be run on $n = 100,000$, due to reaching operating memory limits. The original SPC algorithm is shown to have a computational time complexity of $O(n^2)$ approximately for all the simulated data scenarios with a slightly shorter time when there is little noise. Mclust unsurprisingly shows much better speed than the original SPC algorithm. ISSPC, however, outperforms mclust on speed, especially with larger data sizes. ISSPC is able to cluster a large dataset, e.g. $n = 100,000$, in a reasonable amount of time, even when the subsample size is $\nu = 10\sqrt{n}$, and it can favorably compete with mclust in terms of speed, especially considering the fact that ISSPC is currently implemented in R, while mclust is implemented in Fortran.

3.3.2 Accuracy

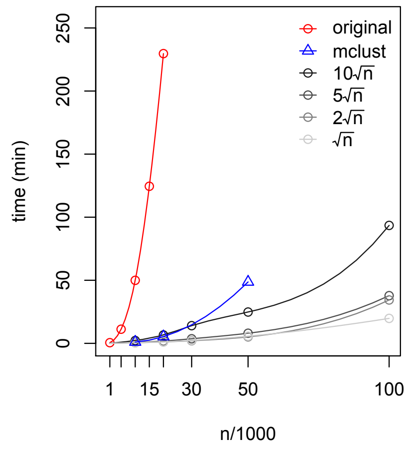
We now report the comparative performance of original SPC, ISSPC and mclust. Figures 3.3a-3.3d demonstrate ARI_c (top panel), ARI_n (middle panel) and the estimated number of clusters (bottom panel) for each data size n. The boxplots for ISSPC show the results across 20 independent runs on the same dataset, while only one (deterministic) result is reported for each dataset for SPC and mclust. For each subsample size, increasing amounts of noise in the data are represented by a lighter gray scale. The results for the original SPC algorithm are reported after the hypothesis testing and cluster selection procedure, described in Section 3.2.2,



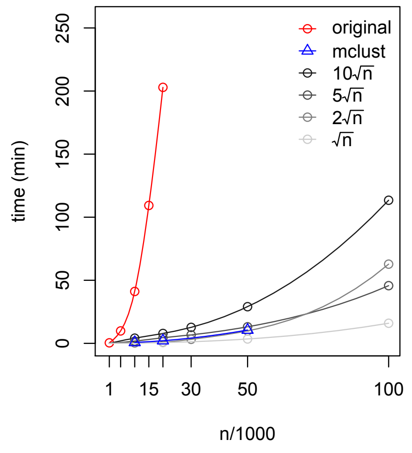
(a) noise $0.1n$



(b) noise $0.3n$



(c) noise $0.5n$



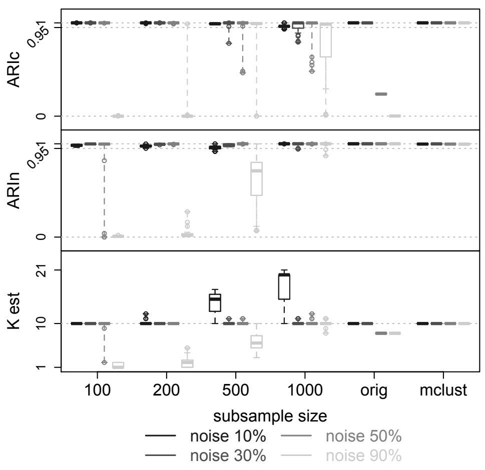
(d) noise $0.9n$

Figure 3.2: The run times in minutes for the original SPC (red), mclust (blue) and ISSPC (gray scale). Each panel corresponds to a different noise proportion in the datasets.

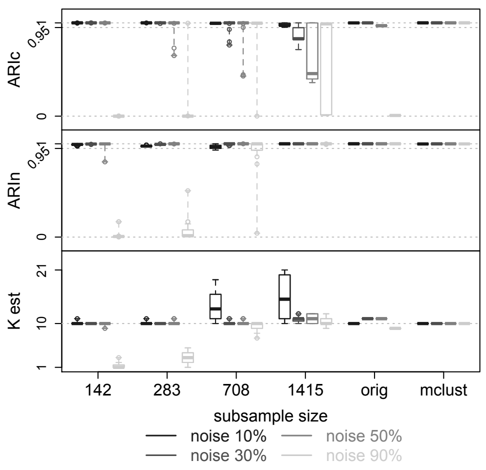
is applied to the solution with the largest number of clusters.

Overall, in these examples ISSPC shows very strong results when the noise proportion is no more than 50%, especially considering the small sizes of the subsamples. These results are quite competitive compared to the original SPC and mclust, even though somewhat inferior as expected. In fact, the results for smaller subsamples, $\nu = \sqrt{n}$ and $2\sqrt{n}$, have high and stable ARI scores across all the scenarios. There are several outliers in the ARI scores, especially when there is a larger amount of noise, indicating that the quality of the clustering result may depend on a particular random subsample. ARI_c scores for larger subsamples, $\nu = 5\sqrt{n}$ and $10\sqrt{n}$, become more volatile, and the clustering results vary more in their quality and are generally somewhat inferior in ARI_c, compared to those with the smaller subsamples. There are at least two reasons for this observation. First, the SPC solution for a larger subsample tends to have a higher variance. Second, in later iterations of ISSPC, the use of larger subsamples increases the risk of creating, relatively large clusters that consist mostly or exclusively of noise but are accepted by the cluster selection procedure due to their substantial sizes. Thus, a smaller subsample may be beneficial not only for computational savings but also for the quality of the clustering result. Furthermore, it is seen that ISSPC does not show consistently good performance with the highest proportion of noise 90%. However, for a suitable subsample size somewhere between 600 and 1,600, it can give a very satisfactory result, close to that of mclust, for such amount of noise, as seen from the ARI scores and the estimated number of clusters in Figure 3.3 (the lightest gray color). With a relatively small subsample size, ISSPC was able to provide a fairly accurate estimated number of clusters for all the scenarios in Figure 3.3, with the exception of the cases with the highest proportion of noise, which were discussed above. However, ISSPC tends to overestimate the number of clusters when the subsample size is larger as a result of splitting clusters.

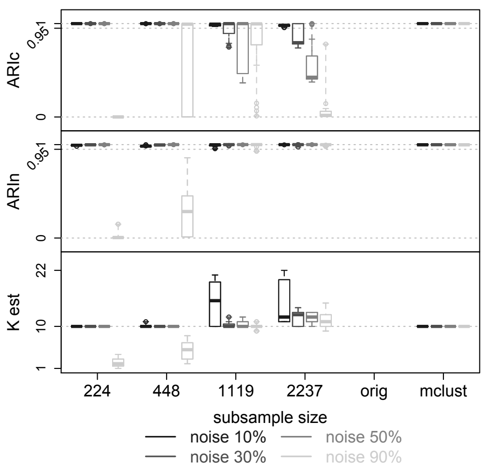
In summary, we see competitive performance of ISSPC with relatively small



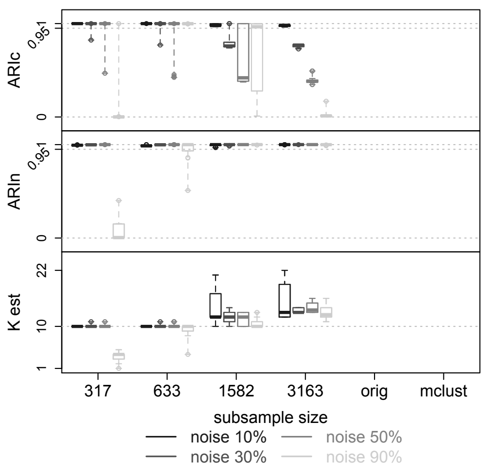
(a) $n = 10,000$



(b) $n = 20,000$



(c) $n = 50,000$



(d) $n = 100,000$

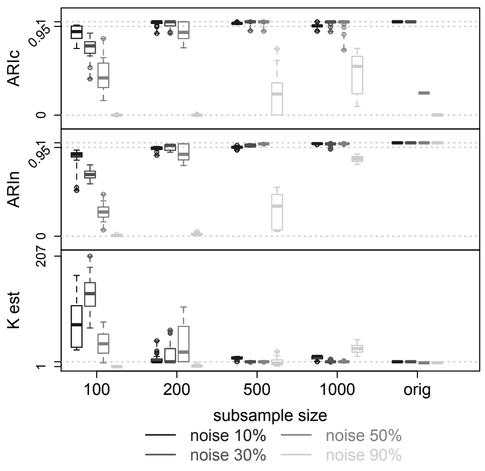
Figure 3.3: ARI scores and the number of estimated clusters for ISSPC, original SPC and mclust for various dataset sizes. From left to right the subsample sizes are $\sqrt{n}$, $2\sqrt{n}$, $5\sqrt{n}$, and $10\sqrt{n}$.

subsample sizes, compared to the results of SPC and mclust, as long as the noise proportion in the data is not too high ($\leq 50\%$). On the other hand, for such small subsample sizes, ISSPC is up to two orders of magnitude faster as shown in the previous subsection.

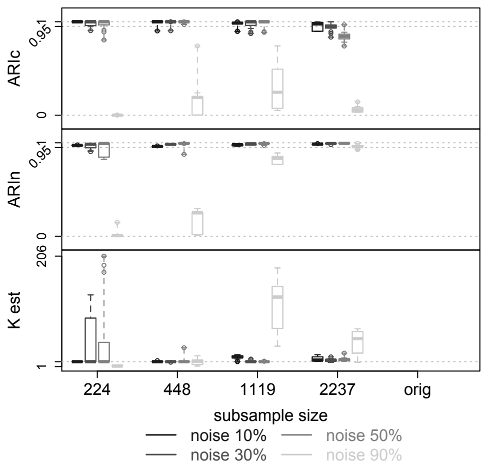
3.3.3 Trimming

As mentioned in Section 3.2.3, trimmed estimates of cluster means μ_k's and variances Σ_k's can be useful if the clusters significantly deviate from the spherical shape, are not well separated, or generally if data are complex and high-dimensional. These situations are certainly typical for real data. Our simulated data have a simplified structure; however, to understand its potential impact, we have applied 10% trimming to the simulated data with $n = 10,000$ and $n = 50,000$.

Figure 3.4 shows boxplots of ARI scores in the top and middle panels and the estimated number of clusters in the bottom panel for both sizes. When trimming is applied to the simulated data, the clustering results are inferior to the results in Figure 3.3 for smaller subsamples. It is mainly due to the fact that with trimming, the clusters tend to be split, especially for clusters of a smaller size. Specifically, trimming tends to remove valid clustered data points for the calculation of the centers and variances and forces the likelihood ratios in the sequential assignment to be too small, which results in some clustered data points being assigned to the background model. These data then create their own clusters in the subsequent clustering and sequential assignment steps. This problem is particularly severe for smaller subsample sizes, in which cases the clusters generated by the clustering steps are necessarily small. As the subsample size increases or the size of the clusters becomes larger, as in Figure 3.4b, the results improve considerably. It should also be mentioned that trimming in fact improves the results for larger subsample sizes, $\nu > 1000$, compared to those without trimming in Figures 3.3a



(a) $n = 10,000$



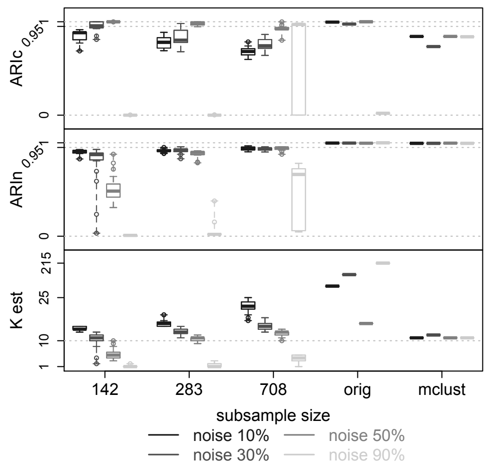
(b) $n = 50,000$

Figure 3.4: ARI scores and the number of estimated clusters for 10% mean and variance trimming.

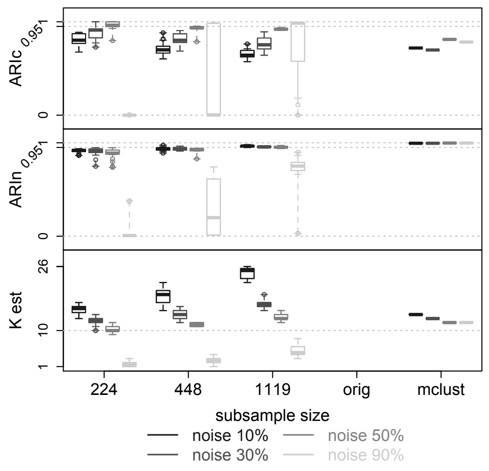
and 3.3c. Generally, greater amounts of trimming will produce a larger number of smaller clusters. While unfavorable for some of the simulated data scenarios, trimming can be beneficial for obtaining compact, small clusters in large datasets, which we will show in Section 3.4.

3.3.4 Non-spherical clusters

Finally, we demonstrate the performance of ISSPC on data with non-spherical clusters. For this purpose we generated $K = 10$ clusters with $p = 20$ correlated dimensions for $n = 20,000$ and $n = 50,000$. Four out of 10 clusters were generated with 0.5 correlation, and the rest of the clusters had a correlation of 0.3. The four clusters with the highest correlation were also the largest clusters with sizes $\approx (0.3n, 0.2n, 0.1n, 0.1n)$, and the remaining six clusters were smaller and of size $\approx 0.05n$. The noisy data points were added in the same way as in the spherical data scenarios and in the same proportions. As previously, we ran ISSPC 20 times independently on each non-spherical scenario and report the average ARI scores



(a) $n = 20,000$



(b) $n = 50,000$

Figure 3.5: ARI scores and the number of estimated clusters for non-spherical simulated clusters. The subsample sizes are $\sqrt{n}$, $2\sqrt{n}$, and $5\sqrt{n}$.

and their distributions in boxplots in Figure 3.5. We again include the comparison with the original SPC algorithm and mclust.

Overall, ISSPC gives a satisfactory result with non-spherical clusters, expect for the cases with 90% of noise, which still prove to be challenging. As can be expected, the non-spherical clusters tend to be split into smaller clusters. ARI$_c$ and the estimated number of clusters in Figure 3.5 indicate that the splitting creates several very small clusters; however, the majority of the "true" clusters is preserved correctly, albeit with a very small amount of misclassified noise. It can also be seen from the figure that mclust's results are quite comparable to those of ISSPC. Particularly, the ARI$_c$ scores of the two methods are close for the datasets with less than 90% of noise. Mclust recognizes noisy data points with slightly higher accuracy, reflected by slightly better ARI$_n$ scores, mainly due to the Kth nearest neighbor cleaning method, but the largest clusters are also split as seen from ARI$_c$ scores and the estimated number of clusters. We believe that ISSPC is useful for data with non-spherical clusters or data that violate Gaussian assumptions as it can provide a competitive result with significant computational savings, and it is able to deliver this result for large datasets that otherwise would not be tractable for regular full data approaches such as mclust.

3.4 Gene expression data

We applied ISSPC to two gene expression datasets. We first analyzed a mouse embryonic stem (ES) cell dataset [ZCM07] (Zhou dataset), which was analyzed previously in Section 2.6 with the original SPC algorithm. The full dataset consists of about 45,000 genes across 16 experimental conditions; however, we used only a subset of this dataset, obtained in the same way as in Section 2.6. The subset consists of $n = 5,765$ genes in $K = 2$ large distinct clusters ($1,325$ genes in the Oct4+ and $1,440$ in the Oct4$-$ clusters) with $3,000$ randomly selected genes whose profiles were perturbed such that these genes can be considered noise. This abridged version of the Zhou dataset is studied here as it combines the complexity of real data with a simplified noise structure. The distinct grouping patterns and the availability of information about the involved clustered genes make the clustering result easier to evaluate. The second dataset [IDL06] (Ivanova dataset) consists of 45,264 gene expression profiles generated under different treatments in mouse ES cells across 70 experimental conditions. While gauging the behavior of ISSPC on clean, noncomplex data with the first dataset, we chose the second dataset for testing its ability to handle a full, noisy gene expression dataset.

3.4.1 Results for the Zhou dataset

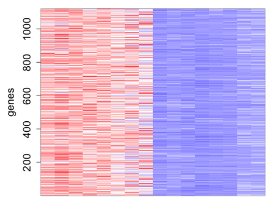
The Zhou dataset was run with $\eta = 5$ and the subsample size $\nu = 5\sqrt{n} = 380$ as the full data size was relatively small. If ISSPC is applied to the Zhou data without the trimming as per Section 3.2.3, then only the two largest Oct4+ and Oct4$-$ clusters are found with just 4 random genes incorrectly included in these

Table 3.1: Top GO terms enriched in the small cluster shown in Figure 3.6d

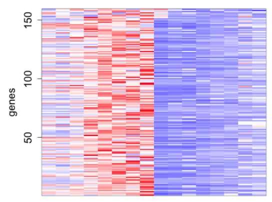
GO term	% in cluster	% in full set	P-value	FDR(%)
olfactory bulb development	3.7	0.5	1.2×10^{-5}	0.0
olfactory lobe development	3.7	0.5	1.2×10^{-5}	0.0
negative regulation of Wnt signaling pathway	6.2	1.5	4.7×10^{-5}	2.7
embryonic organ morphogenesis	8.7	3.2	2.7×10^{-4}	7.0
central nervous system development	13.0	6.2	4.6×10^{-4}	7.2
regulation of Wnt signaling pathway	7.5	2.7	6.9×10^{-4}	10.0

clusters and 174 Oct4 genes erroneously excluded as random. We then applied 10% trimming and obtained five clusters with distinct grouping patterns as shown in Figure 3.6 in under a minute of run time. The two largest clusters in Figure 3.6a and Figure 3.6c recover the large Oct4+ and Oct4− groups, respectively. There were 10 misclassified random genes and 138 misclassified Oct4 genes in these two clusters. The other three clusters (Figures 3.6b, 3.6d, and 3.6e) are relatively small and are seen to have some subtle differences in expression patterns from the two big clusters. To check whether these clusters are functionally distinct from the Oct4+ and Oct− groups, we performed Gene Ontology (GO) term enrichment analysis and compared the three small clusters to the full set of Oct4+ or Oct4− genes. We found that the small cluster in Figure 3.6d had several significant terms with an FDR between 0% and 10% (Table 3.1), confirming that genes in this cluster are indeed involved in distinct biological processes.

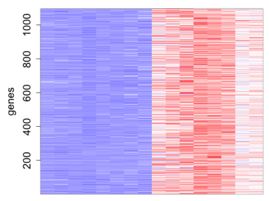
Although the two clusters in Figures 3.6b and 3.6e did not have any GO terms with FDR $< 10\%$, it can be clearly seen from the figures that they both have distinct expression patterns from the other clusters. Compared to all the other



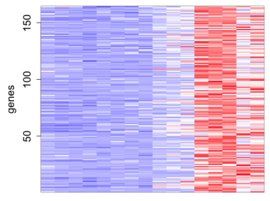
(a) 1, 120 genes



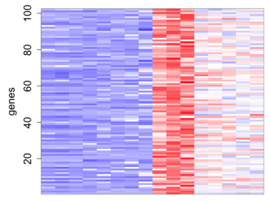
(b) 159 genes



(c) 1, 095 genes



(d) 164 genes



(e) 102 genes

Figure 3.6: Five clusters obtained from the Zhou dataset. The caption of each plot indicates the size of the cluster. The experimental conditions 1-16 are located along the x-axis. Blue color indicates low expression and red indicates high expression.

clusters, the cluster in Figure 3.6b has much lower expression levels in the first three conditions and somewhat higher levels in conditions 6-8 compared to the large Oct4+ cluster in Figure 3.6a. The cluster in Figure 3.6e has a particularly high expression pattern across conditions 9-11. Detecting such heterogeneity may be useful for novel findings from a big dataset.

3.4.2 Results for the Ivanova dataset

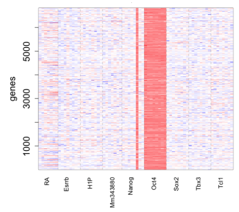
The gene expression data in [IDL06] were generated under retinoid acid (RA) induction, a control condition, and the knockdown experiments of seven transcription factors (Oct4, Nanog, Sox2, Esrrb, Tbx3, Tcl1 and Mm343880) over approximately eight days to explore the mechanisms of self-renewal and differentiation of mouse ES cells. The study in [IDL06] clustered 3,109 genes, and another study in [MFP09] clustered about 17,000 genes from this dataset. We attempt to cluster the full dataset of 45,264 genes without any filtering to identify more potential patterns. The ISSPC algorithm was applied with $\eta = 15$, 20% trimming, and subsample size $\nu = 2\sqrt{n} = 425$. We obtained 13 clusters of various sizes, depicted in Figure 3.7, and identified a total of about 15,000 genes as noise. Hereafter, we characterize a cluster from this dataset as condition-high or condition-low. For example, if a cluster is described as Oct4-high, then it means that genes in this cluster have high expression in the Oct4 knockdown condition. Overall, the obtained clusters display four main distinct expression patterns: high expression in Oct4 knockdown in Figure 3.7a, high expression in the control condition (H1P) in Figure 3.7b, low expression in Oct4 knockdown in Figures 3.7c-3.7h, and low expression in H1P in Figures 3.7i-3.7m. While the first two patterns are represented by two large clusters of sizes approximately 6,800 and 6,100 genes, respectively, each of the latter two is separated into one big cluster of size $> 6,000$ and a few smaller groupings. We again turn to GO term enrichment analysis to determine whether the clusters, particularly the small clusters with low expression in Oct4 knockdown, contain any biologically relevant genes.

GO term enrichment analysis indicates that all of the discovered clusters are in fact biologically relevant, containing genes enriched in both function and process categories. All of the clusters, however, shared many broad terms such as "binding" or "cellular process" with a large number of genes involved in these high-level functions and processes, and thus, we report only top unique GO terms in each cluster. The large H1P-low cluster in Figure 3.7i as well as any of the smaller H1P-low clusters in Figures 3.7j-3.7m did not have any unique GO terms associated with them. The H1P-high cluster in Figure 3.7b contained approximately one third of the genes in the "unannotated" process and function categories with $P = 4.8 \times 10^{-24}$ and FDR $\approx 6\%$, which means that the role of these genes have not yet been annotated and/or not yet determined. The six significant unique GO terms for the Oct4-high and -low clusters are presented in Table 3.2.

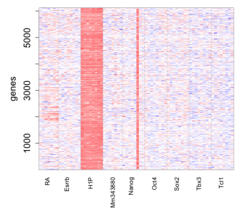
All of the small Oct-low clusters (Figures 3.7d-3.7h) have markedly evident expression differences from the big Oct4-low cluster of Figure 3.7c. Four out of the five small Oct4-low clusters were enriched for multiple unique GO terms with FDR $< 5\%$, mostly in the process ontology, confirming that these small clusters are biologically relevant and might have some distinct biological roles. Some examples of the significant unique GO terms of these clusters included reproductive system development, heart development, and cell cycle phase transition. The associated unique significant GO terms are provided in Table 3.3 and Table 3.4. Among these small Oct4-low clusters the one in Figure 3.7e is particularly interesting. In addition to low expression when Oct4 is knocked down, this cluster shows a high expression pattern in Sox2 knockdown, using the expression level in the control condition (H1P) as a reference. Thus, the two transcription factors, Oct4 and Sox2, regulate the genes in this cluster in an opposite way. This finding is in sharp contrast to the established co-regulation roles between Oct4 and Sox2 in ES cells ([ZCM07] and references therein), often regulating genes in a coordinated way

Table 3.2: Top unique GO terms enriched in the two clusters from Figure 3.7a and Figure 3.7c

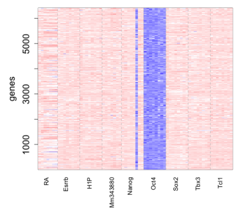
cluster	GO term	% in cluster	% in full set	P-value	FDR(%)
3.7a	regulation of localization	13.2	8.3	9.2×10^{-47}	0.0
	regulation of transport	10.1	6.1	7.2×10^{-41}	0.0
	cell surface receptor signaling pathway	15.3	10.4	2.7×10^{-39}	0.0
	receptor binding	9.5	5.7	1.3×10^{-38}	0.0
	receptor activity	7.4	4.6	2.2×10^{-27}	1.6
	signal transducer activity	7.8	4.9	1.1×10^{-26}	1.3
3.7c	cellular response to DNA damage stimulus	5.1	2.7	2.9×10^{-28}	0.1
	cellular macromolecular complex assembly	4.6	2.5	1.0×10^{-23}	0.3
	mitochondrion organization	2.8	1.3	1.5×10^{-22}	0.4
	structural constituent of ribosome	2.1	0.5	9.1×10^{-48}	0.0
	hydrolase activity	14.7	10.3	2.3×10^{-29}	0.1
	hydrolase activity, acting on acid anhydrides	5.6	3.1	8.1×10^{-27}	0.4



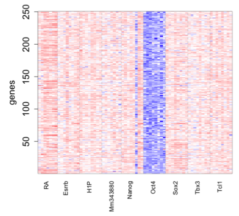
(a) 6,823 genes



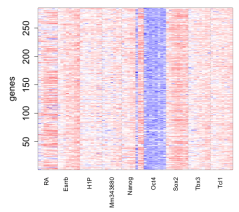
(b) 6,116 genes



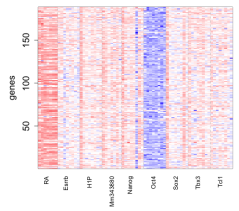
(c) 6,409 genes



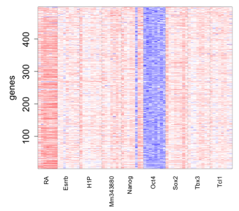
(d) 250 genes



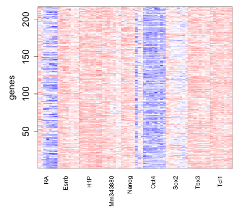
(e) 285 genes



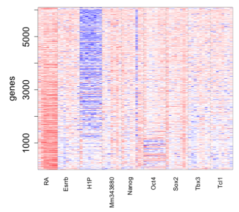
(f) 188 genes



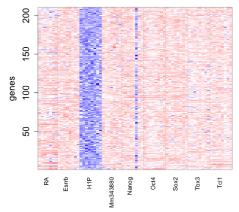
(g) 498 genes



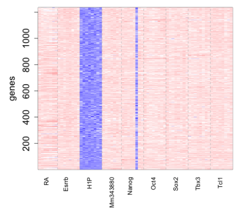
(h) 216 genes



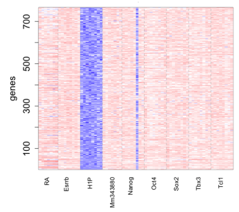
(i) 6,093 genes



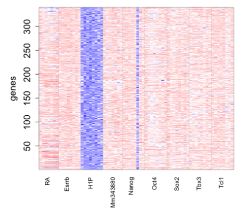
(j) 210 genes



(k) 1,235 genes



(l) 765 genes



(m) 339 genes

Figure 3.7: Clusters obtained from the Ivanova dataset. From left to right the experimental conditions are labeled as RA, Esrrb, control condition (H1P), Mm343880, Nanog, Oct4, Sox2, Tbx3, Tcl1.

Table 3.3: Top unique GO terms enriched in the small low Oct4 clusters from Figures 3.7e and 3.7f

cluster	GO term	% in cluster	% in full set	P-value	FDR(%)
3.7e	placenta development	4.7	0.7	5.9×10^{-8}	0.0
	embryonic placenta development	4.0	0.5	7.7×10^{-8}	0.0
	reproductive structure development	7.2	1.8	1.9×10^{-7}	0.0
	reproductive system development	7.2	1.8	2.2×10^{-7}	0.0
	cytoskeleton organization	11.2	4.0	2.6×10^{-7}	0.2
	hydro-lyase activity	2.2	0.3	1.1×10^{-4}	5.0
	bombesin receptor binding	0.7	0.0	1.7×10^{-4}	5.0
	TAP binding	1.1	0.0	1.7×10^{-4}	4.7
3.7f	regulation of G1/S transition of mitotic cell cycle	3.9	0.4	3.2×10^{-6}	0.0
	regulation of cell cycle G1/S phase transition	3.9	0.4	3.4×10^{-6}	0.0
	positive regulation of G1/S transition of mitotic cell cycle	2.2	0.1	1.3×10^{-5}	0.1
	positive regulation of mitotic cell cycle phase transition	2.8	0.2	1.4×10^{-5}	0.1
	positive regulation of cell cycle phase transition	2.8	0.2	1.8×10^{-5}	0.1
	poly(A) binding	2.2	0.1	6.0×10^{-6}	0.2
	poly-purine tract binding	2.2	0.1	1.7×10^{-5}	0.2
	mRNA binding	3.9	0.5	4.0×10^{-5}	0.6
	single-stranded RNA binding	2.8	0.2	6.7×10^{-5}	1.0
	heat shock protein binding	2.8	0.3	3.2×10^{-4}	4.0

NOTE: Tabulated are the top five unique GO terms from the process (top) and the function (bottom) ontology. The number in the first column refers to the cluster represented in each subplot in Figure 3.7.

by binding to adjacent DNA sites [MPZ10]. This cluster was uniquely enriched for genes responsible for reproductive system development. Another class of GO terms that were significant for this cluster can be classified as relating to cardiac and heart system development, all with $P \leq 5.1 \times 10^{-4}$ and FDR $< 5\%$.

It appears that none of the H1P-low clusters, including the largest one in Figure 3.7i, contained any unique GO terms, and moreover, none of the small H1P-low clusters were enriched for genes with a significant GO terms when compared to the big H1P-low cluster. It is possible that these clusters are a result of splitting as experienced with the non-spherical simulated data. Gene expression data usually contain clusters with a correlated structure, and splitting might be compounded due to a high degree of trimming for this particular cluster. As a result, splitting of some big clusters might be the price of discovering tight clusters in such a dataset. Fortunately, split clusters can be easily detected and merged, especially given a reasonably small number of them. A possible remedy is to decrease the amount of trimming. For example, if 10% trimming is applied to the clusters before the sequential assignment step, then 8 clusters will be obtained with the four largest clusters preserved. The other small clusters include one with the same pattern as in Figure 3.7h, one similar to that in Figure 3.7g, and two small H1P-low clusters instead of four. The smaller amount of trimming thus created larger clusters with less splitting, but some special patterns were not extracted from the big Oct4-low cluster.

The Ivanova dataset has shown that ISSPC is able to obtain previously unseen clusters that have meaningful biological functions. Moreover, some of the obtained clusters were very small, of sizes between 200 and 500. These small clusters were extracted from a dataset of over 45,000 genes along with clusters of size approximately 6,000, demonstrating the ability of ISSPC to find tight clusters with very small subsamples. These results were achieved in about 30 minutes of computational time and would not be computationally feasible with the orig-

Table 3.4: Top unique GO terms enriched in the small low Oct4 clusters from Figures 3.7g and 3.7h

cluster	GO term	% in cluster	% in full set	P-value	FDR(%)
3.7g	reproduction	9.4	3.6	7.8×10^{-9}	0.0
	multi-organism reproductive process	8.3	3.2	4.6×10^{-8}	0.0
	single organism reproductive process	9.8	4.2	6.7×10^{-8}	0.1
	sexual reproduction	7.5	2.8	1.1×10^{-7}	0.2
	gamete generation	6.4	2.2	1.3×10^{-7}	0.2
	protein domain specific binding	6.8	2.9	9.9×10^{-6}	2.1
3.7h	DNA duplex unwinding	2.4	0.2	2.3×10^{-5}	0.7
	DNA geometric change	2.4	0.2	2.6×10^{-5}	0.8
	regulation of cellular component biogenesis	7.1	2.1	4.7×10^{-5}	1.4
	regulation of lamellipodium assembly	1.9	0.1	4.9×10^{-5}	1.4
	response to ionizing radiation	3.3	0.5	7.4×10^{-5}	2.2
	DNA helicase activity	3.8	0.2	2.5×10^{-8}	0.0
	transcription regulatory region DNA binding	10	2.4	4.3×10^{-8}	0.0
	regulatory region DNA binding	10	2.4	5.4×10^{-8}	0.0
	regulatory region nucleic acid binding	10	2.4	5.4×10^{-8}	0.0
	ATP-dependent DNA helicase activity	3.3	0.2	1.2×10^{-7}	0.0

NOTE: Tabulated are the top five unique GO terms from the process (top) and the function (bottom) ontology. The number in the first column refers to the cluster represented in each subplot in Figure 3.7.

inal SPC algorithm or other clustering algorithms with time complexity $O(n^2)$. The computational savings of the iterative subsampling approach make it feasible to explore this dataset even further by varying the degree of trimming or the dimension number cutoff η and to discover more interesting patterns.

3.5 Discussion

We believe that ISSPC is a very promising approach to clustering as it allows us to perform computation with a relatively low trade off between speed and accuracy. It can successfully cluster large datasets in a reasonable amount of time. Such large datasets would otherwise require either filtering or some other preprocessing, which could potentially remove valuable data points. Gene expression datasets, for instance, are typically filtered based on coefficient of variation, and the genes that do not make a certain coefficient of variation cutoff are removed. Thus, some genes that do in fact carry patterns in their expression profiles and play a role in certain biological functions can be lost, which could confound the analysis. We have shown that ISSPC is able to produce good results with real data. Moreover, it generates a solution without the prior knowledge of the number of clusters and is able to effectively separate noise.

The ISSPC algorithm requires some fine tuning, which involves the dimension number cutoff η, the trimming percentage, and the subsample size ν. The solution can be sensitive to each of these tuning parameters. The computational savings, however, make it possible to quickly rerun a dataset with various values of these parameters. Additionally, in light of the problem of split clusters, especially with trimming, it is possible to address and introduce a merging step after the recursions are terminated. Such a merging step would make it easier for the user to interpret the clustering result and can potentially help with separating overlapping versus truly homogeneous clusters.

Iterative subsampling with sequential assignment can be applied generally to accelerate any advanced clustering method that has inferior time complexity, such as mclust, PWK-means [Tse07], convex clustering [CL13], or PRclust [PSL13].

However, the clustering mechanism at the center of the iterative subsampling approach would need to be able to estimate the number of clusters or provide a solution path as well as be able to isolate noisy data points. The simplicity of implementation and easy intuition alongside the orders of magnitude of computational savings could present further appeal of this methodology for big data applications.

CHAPTER 4

Conclusions and Future Work

In this dissertation we formulated and implemented a new clustering method SPC that uses a penalty on pairwise distances between the cluster centers to achieve greater degrees of sparsity and the merging of the cluster centers. The output of this clustering method is a short solution path, where each solution contains the cluster assignments for the data points and the estimated number of clusters. Since it is known that penalization can produce considerably biased estimates, which is particularly undesirable for clustering, we use the minimax concave penalty [Zha10] that has an explicit concavity parameter along with the regularization degree parameter. We control the bias directly through this concavity parameter when such bias becomes too high. Both the regularization and the concavity parameters are chosen in a data-driven way for each solution. Moreover, in each solution we determine which data points belong to noise by simply thresholding those clusters that have a very small size, e.g. $N_k \leq 3$. To accelerate this method and to allow it to handle big noisy data we introduce a simple iterative subsampling approach ISSPC with sequential cluster assignment, similar to discriminant analysis. This proves to save orders of magnitude of computation time and to largely preserve accuracy of the clustering partition. Simulated and real data studies further provide evidence of usefulness of this method.

In this dissertation we propose to use a fast and simple solution selection method as proposed in [FZ13] . However, we believe that a more rigorous investigation of the solution selection problem can be undertaken in the future. Solution selection does not need to be limited to choosing a particular solution from a solution path, but it can be approached as a node selection problem in a directed graph. A directed graph can be created from all the clusters across the whole solution path since it contains gradually merging clusters. Some rule could be created to choose particular nodes represented by clusters. For example, p-values or FDR from gene ontology analysis could be such a selection criterion for gene expression data.

Another interesting direction for future work is simultaneous feature selection. So far we have focused on clustering data points y_i by taking into account all of the dimensions or variables. It has been noted, however, that simultaneous identification of the so-called informative features or variables, especially for high-dimensional low-sample data, can enhance the clustering results and make interpretability much easier. Traditional approaches that separate variable selection and clustering, such as dimension reduction by PCA prior to clustering of the reduced space, can also destroy the original structure of the data and distort the result [PS07, WZ08, WT10]. A number of regularization approaches have been proposed to perform penalized log-likelihood clustering by imposing sparsity on the cluster mean parameters μ_k, $k = 1, \ldots, K$, where K is the number of clusters. The first such penalization method proposed in [PS07] applied the ℓ_1 penalty to all the mean parameters μ_{km} for $k = 1, \ldots, K$ and $m = 1, \ldots, p$, with p as a number of variables. Subsequently developed methods improved variable selection results by extending a group parameter penalty idea of group lasso regression [YL06] to the clustering problem. For example, the group penalty in [WZ08] is a weighted ℓ_∞ regularization of the mean parameters μ_{km}, and [XPS08] uses the ℓ_2 penalty for "vertical" (variable selection) and "horizontal" (addition of some prior knowledge about dimension groups) mean parameters. The idea behind these types of penalties is that they can shrink the mean parameters towards 0 if the data is centered to have the global mean 0 at each dimension, and thus they can distinguish which variables are not informative for clustering. If the mean is not 0, then the mean parameters can be re-parameterized as a sum of some global mean ϕ_m and a local mean ξ_{km}, and the shrinkage can be performed towards the global mean [PS07]. Another method for variable selection suggests a pairwise fusion penalty or an ℓ_1 penalty on the difference between μ_{km}'s for each pair of clusters, which can shrink some cluster means to a common value [GLM10]. All of the above mentioned methods rely on the EM algorithm to estimate the mean parameters, which requires the input of the number of clusters.

We have attempted to introduce simultaneous feature selection through an additional penalty on the cluster centers μ_k, similarly to the previously discussed methods; however, even though the minimizer could be derived in closed form, the resulting algorithm proved to be particularly slow, due its need to cycle through not only all the cluster centers but also all the dimensions. Consequently, we think that feature selection could be implemented in an easier and faster way through multiple comparison analysis of variance (ANOVA). For each cluster with size $N_k > 3$ and for each dimension m, we can compute the test statistic as $F_{km} = \frac{\sum_k N_k(\bar{y}_{km}-\bar{y}_m)^2/(N_k-1)}{\sum_k \sum_{i \in C_k}(y_{im}-\bar{y}_{km})^2/(n-K)}$, where $\bar{y}_m$ is the overall mean for dimension m, and $\bar{y}_{km}$ is the mean for cluster k in dimension m. P-values can then be calculated based on the F-distribution with $N_k - 1$ and $n - K$ degrees of freedom. Finally, the Benjamini-Hochberg procedure can be applied and only those dimensions chosen to be relevant, where FDR $\geq \beta$, for some cutoff β. Such procedure can be repeated after each SPC solution and the irrelevant dimensions removed for the subsequent set of SPC iterations.

Another possible future research direction related to this work is the generalization of the model in (2.1) to impose additional sparsity in the difference between μ_0, the mean of the null cluster C_0 that is described in Section 3.2.1, and μ_k's. This will introduce an additional penalty parameter λ_0, and to match the two penalties, we can choose $\lambda_0 = n\lambda$ since the size of the null cluster is close to n. The minimization problem (2.5) could then be easily reformulated and the same MM algorithm used to compute the solutions. Such sparsity in the null cluster distances could be useful in situations when a large majority (around 80-90%) of data does not show any grouping pattern. Handling noisy observations in an efficient manner could make the original SPC method considerably faster computationally and more convenient to the user.

Finally, we have not yet fully explored the theoretical properties of the estimator resulting from our method. Based on our very preliminary analysis, it seems that establishing the consistency of our method might be very technical and would require a substantial amount of additional work. As the main focus of this dissertation is on the practical side of this method, we plan to postpone these interesting and challenging theoretical topics to the future.

References

[AR13] C.C. Aggarwal and C.K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC, 2013.

[AV07] D. Arthur and S. Vassilvitskii. "K-means++: The advantages of careful seeding." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, Philadenphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[BCR97] H. Bensmail, G. Celeux, A.E. Raftery, and C. Robert. "Inference in model-based cluster analysis." *Statist. Comput.*, **7**:1–10, 1997.

[BFR98] P.S. Bradley, U.M. Fayyad, and C.A. Reina. "Scaling clustering algorithms to large databases." In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD98)*, pp. 194–198, Menlo Park, CA, 1998.

[BHS01] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. "Support vector clustering." *J. Mach. Learn. Res.*, **2**:125–137, 2001.

[BR93] J.D. Banfield and A.E. Raftery. "Model-based Gaussian and non-Gaussian Clustering." *Biometrics*, **49**:803–821, 1993.

[BR98] S. Byers and A.E. Raftery. "Nearest-neighbor clutter removal for estimating features in spatial point processes." *J. Amer. Statist. Assoc.*, **93**:577–584, 1998.

[CL13] E.C. Chi and K. Lange. "Splitting methods for convex clustering." preprint, available at `http://arXiv:1304.0499 [stat.ML]`, 2013.

[De 94] J. De Leeuw. "Block Relaxation Algorithms in Statistics." In H.H. Bock, W. Lenski, and M.M. Richter, editors, *Information Systems and Data Analysis*, pp. 308–324. Springer Verlag, 1994.

[DMM02] F. De Smet, J. Mathys, K. Machal, G. Thijis, B. De Moor, and Y. Moreau. "Adaptive quality-based clustering of gene expression profiles." *Bioinformatics*, **18**:735–746, 2002.

[EHJ04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. "Least angle regression (with discussion)." *Ann. Statist.*, **32**:407–499, 2004.

[EIM11] A. Ene, S. Im, and B. Moseley. "Fast clustering using mapreduce." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 271–280, 2011.

[FHH07] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani. “Pathwise coordinate optimization.” *Ann. Appl. Statist.*, **1**:302–332, 2007.

[FHT10] J. Friedman, T. Hastie, and R. Tibshirani. “Regularization paths for generalized linear models via coordinate descent.” *J. Statist. Softw.*, **33**:1–22, 2010.

[FKG12] P. Forero, V. Kekatos, and G.B. Giannakis. “Robust clustering using outlier-sparsity regularization.” *IEEE Transactions on Signal Processing*, **60**:4163–4177, 2012.

[FL01] J. Fan and R. Li. “Variable selection via non-concave penalized likelihood and its oracle properties.” *J. Amer. Statist. Assoc.*, **96**:1348–1360, 2001.

[FR02] C. Fraley and A.E. Raftery. “Model-based clustering, discriminant analysis, and density estimation.” *J. Amer. Statist. Assoc.*, **97**:611–631, 2002.

[FRB98] U.M. Fayyad, C.A. Reina, and P.S. Bradley. “Initialization of iterative refinement clustering algorithms.” In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD98)*, pp. 194–198, Menlo Park, CA, 1998.

[FRW05] C. Fraley, A. Raftery, and R. Wehrens. “Incremental model-based clustering for large datasets with small clusters.” *Journal of Computational and Graphical Statistics*, **14**(3), 2005.

[FS96] U. Fayyad and P. Smyth. “From massive data sets to science catalogs: applications and challenges.” In *Proceedings of the Workshop on Massive Data Sets*, pp. 29–142, 1996.

[FW12] Y. Fang and J. Wang. “Selection of the number of clusters via the bootstrap method.” *Comput. Stat. Data Anal.*, **56**:468–477, 2012.

[FZ13] F. Fu and Q. Zhou. “Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent.” *J. Amer. Statist. Assoc.*, **108**:288–300, 2013.

[GGM08] L.A. Garcia-Escudero, A. Gordaliza, C. Matran, and A. Mayo-Iscar. “A general trimming approach to robust cluster analysis.” *Ann. Statist.*, **36**:1324–1345, 2008.

[GLM10] J. Guo, E. Levina, G. Michailidis, and J. Zhu. “Pairwise variable selection for high-dimensional model-based clustering.” *Biometrics*, **66**:793–804, 2010.

[GRS01] S. Guha, R. Rastogi, and K. Shim. “Cure: an efficient clustering algorithm for large datasets.” *Information Systems*, **26**:35–58, 2001.

[HA85] J. Hubert and P. Arabie. “Comparing partitions.” *J. Classif.*, **2**:193–218, 1985.

[HJ06] R.J. Hathaway and Bezdek J.C. “Extending fuzzy and probabilistic clustering to very large data sets.” *Computational Statistics and Data Analysis*, **51**:215–234, 2006.

[HJB11] T.D. Hocking, A. Joulin, F. Bach, and J-P. Vert. “Clusterpath: an algorithm for clustering using convex fusion penalties.” In *28th international conference on machine learning*, 2011.

[HT96] T. Hastie and R. Tibshirani. “Discriminant analysis by Gaussian mixtures.” *J. Roy. Statist. Soc. Ser. B*, **58**:155–176, 1996.

[HTE00] T. Hastie, R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W. Chan, D. Botstein, and P. Brown. “Gene shaving as a method for identifying distinct sets of genes with similar expression patterns.” *Genome Biol.*, **1**:0003.1–0003.21, 2000.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2 edition, 2009.

[IDL06] N. Ivanova, R. Dobrin, R. Lu, I. Kotenko, J. Levorse, C. DeCoste, X. Schafer, Y. Lun, and I.R. Lemischka. “Dissecting self-renewal in stem cells with RNA interference.” *Nature*, **442**:533–538, 2006.

[Jai10] A.K. Jain. “Data clustering: 50 years beyond k-means.” *Pattern Recognition Lett.*, **31**:651–666, 2010.

[KM11] B. Kulis and I.J. Michael. “Revisiting k-means: new algorithms via bayesian nonparametrics.” preprint, available at `http://arXiv preprint arXiv:1111.0352`, 2011.

[KMC10] D.C. Koestler, C.J. Marsit, B.C. Christensen, M.R. Karagas, R. Bueno, D.J. Sugarbaker, K.T. Kelsey, and E.A. Houseman. “Semi-supervised recursively partitioned mixture models for identifying cancer subtypes.” *Bioinformatics*, **26**:2578–2585, 2010.

[Koh90] T. Kohonen. “The self-organizing map.” *Proceedings of the IEEE*, **78**:1464–1479, 1990.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* John Wiley and Sons, 1990.

[Lan95] K. Lange. "A gradient algorithm locally equivalent to the EM algorithm." *J. Roy. Statist. Soc. Ser. B*, **57**:425–437, 1995.

[Lan04] K. Lange. *Optimization.* Springer, New York, NY, 1 edition, 2004.

[LDY00] K. Lange, Hunter D.R., and I. Yang. "Optimization transfer using surrogate objective functions." *J. Comput. Graph. Statist.*, **9**:1–20, 2000.

[LOL11] F. Lindsten, H. Ohlsson, and L. Ljung. "Just relax and come clustering! a convexication of k-means clustering." Technical report, Linköping University, Department of Electrical Engineering, Automatic Control, 2011.

[Lux07] U. von Luxburg. "A tutorial on spectral clustering." *Stat. Comput.*, **17**:395–416, 2007.

[Mai01] R. Maitra. "Clustering massive datasets with application in software metrics and tomography." *Technometrics*, **43**(3):336–346, 2001.

[MBP02] G.J. Mclachlan, R.W. Bean, and D. Peel. "A mixture model-based approach to the clustering of microarray expression data." *Bioinformatics*, **18**:413–422, 2002.

[MFH11] R. Mazumder, J. Friedman, and T. Hastie. "SparseNet: coordinate descent with non-convex penalties." *J. Amer. Statist. Assoc.*, **106**:1125–1138, 2011.

[MFP09] M.J. Mason, G. Fan, K. Plath, Q. Zhou, and S. Horvath. "Signed weighted gene co-expression network analysis of transcription regulation in murine embryonic stem cells." *BMC Genomics*, **10**:327, 2009.

[MPZ10] M. J. Mason, K. Plath, and Q. Zhou. "Identification of context-dependent motifs by contrasting ChIP binding data." *Bioinformatics*, **26**(22):2826–2832, 2010.

[MR09] R. Maitra and I.P. Ramler. "Clustering in the presence of scatter." *Biometrics*, **65**:341–352, 2009.

[NDR14] I. Naim, S. Datta, J. Rebhahn, J.S. Cavenaugh, T.R. Mosmann, and G. Sharma. "SWIFT – Scalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, part1: algorithm design." *Cytometry Part A*, **85**:408–421, 2014.

[NH02] R. Ng and J. Han. "CLARANS: A method for clustering objects for spatial data mining." *IEEE Transactions on Knowledge and Data Engineering*, **14**:1003–1016, 2002.

[OR07] M.S. Oh and A.E. Raftery. "Model-based clustering with dissimilarities: a bayesian approach." *J. Comput. Graph. Statist.*, **16**:559–585, 2007.

[PDS05] K. Pelckmans, J. De Brabanter, J. Suykens, and B. De Moor. "Convex clustering shrinkage." In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[Pos01] C. Posse. "Hierarchical model-based clustering for large datasets." *Journal of Computational and Graphical Statistics*, **10**:464–486, 2001.

[PS07] W. Pan and X. Shen. "Penalized model-based clustering with application to variable selection." *J. Mach. Learn. Res.*, **8**:1145–1164, 2007.

[PSL13] W. Pan, X. Shen, and B. Liu. "Cluster analysis: unsupervized learning via supervized learning with a non-convex penalty." *J. Mach. Learn. Res.*, **14**:1865–1889, 2013.

[RD03] D. M. Rocke and J. Dai. "Sampling and subsampling for cluster analysis in data mining: With applications to sky survey data." *Data Mining and Knowledge Discovery*, **7**:215–232, 2003.

[RRP07] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. "Evaluating mapreduce for multi-core and multiprocessor systems." In *IEEE Symposium on High Performance Computer Architecture*, pp. 13–24, 2007.

[SAW14] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan. "Big Data Clustering: A Review." In *Computational Science and Its Applications?ICCSA 2014*, pp. 707–720. Springer International Publishing, 2014.

[SEC13] M. Soltanolkotabi, E. Elhamifar, and E.J. Candes. "Robust subspace clustering." preprint, available at `http://arXiv:1301.2603`, 2013.

[SMS03] R. Sharan, A. Maron-Katz, and R. Shamir. "CLICK and EXPANDER: a system for clustering and visualizing gene expression data." *Bioinformatics*, **19**:1787–1799, 2003.

[SSL10] Y. Shen, W. Sun, and K.C. Li. "Dynamically weighted clustering with noise set." *Bioinformatics*, **26**:341–347, 2010.

[SW12] W. Sun and J. Wang. "Regularized k-means clustering of high-dimensional data and its asymptotic consistency." *Elec. J. Stat.*, **6**:148–167, 2012.

[Tib96] R. Tibshirani. "Regression shrinkage and selection via the lasso." *J. Roy. Statist. Soc. Ser. B*, **58**:267–288, 1996.

[TMZ06] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G.C. Tseng. "Evaluation and comparison of gene clustering methods in microarray analysis." *Bioinformatics*, **22**:2405–2412, 2006.

[Tse07] G.C. Tseng. "Penalized and weighted k-means for clustering with scattered objects and prior information in high-throughput biological data." *Bioinformatics*, **23**:2247–2255, 2007.

[TSR05] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. "Sparsity and smoothness via the fused lasso." *J. Roy. Statist. Soc. Ser. B*, **67**:91–108, 2005.

[TW05a] R. Tibshirani and G. Walther. "Cluster validation by prediction strength." *J. Comput. Graph. Statist.*, **14**:511–528, 2005.

[TW05b] G.C. Tseng and W.H. Wong. "Tight clustering: a resampling-based approach for identifying stable and tight patterns in data." *Biometrics*, **61**:10–16, 2005.

[TWH01] R. Tibshirani, G. Walther, and T. Hastie. "Estimating the number of clusters in a data set via the gap statistic." *J. Roy. Statist. Soc. Ser. B*, **63**:411–423, 2001.

[TY09] P. Tseng and S. Yun. "A coordinate gradient descent method for nonsmooth separable minimization." *Mathematical Programming*, **117**:387–423, 2009.

[WBF04] R. Wehrens, L. M. Buydens, C. Fraley, and A. E. Raftery. "Model-based clustering for image segmentation and large datasets via sampling." *Journal of Classification*, **21**(2):231–253, 2004.

[WL08] T.T. Wu and K. Lange. "Coordinate descent algorithms for lasso penalized regression." *Ann. Appl. Statist.*, **2**:224–244, 2008.

[WNM08] H. Wang, J. Neill, and F. Miller. "Nonparametric clustering of functional data." *Stat. Interface*, **1**:47–62, 2008.

[WT10] D. Witten and R. Tibshirani. "A framework for feature selection in clustering." *J. Amer. Statist. Assoc.*, **105**:713–726, 2010.

[WZ08] S. Wang and J. Zhu. "Variable selection for model-based high-dimensional clustering and its applications to microarray data." *Biometrics*, **64**:440–448, 2008.

[XPS08] B. Xie, W. Pan, and X. Shen. "Variable selection in penalized model-based clustering via regularization on grouped parameters." *Biometrics*, **64**:921–930, 2008.

[YFM01] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, and W.L. Ruzzo. "Model-based clustering and data transformations for gene expression data." *Bioinformatics*, **17**:977–987, 2001.

[YL06] M. Yuan and Y. Lin. "Model selection and estimation in regression with grouped variables." *J. Roy. Statist. Soc. Ser. B*, **68**:49–67, 2006.

[ZCM07] Q. Zhou, H. Chipperfield, D.A. Melton, and W.H. Wong. "A gene regulatory network in mouse embryonic stem cells." *PNAS*, **104**:16438–16443, 2007.

[ZH05] H. Zou and T. Hastie. "Regularization and variable selection via the elastic net." *J. Roy. Statist. Soc. Ser. B*, **67**:301–320, 2005.

[Zha10] C.H. Zhang. "Nearly unbiased variable selection under minimax concave penalty." *Ann. Statist.*, **38**:894–942, 2010.

[ZPS09] H. Zhou, W. Pan, and X. Shen. "Penalized model-based clustering with unconstrained covariance matrices." *Elec. J. Stat.*, **3**:1473–1496, 2009.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: an efficient data clustering method for very large databases." In *ACM SIGMOD Record*, volume 25, pp. 103–114, 1996.