ABSTRACT

Hierarchical Stability Based Model Selection for Clustering Algorithms

Bing Yin, M.S.C.S.

Advisor: Gregory J. Hamerly, Ph.D.

We present an algorithm called HS-means, which is able to learn the number of clusters in a mixture model based on the hierarchical analysis of clustering stability. Our method extends the concept of clustering stability to a concept of hierarchical stability. The method estimates a stable model for the data based on analysis of stability; it then analyzes the stability of each component in the estimated model and chooses a stable model for this component. It continues this recursive stability analysis until all the estimated components are unimodal. In so doing, the method is able to handle data symmetry that existing stability based algorithms have difficulty with. We test our algorithm on both synthetic datasets and real world datasets. The results show that HS-means apparently outperforms existing stability based model selection algorithms and is competitive to other often-used model selection methods.

Hierarchical Stability Based Model Selection for Clustering Algorithm

by

Bing Yin, B.S.

A Thesis

Approved by the Department of Computer Science

_____

Donald L. Gaitros, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science

Approved by the Thesis Committee

_____

Gregory J. Hamerly, Ph.D.

_____

Michael J. Donahoo, Ph.D.

_____

James D. Stamey, Ph.D.

Accepted by the Graduate School
August 2009

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES vii

ACKNOWLEDGMENTS vii

DEDICATION ix

Introduction 1

1.1 Motivation 1

1.1.1 Introduction to Clustering Analysis 1

1.1.2 Introduction to Model Selection 5

1.1.3 Introduction to Clustering Stability 6

1.2 Research Summary 8

1.3 Structure of This Thesis 9

Related Work 11

2.1 Clustering Algorithms 11

2.1.1 K-Means Algorithm 11

2.1.2 EM Algorithm 12

2.2 Some Popular Model Selection Algorithms 14

2.2.1 G-Means 14

2.2.2 PG-Means 16

2.2.3 Gap Statistic 17

2.3 Existing Stability Based Model Selection Algorithms 18

2.3.1 Levin and Domany's Algorithm 20

2.3.2 Ben-Hur, Elisseeff and Guyon's Algorithm 21

2.3.3 Lange, Roth, Braun and Buhmann's Algorithm 23

The Data Symmetry Problem 25

3.1 The Problem 25

3.2 The Ideas for Solution 27

Hierarchical Stability Based Model Selection 29

4.1 The Algorithm 29

4.2 Stability Analysis 31

4.2.1 Compare Clustering by Variation of Information (VI) 32

4.2.2 Stability Analysis 35

4.3 Unimodality Test 35

4.3.1 Dip Statistics 36

4.3.2 Gap Statistics 37

4.3.3 χ^2 *Unimodality Test* 37

Experiment Evaluation 41

5.1 Performance on Generated Data Sets 41

5.1.1 Generating the Data Sets 41

5.1.2 Algorithm Performance 42

5.1.3 Performance on the Symmetric Data Sets 45

5.2 Performance on Real World Data Sets 46

5.2.1 Handwriting Digits Data Set 46

5.2.2 Synthetic Control Curves Data Set 47

5.2.3 Data Projection 47

5.2.4 Algorithm Performance 50

5.3 Experiments on SimPoint 51

Conclusion and Future Work 54

BIBLIOGRAPHY 56

LIST OF FIGURES

Figure 1:	Two Clusterings Where *k* Was Not Properly Chosen	4
Figure 2:	A Simple Illustration of Stability	10
Figure 3:	Illustration of Gap Statistics	19
Figure 4:	An Illustration of Data Symmetry	26
Figure 5:	Data Symmetry in Program Execution Data	27
Figure 6:	Distribution of Sum of Squared Gaussians	39
Figure 7:	An Illustration of HS-means	43
Figure 8:	HS-means versus Lange Algorithm on Symmetric Data Sets	48
Figure 9:	Six Clusters in Control Charts Data Set	49
Figure 10:	PCA Illustration on Handwriting Data and Control Charts Data	49

LIST OF TABLES

Table 1:	Learned k on Different Synthetically Generated Data Sets	44
Table 2:	Distribution of Examples for Handwriting Data Set	46
Table 3:	Learned *k* on Handwriting Data and Control Charts Data	51
Table 4:	SPEC CPU 2000 Benchmark Results on CPI Variance Reduction.	53
Table 5:	SPEC CPU 2000 CPI Prediction Results	53

ACKNOWLEDGMENTS

I would like to thank Dr. Hamerly for his guidance and encouragement throughout my thesis work. It is great pleasure and happiness to work with him on this research. I would like to thank Dr. Donahoo for his suggestions and comments on this thesis. The past two years time of working together with him has been a great time which I would never forget. I would like to thank Dr. Stamey for his help on revising this thesis and the knowledge I learned in his statistics class which is a great help to my thesis work. I would like to thank my program director Dr. Sturgill for his kindness and his guidance to me throughout my two years study in Baylor University.

DEDICATION

To my dear parents

for their love and their sacrifice for me to have an education

CHAPTER ONE

Introduction

1.1 Motivation

1.1.1 Introduction to Clustering Analysis

Clustering analysis has been considered as one of the most important exploratory data analysis tools, which aims at sorting different objects into groups in a way that the degree of association between two objects is maximal if they belong to the same group and minimal otherwise. Given the above, cluster analysis can be used to discover structures in data without providing an explanation/interpretation. In other words, cluster analysis simply discovers structures in data without explaining why they exist.

Clustering techniques have been applied to a wide variety of research problems and real world applications which are helping us in everyday life. For example, in psychology and medicine, clustering analysis can be used to detect patterns in the spatial or temporal distribution of a disease and identify different types of subcategories of a illness; in climate research, clustering analysis has been applied to find patterns in the atmospheric pressure of polar regions and areas of the ocean that have a significant impact on land climate; in business, clustering analysis can be used to analyze the large amount of information on current and potential customers by segmenting customers into small number of groups for additional analysis and marketing activities; in information retrieval, clustering techniques are widely applied as a tool for post search analysis in the search engine which groups the searched results into several categories for the users; in biology, the analysis of gene family information, gene expression information, and

taxonomy all find the importance of clustering analysis; in data compression, clustering analysis is used in image, sound, and video data compression where many of the data objects are highly similar to each other. In general, whenever one needs to classify a "mountain" of information into manageable meaningful piles, clustering analysis is of great utility.

Throughout many years, researchers in statistics and machine learning have developed many clustering algorithms such as Hierarchical Clustering (J.H. Ward 1963), K-means (S.P. Lloyd 1982), Expectation Maximization (EM) (Arthur, Laird and Rubin 1977), Fuzzy C-Means (J. Bezdek 1981), spectral clustering algorithms (A.Y. Ng, M. Jordan and Y. Weiss 2001) and so on. In fact, there is a large collection of clustering algorithms available now. We can first distinguish these various types of clustering algorithms by how they group the data: hierarchical versus partitional, and exclusive versus overlapping.

- Hierarchical versus Partitional. A partitional clustering is a division of the data set into non-overlapping subsets such that each data objects belongs to exactly one subset. A typical example is the K-Means clustering algorithm. Hierarchical clustering permit clusters to have subclusters. The clusters and their subclusters are organized as a tree where each node is the union of its children (subclusters) with the root node containing all the objects in the data set. A partitional clustering can be obtained by cutting the tree at a particular height. The commonly used single linkage clustering is an example of this kind of algorithms.
- Exclusive versus Overlapping. In exclusive clustering algorithms, one data object is assigned to a single cluster, for example the K-Means algorithm. In some cases this could be a problem because where objects could reasonably be assigned to

more than one cluster. The overlapping clustering algorithms solves this problem by allowing objects can be assigned simultaneously to multiple groups. The EM algorithm and the Fuzzy-Means are examples where a single data point can be assigned to multiple clusters.

Different clustering algorithms also have different view of what is a cluster. Generally they could interpret clusters as center based, connectivity based and density based. In center based clustering, like K-Means, a cluster is a set of objects in which each object is closer to the center that defines that cluster than to the centers of any other cluster. In connectivity based clustering, like spectral clustering algorithms, a cluster can be viewed as a connected component in the graph in which each object is highly connected to objects within the cluster and loosely connected to objects outside the cluster. In density based clustering, like the EM algorithm with a mixture model, a cluster is a dense region of objects surrounded by a region of low density.

Though these algorithms vary in their definition of a cluster and the way to group objects into clusters, most of them have a common problem: the number of clusters in the data should be known before running of the algorithms. Without this prior knowledge, these algorithms cannot find the optimal grouping of the objects in the data. In other words, they may result in some clusterings which are neither meaningful nor useful. As we walk through the details of several clustering algorithm in next chapter, this problem will be clearer. At this point, we can understand this problem in the following way. Clustering algorithms work in this way: you must first tell the algorithm to look for *k* clusters in the data. Then the algorithm analyzes the data set and groups the objects into *k* clusters according to its definition of cluster and its strategy of grouping objects into clusters. The algorithm cannot discover how many clusters are really in the data. It will

just return *k* clusters you tell it to look for. The wrong *k* will result in a clustering of bad quality. For example, if the data set contains four clusters but the algorithm is given the order to look for five clusters, the resulting clustering would not be the best clustering for the data set. We are using clustering algorithms because we want get a useful and meaningful grouping of the data. If the returned clustering is meaningless or useless, why do we waste the resources on clustering algorithms? So giving clustering algorithms the right *k* is a critical part in the application of clustering algorithms. Figure 1 shows examples where *k* has been improperly chosen.

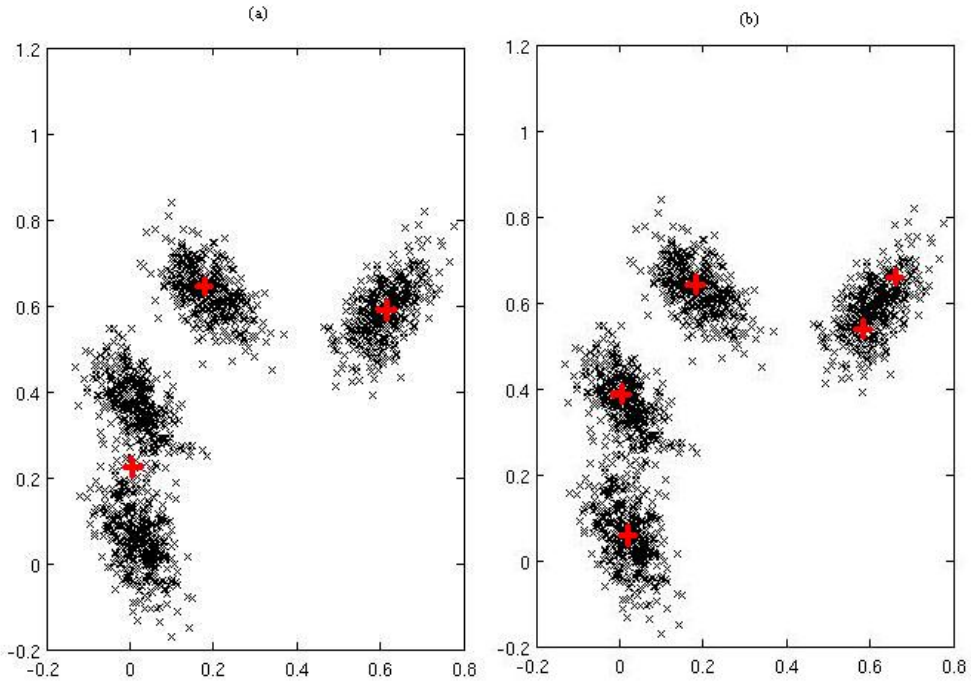

Figure 1: Two Clusterings Where *k* Was Not Properly Chosen. The crosses represent cluster centers. The data contains four clusters. In figure 1.a, there are too few clusters used, and in figure 1.b there are too many clusters used.

Here is a problem: how do we know the number of clusters beforehand? In some cases, we can use the domain knowledge or some expert guess to obtain this information. However, in most cases, the information is unavailable. Automatically selecting the right number of clusters in the data is often referred as the model selection problem for

clustering algorithms. Unfortunately, despite the fact that clustering algorithms have been thoroughly researched, the model selection problem remains a challenging issue for data clustering. Though, many ideas have been proposed, but all of them have limitations and neither of them works perfectly.

1.1.2 Introduction to Model Selection

Model selection is originally a term widely used in statistics, which aims at selecting a statistical model that best fits the given data from a set of potential models. This is a more general view of model selection. In clustering algorithms, researchers often use model selection to refer to the task of automatically selecting the number of clusters (k) which best fit the given data. Though some times other parameters of the clustering algorithms such as the centers of clusters and membership of the objects to the clusters are also included in the model selection, most of the time this term focuses on selecting the right *k*. As we said, model selection is an important and challenging problem of cluster analysis. Thus a significant amount of research effort has been devoted to this problem. And as a result, a variety of different methods are proposed. Several popular methods include the Akaike information criterion (AIC) (H. Akaike 1974), Bayesian information criterion (BIC) (G.E. Schwarz 1978), X-Means (A.Moore and D. Pelleg 2000), the gap statistic (R. Tibshirani, T. Hastie and G. Walther 2001), G-Means (G. Hamerly and C. Elkan 2003) and PG-Means (Y. Feng and G. Hamerly 2006), and stability based approaches. AIC, BIC and Deviance information criterion are proposed in the area of statistics based on the likelihood ratio approach. G-Means and PG-Means were proposed by Greg Hamerly and his colleagues based on model fitness testing. G-Means incrementally increase *k* and uses hypothesis testing technique called Anderson-Darling Test (T.W. Anderson and D.A. Darling 1952) which tests whether each cluster in

the new model is Gaussian or not. If not, increase *k* and repeat the process again. PG-Means also incrementally increase *k*. Unlike G-Means, for each *k* PG-Means tests whether the whole model fits the data or not instead of testing whether each cluster is a Gaussian or not. The number of clusters will be increased until a model fitting the data is found. The Gap statistic is based on the fact that the within-cluster dispersion (the sum of distance from each object to the center of its cluster) drops most apparently around the right *k*. Among these methods, stability is drawing a lot of attention recently which is based on the observed fact that the clustering results are stable when the algorithm is told the right *k* and instable with the wrong *k*.

Despite the fact that many model selection methods for clustering algorithms have been proposed, there is no common agreement on which is the best solution to this problem. Those proposed methods work under certain situations, but they all have limitations. The problem is so hard because generally there is no common standard on what is a good clustering, which make sense because when different algorithms look at a data set their have different interpretations of the data and their own view of the "good clustering". However, stability was proposed as a model selection algorithm which is independent of the underlying clustering algorithm used. In other words, stability based model selection methods work with any clustering algorithm, and the result should not vary with the particular clusterings algorithm used.

1.1.3 Introduction to Clustering Stability

As we said stability based model selection algorithms are based on the fact that correct model will often lead the clustering results, obtained by repeating the clustering algorithms on the same data for many times, to be very similar to each other. In other words, correct model tends to have the stable clustering results with regarding to the

repeating of the algorithm while on the other hand wrong models will yield unstable results (S. Ben-David, U. von Luxburg and D. Pal 2006; S. Ben-David, D. Pal and H.U. Simon 2007). Several algorithms have been proposed and tested on some generated toy data. It has appeared as a promising approach and drawn significant attention from researchers.

The intuitive idea behind this stability based approach is that if we repeatedly sample the data and apply the clustering algorithm with the parameter k, then a "correct" k should lead the algorithm to produce clusterings that do not vary much from one sample to another. In other words, the algorithm with the specified k is stable with respect to input randomization. On the other hand, if k is a wrong number then the resulting clustering from the algorithm will be unstable. For example, if the given data contains two clusters, but we run the algorithm for three clusters in the data, the algorithm needs to split one of the clusters into two. Which of the two clusters is split may change from one sample to another and result in instable clustering results. Generally speaking, this fact follows the common rule that scientific truth should be reproducible in experiments with respect to randomizations of the experiments. Figure 2 is a simple illustration on this intuitive idea about how stability can tell the number of clusters in the data set.

Based on this intuition, several methods are proposed to tune the parameters of clusterings algorithms, like the number of clusters, centers of the clusters, or various stopping criteria. As we can see from the intuitive idea discussed above, stability will be defined by comparing the clustering results from the same algorithm with input randomizations. This brings the two key parts for all stability based model selection approaches: how to get different results from the algorithm using randomization and how

to calculate how much these results vary from each other. A common approach to introduce the input randomization to clustering algorithm is resampling the data set which is widely used by the existing stability based model selection algorithms. Note that the size of the resampled data set should be large enough to preserve the clustering behavior of the original data set. Another approach we found is randomly choosing the starting position of the cluster centers without resampling the original data. We applied this approach in our algorithm and compared the performance with resampling. The results show that this is also an useful approach to introduce input randomization to the algorithm. As we talk about several existing stability based model selection algorithms, we will explain in details how the resampling approach works. The details on random initialization of cluster centers will be discussed in Chapter Four as we move on to our algorithm.

Not only working on proposing many stability based model selection methods, researchers also work on the theoretic basis of these methods. However, after thorough investigation, we find that these proposed stability based model selection algorithms often have bad performance on some data because they cannot handle a special structure we call symmetric data which happens a lot in real application data. To solve this problem, we propose our hierarchical stability based model selection algorithm.

1.2 Research Summary

In this thesis work, we thoroughly investigated the existing stability based model selection algorithms and tested them on a large amount of data including both synthetically generated datasets and real application datasets. By analysis the performance and results of these methods, we identified the symmetric data problem which is a bottleneck for the current concept of stability. To solve the problem we

extended the current concept of stability to a concept of hierarchical stability which can handle the symmetric data very well. During the work, we also found that the common belief that resampling is a necessity of evaluating stability is not always true.

To extend the stability into a hierarchical manner, one critical question need to be addressed is unimodality testing (single cluster problem). Because the current concept of stability has an assumption that the data set always contains at least two clusters, when it comes to a single cluster, stability based methods cannot detect this case. To address this problem, we involve unimodality testing during the analysis of stability to identify these single clusters against mixture of clusters. Unimodality testing is a key part in hierarchical stability based model selection. Unfortunately, unimodality testing is a still an open and challenging problem in statistics. We investigated many existing techniques for unimodality testing and developed our own unimodality testing method which works well.

We tested our hierarchical stability based algorithm on a variety of data. The tests have shown that our work made the stability a competing and powerful tool for model selection. We also tested our algorithm on data from SimPoint application and compared the results with current SimPoint application. This application has shown that our algorithm can be applied in real world applications.

1.3 Structure of This Thesis

The rest of this paper is organized as following. Chapter Two will describe in details about the related work. This will include the introduction to two clustering algorithms we used in our program, some popular non-stability based model selection algorithms and some existing stability based model selection algorithms. The focus of Chapter Two will be the existing stability based model selection algorithms. In Chapter

Three, we will describe the data symmetry problem which causes these existing stability based model selection algorithm not performing well on some real application data. In Chapter Four, we will talk about the detail of our hierarchical stability based algorithm which we call HS-means as well as the unimodality testing techniques we used in our algorithm. The experimental results will be in Chapter Five followed by conclusion and discussing in Chapter Six.

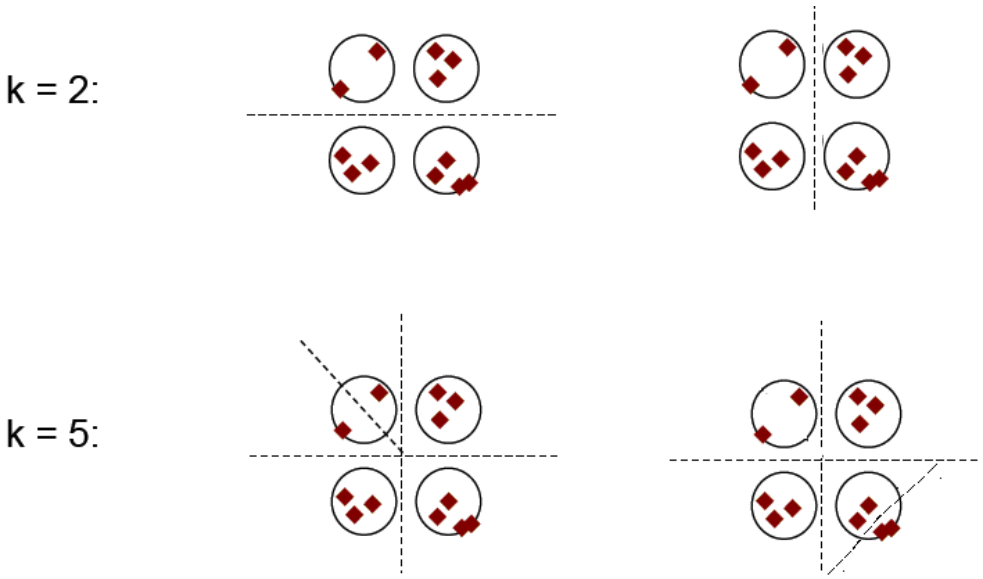

Figure 2: A Simple Illustration of Stability. The toy data contains four clusters. Clustering the data into five clusters confuses the algorithm on how to split the data and the resulted clustering varies much when we repeatedly run the algorithm. The same thing happens when clustering the data into two clusters. Only when clustering the data into four clusters, there is no confusing and the results are stable. Thus we can say the data contains four clusters as shown by the circles.

CHAPTER TWO

Related Work

In this chapter, I summarize the related work from other researchers. I first talk about details of two clusterings algorithm which are used in my research: K-Means and Expectation-Maximization (EM). Then I introduce some popular model selection methods and algorithms, followed by introduction to details of some existing stability based model selection algorithms.

2.1 Clustering Algorithms

A large number of clustering algorithms are available, as we talked in Chapter One. In my thesis work, two popular clustering algorithms are used: K-Means and Expectation-Maximization (EM). The K-Means algorithm is a fairly fast hard clustering algorithm which assigns one observation to only one cluster while EM algorithm is a soft clustering algorithm which may assign one observation to multiple clusters.

2.1.1 K-Means Algorithm

The K-Means algorithm was first proposed by Stuart Lloyd as a technique for pulse-code modulation (S.P. Lloyd 1982). Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a *d*-dimensional vector, then K-Means algorithm aims to partition the observations into k sets $(k < n)$ $S = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster dispersion which is calculated as

$$W = \sum_{i=1}^{k} \sum_{x_j \in S_i} \| x_j - m_i \|^2$$

where m_i is the mean of S_i.

The algorithm uses an iterative refinement technique. Given an initial set of *k* means $m_1^{(0)}, m_2^{(0)}, ..., m_k^{(0)}$ where the number in the parentheses indicates how many iterations have been performed including the current iteration. The initial set of *k* means can be specified randomly or by some heuristic. After starting, the algorithm proceeds by alternating between assignment step and update step until it converges. In the assignment step, the algorithm assigns each observation to one cluster whose mean (center) is the closest to this observation, that is, partitions the observations according to Voronoi diagram generated by the means of the *k* sets. In the update step, the algorithm calculates the new mean for each set according to the assignments obtained in the assignment step. Algorithm 1 is a formal description of K-Means algorithm. The algorithm will run until it converges. To increase the speed, many different versions of K-Means have been proposed. The version we use in our program is called K-Means++ which chooses better starting clusters such that the algorithm will converge quickly (D. Arthur and S. Vassilvitskii 2007).

2.1.2 EM Algorithm

An expectation-maximization (EM) algorithm is used in statistics for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. Like K-Means, EM is also an iterative method which alternates between performing an expectation (E) step, which computes an expectation of the log likelihood with respect to the current estimate of the distribution

for the latent variables, and a maximization (M) step, which computes the parameters which maximize the expected log likelihood found on the E step. These parameters are then used to determine the distribution of the latent variables in the next E step.

Algorithm 1 K-Means Algorithm

Input: data set X containing n observations on d attributes; initial set of k centers $m_1^{(0)}, m_2^{(0)}, ..., m_k^{(0)}$

Output: $S = \{S_1, S_2, ..., S_k\}$, the partition of X in k sets

Repeat until convergence:

1. assign each observation to the closest center

$$S_i^{(t)} = \{x_j : \| x_j - m_i^{(t)} \| \leq \| x_j - m_l^{(t)} \| \ for\ all\ l = 1, ..., k\}$$

2. update the k centers in $(t+1)^{th}$ iteration according to S in t^{th} iteration

$$m_i^{(t+1)} = \frac{1}{| S_i^t |} \sum_{x_j \in S_i^{(t)}} x_j$$

We use EM algorithm to estimate a Gaussian mixture from the given data. The Gaussian EM is one of the implementation of EM algorithm which aims at learning Gaussian mixture model from the data. Given a set of observations $(x_1, x_2, ..., x_n)$ and the number of Gaussians in the mixture (k), the algorithm estimates the centers $\mu_1, \mu_2, ..., \mu_k$, covariance matrixes $\sum_1, \sum_2, ..., \sum_k$ and the prior probabilities $p_1, p_2, ..., p_k$ for all Gaussians in the mixture. Denote by $\lambda_t = \{\mu_1, \mu_2, ..., \mu_k, \sum_1, \sum_2, ..., \sum_k, p_1, p_2, ..., p_k\}$ our estimates on the t^{th} iteration. In the E-step, the algorithm calculates for each observation the probability it comes from each Gaussian in the mixture. Then based on this new estimation in E-step, in the M-step the algorithm updates the new mean, covariance, and prior probability for each Gaussian. These two steps will be alternated until the algorithm

converges or reaches some specified stop criteria. Algorithm 2 is a more formal description of Gaussian EM algorithm.

2.2 Some Popular Model Selection Algorithms

Several algorithms have already been proposed to automatically learn the number of clusters from the data. I will talk about some of them with which I will compare the performance of our algorithm.

Algorithm 2 Gaussian EM Algorithm

Input: data set X containing *n* observations; initial model parameters λ_1

Output: $\{\mu_1, \mu_2, ..., \mu_k, \sum_1, \sum_2, ..., \sum_k, p_1, p_2, ..., p_k\}$, the estimated mixture model

Repeat until convergence:

1. E-step: calculate for each observation the probability it from each Gaussian

$$P(w_i \mid x_l, \lambda_t) = \frac{P(x_l \mid w_i, \lambda_t) P(w_i \mid \lambda_t)}{P(x_l \mid \lambda_t)} = \frac{g(x_l \mid w_i, u_i(t), \sum_i(t)) p_i(t)}{\sum_{j=1}^{k} g(x_l \mid w_j, u_j(t), \sum_j(t)) p_j(t)} \quad \textbf{for } i = 1, ..., k$$

where w_i indicates the probability this observation comes from the i^{th} Gaussian, $g(x_l \mid w_i, u_i(t), \sum_i(t))$ is the evaluation of a Gaussian at this observation x_l, $\mu_i(t), \sum_i(t), p_i(t)$ are the estimates of those parameters in the t^{th} iteration.

2. M-step: updates the estimates for the three model parameters

$$\mu_i(t+1) = \frac{\sum_l P(w_i \mid x_l, \lambda_t) x_l}{\sum_l P(w_i \mid x_l, \lambda_t)}, \quad p_i(t+1) = \frac{\sum_l P(w_i \mid x_l, \lambda_t)}{n}$$

$$\sum_i(t+1) = \frac{\sum_l P(w_i \mid x_l, \lambda_t)[x_l - \mu_i(t+1)][x_l - \mu_i(t+1)]^T}{\sum_l P(w_i \mid x_l, \lambda_t)}$$

2.2.1 G-Means

G-means stands for Gaussian Means, proposed by Greg Hamerly and Charles Elkan as method which can learn the best k for a given data (G. Hamerly and C. Elkan

2003). G-means runs k-means with increasing k until the test accepts the hypothesis that the data assigned to each k-means center are Gaussian. G-means algorithm starts with a small *k* like *k* = 1 and grows the *k* as the algorithm runs. Each iteration of the algorithm splits the clusters whose data appear not to come from a Gaussian distribution into two. Between each round of splitting, k-means is run on the entire dataset and all the centers to refine the current clustering solution. The decision of whether to split a cluster into two is made based on the Anderson-Darling statistic, which is shown to be a powerful one dimensional normality test based on the empirical cumulative distribution function. To test whether a cluster is from Gaussian or not, the algorithm projects the data to one dimension along the direction that k-means has found to be important for separating the data. If the data currently assigned to a k-means cluster appear to be Gaussian, then only one center will be used to represent the data. However, if the same data do not appear to be Gaussian, multiple centers will be used to model the data properly. G-means runs k-means up to k times when finding k centers, so the time complexity is at most O(*k*) times that of k-means.

Algorithm 3 G-means(Dataset X, significant level α)

Input: The data set X; the significant level of the normality fitness testing α.
Output: the number of clusters best fit the data

1: Let *C* be the initial set of cluster centers
2: $C \leftarrow$ K-Means (*C*, *X*).
3: Let $\{x_i \mid class(x_i) = j\}$ be the set of data points assigned to center c_j .
4: Use Anderson-Darling statistic test to detect if each $\{x_i \mid class(x_i) = j\}$ follow a Gaussian distribution (at confidence level α).
5: If the data look Gaussian, keep c_j. Otherwise replace c_j with two centers.
6: Repeat from step 2 until no more centers are added.

Algorithm 3 describes the G-means algorithm. The critical part is in line 4 where the statistical testing is used to determine whether new clusters should be added or not. Note that this algorithm has a strong assumption that the clusters in the given data set can be modeled by Gaussians. This assumption leads the algorithm to perform badly on the data sets which are not from Gaussian mixtures. Also since the G-means is a wrapper around k-means algorithm which is an exclusive clustering algorithm, when the data set contains overlapping clusters the algorithm will result in a number of non-Gaussian "clipped" clusters. These "clipped" clusters will fail in the normality test and thus be split further. As a result, if the data contains overlapping clusters, G-means will find a *k* much larger than the correct one. We will see this disadvantage in next chapter when we compare them with our algorithm on several real data.

2.2.2 PG-Means

As we mentioned above, G-means does not work well with non-Gaussian data sets and overlapped data sets. To fix this, Greg Hamerly proposed another algorithm called Projected Gaussian Means (PG-means) which keep the basic idea of G-means but improved the statistical testing for model fitness and resulted in a greatly improved performance (Y. Feng and G. Hamerly 2006). Like G-means, PG-means starts with a simple model and increases *k* by one at each iteration until it finds a model that fits the data well. Unlike G-means, PG-means test whether the whole projected model (all the clusters found so far) fits the whole projected data well instead of performing statistical test for each cluster individually. This avoids the "clipped" cluster problem as we have seen in G-means. The statistical test performed by PG-means is Kolmogorov-Smirnov test (F.J. Massey 1951). Algorithm 4 describes the PG-means algorithm.

Algorithm 4: PG-means (dataset *X*, confidence α, number of projections *p*)

Input: Data set *X*; confidence level α for the Kolmogorov-Smirnov test; number of projects *p*

Output: number of clusters best fit the data and the clustering

1: Let $k \leftarrow 1$. Initialize the cluster with the mean and covariance of *X*.
2: for $i = 1,..., p$ do
3: Project *X* and the model to one dimension with the same projection.
4: Use the KS test at significance level α to test if the projected model fits the projected dataset.
5: If the test rejects the null hypothesis, then break out of the loop.
6: end for
7: if any test rejected the null hypothesis then
8: for $i = 1,...,10$ do
9: Initialize *k + 1* clusters as the *k* previously learned plus one new cluster.
10: Run EM on the *k + 1* clusters.
11: end for
12: Retain the model of k + 1 clusters with the best likelihood.
13: Let $k \leftarrow k + 1$, and go to step 2.
14: end if
15: Every test accepts the null hypothesis; stop and return the model.

2.2.3 Gap Statistic

The Gap statistics is based on the fact that the sum of within-cluster dispersion drops (difference of the dispersion from one *k* to the next one, assuming we investigate from the smallest *k* to the largest *k*) most apparently when clustering algorithm was told the right k. Tibshirani et al. standardize the graph of within-cluster dispersion by comparing it to its expectation under an appropriate null reference distribution of the data (R. Tibshirani, T. Hastie and G. Walther 2001). Figure 3 is an illustration of the idea behind Gap statistic.

Suppose we clustered data set X into k clusters $X_r, r = 1,2,...,k$ where n_r denotes the size of each cluster. Let d_{ij} denote the squared Euclidean distance between two observations x_i, x_j. The sum of pairwise distances for all points in cluster X_r is

$D_r = \sum_{i=1}^{n_r} \sum_{j=1}^{n_r} d_{ij}$, where i, j are the indices of points in cluster X_r .

Compute W_k as

$$W_k = \sum_{r=1}^{k} \frac{1}{2 * n_r} D_r$$

The Gap statistic of k is calculated as

$$Gap(k) = E_n^*(\log(W_k)) - \log(W_k)$$

$E_n^*(\log(W_k))$ denotes expectation under the sample of size n averaged from several copies of reference distributions which contain only one single cluster.

Algorithm 5 describes Gap Statistics. Note that we estimate the $E_n^*(\log(W_k))$ by an average of B copies of $\log(W_k^*)$, each of which is computed from a Monte Carlo sample drawn from a single cluster uniform distribution. To control the sampling distribution of the Gap, we compute $sd(k)$ and s_k .

$$sd_k = \sqrt{\frac{1}{B} \sum_{b=1}^{B} (\log(W_{kb}^*) - E_n^*(\log(W_k)))}$$

$$s_k = sd_k \sqrt{1 + \frac{1}{B}}$$

We use a "one standard error" rule to choose the *k* as shown in line 5.

2.3 Existing Stability Based Model Selection Algorithms

In this section I will describe several existing model selection algorithms which are based on the concept of stability.

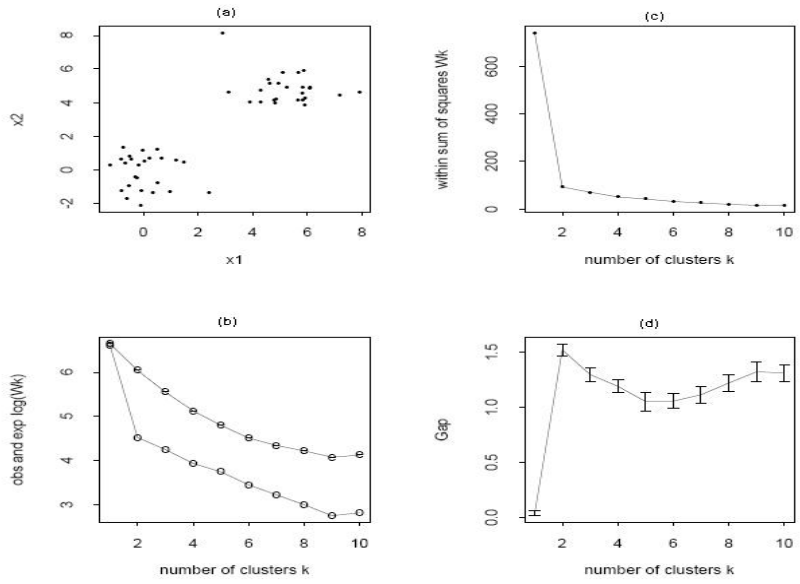

Figure 3: Illustration of Gap Statistics. The data in figure 3.a contains two clusters. The within-cluster dispersion drops most apparent with *k*=2 as shown in figure 3.c. Figure 3.b shows the comparing of within-cluster dispersion (top curve) and its expectation in single cluster null reference (bottom curve). Figure 3.d shows the computed Gap.

Algorithm 5 Gap Statistic(Dataset *X*)

Input: Data set *X*

Output: a Boolean value indicating whether *X* is a mixture or a single cluster

1: Cluster *X* with $k = 1,2,3$, giving within-cluster dispersions $W_k, k = 1,2,3$ respectively.

2: Generate B reference data sets from uniform distribution

3: Cluster each reference data set with $k = 1,2,3$ and obtain within-cluster dispersion measures $W_{kb}^*, k = 1,2,3 \; and \; b = 1,2,...,B$

3: Compute the Gap statistics for each *k* as

$$Gap(k) = \frac{1}{B}\sum_{b=1}^{B}\log(W_{kb}^*) - \log(W_k)$$

$\frac{1}{B}\sum_{b=1}^{B}\log(W_{kb}^*)$ is the $E_n^*(\log(W_k))$ we mentioned before.

4: Compute the standard deviation as

$$sd_k = \sqrt{\frac{1}{B}\sum_{b=1}^{B}(\log(W_{kb}^*) - E_n^*(\log(W_k)))}$$

And compute a quantity s_k as

$$s_k = sd_k\sqrt{1+\frac{1}{B}}$$

5: choose the smallest *k* such that $Gap(k) \geq Gap(k+1) - s_{k+1}$

2.3.1 Levin and Domany's Algorithm

Levin and Domany propose a method for estimating the number of clusters based on the comparison of stability of each *k* (E. Levine and E. Domany 2001). The stability is calculated by comparing the clustering result on resampled data with the result on the original full data. For each *k* under consideration, they first run the clustering algorithm on the data with the parameter *k* and obtain the clustering result on the full data set. Then they take a resample from the full data and run the same clustering algorithm on the sub sample with the same parameter *k* to obtain the clustering result on this sub sample. The distance between this result and the result on the full data set will be calculated using Hamming distance. They repeat this resampling process for *r* times and average the *r* distances as the stability score for this *k*. After calculating the stability scores for all *k*s under consideration, they take the *k* which has the smallest stability score as the selected model for the data set.

As we can notice here, the stability score *stab(k)* Levin and Domany calculate really means how instable the corresponding *k* is: the higher *stab(k)* is, the more the clustering results from resampling vary against the result from the full data, that is, the more instable the *k* is. The *k* with the smallest *stab(k)* leads the most stable results.

As we mentioned, measuring the distance between clustering results is always a key part for stability calculation. Levin and Domany use the Hamming distance on $N \times N$ *cluster connectivity matrices* τ_{ij}, defined by

$$\tau_{ij} = \begin{cases} 1 & i \text{ and } j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

One problem with this method is that the value of $stab(k)$ scales with *k*, which makes sense because the more clusters you have, the more possible ways there are to partition the data. Selecting the *k* without normalizing the stability score some times will choose the wrong model. Algorithm 6 is a formal description of the algorithm.

Algorithm 6 Levin and Domany's Algorithm

Input: data set X containing N observations on D attributes
Output: the *k* which best fits the data set *X*

For each value of k under consideration:
1: Run the clustering algorithm and cluster the full data set *X*
2: Repeat the resample process *r* times
  2.1: Randomly draw a sub sample from *X* and cluster it
  2.2: Calculate the distance D_r between the clustering of the full set and the one of the sub sample (Hamming distance on the connectivity matrix on sub sample)
3: Compute stability for the current *k* as $stab(k) = mean_r(D_r)$

Choose the k for which $stab(k)$ is minimal

2.3.2 Ben-Hur, Elisseeff and Guyon's Algorithm

Ben-Hur, Elisseeff and Guyon propose a stability based model explorer algorithm for visually assessing the presence of structure in clustered data (A. Ben-Hur, A. Elisseeff and I. Guyon 2002). In their method, stability is characterized by the distribution of pairwise similarities between clusterings obtained from sub samples of the data. Unlike Levin and Domany's method which choose the optimal *k* based on a quantitative stability score, Ben-Hur's algorithm explore the candidate models visually by choosing the parameter *k* for which the histogram of the similarities between resulted clusterings under this *k* is most concentrated. As a result, Levin and Domany's method is easy to implement in computer program. However it is hard to write a program to implement

Ben-Hur's algorithm because looking at the histograms and choose the most concentrated one is too complicated for a computer program.

For each *k* under consideration, the algorithm draws two sub samples X_1 and X_2 from the full data set X and clusters X_1 and X_2 with the same *k* respectively. Then these two clusterings are compared on the joint domain $X_1 \cap X_2$ and a similarity score is calculated on this joint domain. This process is repeated *r* times for each *k* and *r* similarity scores are calculated. Then for each *k* we will have a histogram of the *r* similarity scores. If the histogram is highly concentrated, it means that under resampling the clustering results do not vary very much from each other. In other words, the *k* for which the histogram is the most concentrated is the best answer for how many clusters there are in the data set.

The similarity score in Ben-Hur's algorithm is calculated as Jaccard coefficient. As we have already seen in our discussion of Levin and Domany's method, the clustering result can be represented by the *cluster connectivity matrix*. Let C_1 and C_2 represent the clustering results on sub samples X_1 and X_2 respectively and let N_{ij} for $i, j \in \{0,1\}$ be the number of entries on which C_1 and C_2 have values *i* and *j*, respectively. The matching coefficient can be easily calculated as the ratio of entries on which the two clusterings agree:

$$M(C_1, C_2) = \frac{N_{00} + N_{11}}{N_{00} + N_{01} + N_{10} + N_{11}}$$

This coefficient usually varies in a very small range because N_{00} is always the dominant factor. To use it in a more efficient way, the negative matches can be ignored, which leads to the Jaccard coefficient (P. Jaccard 1901):

$$J(C_1, C_2) = \frac{N_{11}}{N_{01} + N_{10} + N_{11}}$$

Ben-Hur's algorithm can be summarized as Algorithm 7.

Algorithm 7 Ben-Hur, Elisseeff and Guyon's Algorithm

Input: data set X containing N observations on D attributes
Output: the optimal number of clusters for X

1: For each value of k under consideration, Repeat the resample process *r* times:
  1.1: Randomly draw two sub samples X_1 and X_2, then cluster them
  1.2: Restrict clusterings to the joint domain $X_1 \cap X_2$ and calculate the two Connectivity Matrices C_1 and C_2.
  1.3: Compute the similarity score using Jaccard coefficient.
2: Generate the histogram of the *r* similarities for each *k.*
3: Choose the *k* for which the histogram of similarities is most concentrated

2.3.3 Lange, Roth, Braun and Buhmann's Algorithm

Tilman Lange and his colleagues propose another stability based model selection algorithm (T. Lange, V. Roth, M. Braun and J. Buhmann 2004). They propose a way to normalize the stability score. Their algorithm uses the same Jaccard coefficient to capture the quantity of similarity. They have their own special way of resampling. They divide the full data into two halves and cluster on them respectively. Then instead of comparing the resulted clusterings on the two halves directly, they extend each clustering to the other half data and obtain two different clusterings on the full data. The distance between these two clusterings on the full data is calculated and averaged as the stability score for the *k*. As we have discussed, the stability score scales with *k* and we need to normalize the score before we use the score to choose the answer. Lange's normalization strategy is on the split two halves data, instead of clustering them, randomly assign labels to each data point and obtain a "clustering" on each half. The obtained two clusterings are then

extended in the same way to the other half to obtain two clusterings on the full data. And a score is calculated in the same way as we calculate the stability score. This score will be used to normalize the stability score. The algorithm can be summarized as the following.

We have tried these algorithms on a large variety of data set and compared their performance. Though each of these existing stability based model selection algorithms has their own way of creating input randomization and their own way of characterizing the stability score, they are similar in performance, which means the selected *k* from these algorithms does not vary too much. We can say that the different ways of input randomization and different ways of characterizing stability are equal in power.

However, this does not mean the answers from these methods are right. In fact, there is a common phenomenon in most real data set which can not be handled by all the existing stability based methods. This is what we call the symmetric data problem, which is discussed in the next chapter.

Algorithm 8 Lange, Roth, Braun and Buhmann's Algorithm

Input: data set X containing N observations on D attributes
Output: the optimal number of clusters for X
1: For each value of k under consideration, Repeat the resample process *r* times:
  1.1: Randomly divide the data into two halves X_1 and X_2, then cluster them
  1.2: Extend both clusterings from their "half" to the other "half" using k-Nearest Neighbors algorithm.
  1.3: Compute the similarity score between the two new clusterings.
2: Average the similarities as the stability score $stab(k)$ for *k*.
For normalization:
  1: Consider the same splits as above, assign random labels to points
  2: Extend both "random clusterings" as above to the other half.
  3: Calculate the stability as $stab_{rand}(k)$
Choose the *k* such that $stab(k) \big/ stab_{rand}(k)$ is minimal

---

CHAPTER THREE

The Data Symmetry Problem

3.1 The Problem

When we evaluate the existing stability based model selection algorithms, we find that on many datasets they find a k which is much smaller than expected (with the expectation coming from prior knowledge or data exploration). After close investigation on the clustering structures discovered in these problem datasets, we find that a common property for many of these datasets is that they have symmetry across multiple groups of clusters. Figure 4 is a simple example of a dataset with symmetric clusters. The data set contains four Gaussians in two dimensions. However these four Gaussians are grouped into two vertical strips which form bigger clusters. The two leftmost Gaussians are closer to each other than to the two rightmost Gaussians, and vice-versa. Clearly, the four clusters have a symmetric and hierarchical structure. As a result, in the stability curve shown in Figure 4.b, though $k = 4$ has a relatively good stability score, $k = 2$ is more stable, even though there are four distinct clusters. We name this the symmetric data problem, because it arises from having symmetric groups of clusters. The reason that it occurs is that hill-climbing clustering algorithms (such as k-means) can more easily fall into sub-optimal solutions when k is large than when k is small. Thus, when k is small, algorithms like k-means will more often find consistent, high-quality solutions. Thus, smaller k is more likely to be stable. Going to the logical extreme, k-means with $k = 1$ is (almost surely) the most stable solution for any dataset.

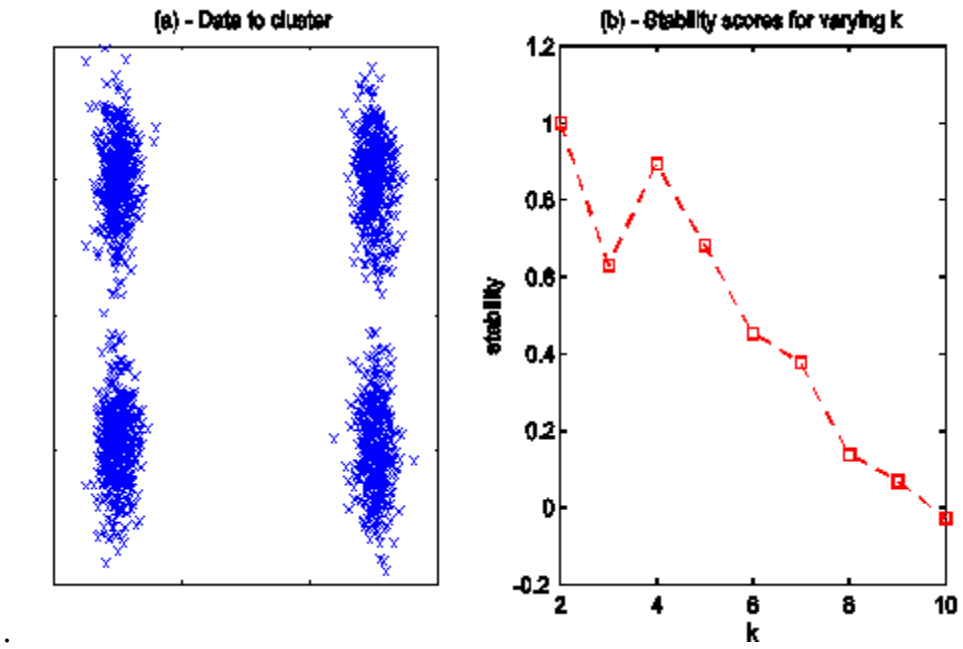

Figure 4: An Illustration of Data Symmetry. Though in figure 4.b, *k=4* have the second highest stability score, *k=2* have the highest stability score and thus will be considered as the best model fitting the data by stability.

The symmetric data problem often occurs in real-world data sets. As a result, existing stability based model selection algorithms do not work well on all the real world application data we have used. One of our primary application areas in the SimPoint project is on prediction of program performance based on code usage. We gather data on which static basic blocks are being used at different times during a program's execution. Many programs naturally have a very hierarchical and symmetric structure, due to loops, function calls, and recursion. Figure 5 are some examples of this data symmetry in several programs from the SPEC CPU 2000 benchmark suite (specifically on the programs art, vpr, and gzip). Note that art has two traces on two different inputs (110 and 470). Figure 5 shows each program along its first two principal components. Each dot represents an interval of time during execution, and the values are based on which basic

blocks are used during that interval. Each of these datasets has several giant clusters which consist of smaller clusters.

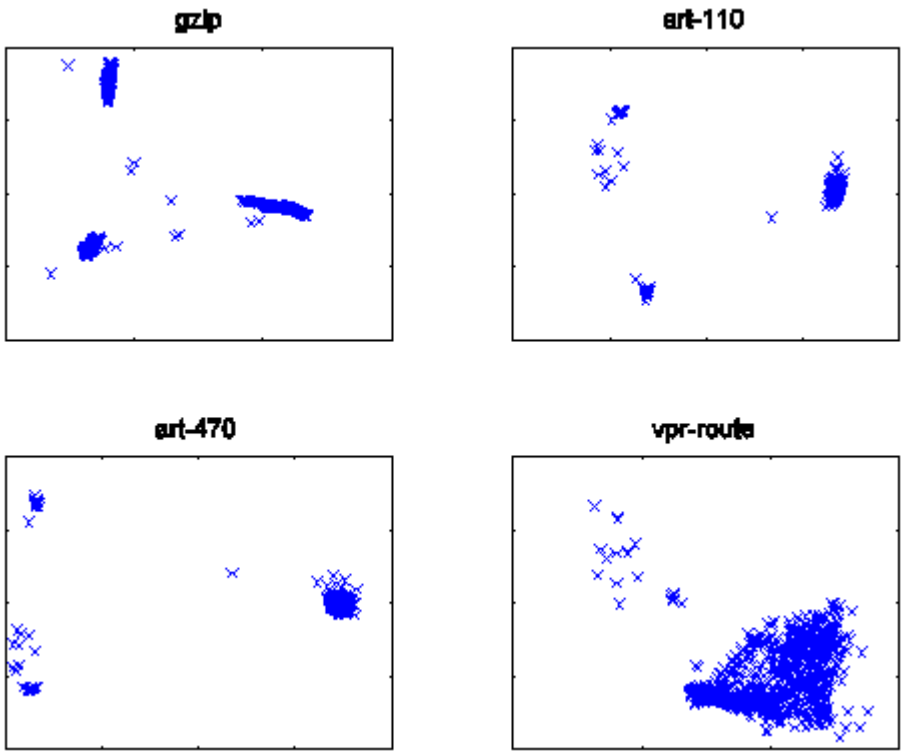

Figure 5: Data Symmetry in Program Execution Data

3.2 The Ideas for Solution

Looking again at Figure 4, we find that it makes sense to analyze the stability of the data in a hierarchical manner. That is, we accept the answer from stability analysis that $k = 2$ and then divide the data into two disjoint subsets according to that clustering. Then we analyze each of the two large clusters respectively using stability analysis to find two clusters for each. Finally we can find the four Gaussians for the data. This hierarchical stability analysis forms the basis of our algorithm.

It is important in this hierarchical stability analysis that when it comes to a single cluster, the algorithm should stop splitting this single cluster. In other words, it is

important to distinguish single cluster from a mixture of clusters. This task is generally referred as unimodality testing and remains a difficult and open issue. We adopted several testing methods and proposed our own testing methods as well.

In the next chapter, I will describe our algorithm which is based on the hierarchical stability analysis. I will discuss the details our algorithm, including the calculation of stability, input randomization, and the unimodality testing strategy we use.

CHAPTER FOUR

Hierarchical Stability Based Model Selection

4.1 The Algorithm

Our approach is called HS-means, where HS stands for hierarchical stability. Given a data set *X*, HS-means starts with unimodality test on the data to determine whether *X* is a single cluster or a mixture of multiple clusters. If the test result shows the data set is from a mixture model instead of a single component model, the method finds a stable clustering model for the data by analyzing the stability of possible models and partitions the data into several subsets (clusters) according to this stable model. Then the method recursively analyzes each subset in the same way and finds the stable model for each subset. At the termination of this recursive stability analysis, a stable clustering model for the data will be obtained by aggregating the stable models for each subset. The basic idea is recursively divide the data set into several subsets by analyzing stability until all the subsets are single clusters. Algorithm 9 describes the overview of the method more formally.

At line 1, the algorithm performs a test to determine whether *X* is unimodal. If *X* is a single component cluster, the algorithm simply returns this cluster as in lines 2 to 7. If X is from a mixture model, then the first question to estimate this mixture model is: how many components are there in this mixture? The algorithm answers this question by analyzing and comparing the stability scores of all possible *k*s under consideration, which are specified by the lower boundary *minK* and upper boundary *maxK* as shown in line 8.

Note that this selected *k* may not be right one which best fits the data since the symmetric data problem might happen. This *k* only means at the current granularity, stability scores indicate that this *k* is the most stable model. Once we find how many clusters there are in X, the algorithm will estimate the parameters of this mixture model by the Expectation-Maximization (EM) algorithm.

Algorithm 9 HS-means(Dataset X , *minK*, *maxK*)

Input: dataset X contains *N* observations on *D* attributes; *minK* and *maxK* are the lower and upper bounds of the possible number of clusters we will consider at this level of recursion.
Output: *K*-the number of clusters; *M*-centers of the clusters; *G*-membership of points to clusters.

1: *unimodal* = Unimodality Test(X)
2: if *unimodal* ≡ *true* then
3: $K = 1$
4: M = mean(X)
5: all observations belong to one cluster, *G* = indices of each point in the original data
6: return
7: else
8: k = Stability Analysis(X , *minK*, *maxK*)
9: [μs, *E*] = K-Means(X , k)
{μs are the estimated centers and *E* is the membership assignments}
10: [X_1, X_2, … , X_k] = Data Partition(X , μs, *E*)
11: end if

12: for all X_i where $i = 1,2,...,k$ do
13: [K_i, M_i, G_i] = HS-means(X_i , *minK*, *maxK*)
14: end for

15: $K = \sum_{i=1}^{k} K_i$
16: $M = \{M_1, M_2, ..., M_k\}$
17: $G = \{G_1, G_2, ..., G_k\}$

Though we choose the K-Means clustering algorithm here, it can be substituted by any other clustering algorithm. At line 9, the K-Means algorithm estimates the centers

of the clusters μs and the point-to-cluster assignments E. The estimated parameters will then be used to partition X into several subsets at line 10. Then the algorithm moves to the next level of recursion to analyze the structure of each subset as lines 12 to 14, applying the same analysis technique described from line 1s to 11. Lines 15 to 17 are the tail of the recursion which aggregates the results of the lower level recursions. Two major parts of this method are unimodality test and stability analysis. I will talk about the details of them in the following sections.

4.2 Stability Analysis

We have seen in chapter two that though failing to handle some special structures in the data like symmetric data problem, stability is still a good tool to find the number of clusters in data set. Our work is to improve this kind of methods. So we still apply stability analysis as the tool to find stable k for the data set. As we said, stability is calculated by comparing clusterings resulting from the same algorithm on the same data with input randomization like resampling or random initialization of the cluster centers. Thus to calculate stability we need a quantitative measure for the distance/similarity between clusterings. We have already seen several proposals like the Jaccard Coefficient, Hamming distance, Rand index and Mallow index which are computed base on a clustering confusion matrix. When comparing clusterings, there is commonly a problem called cluster rotation, for example, cluster 1 in clustering 2 might be called cluster 2 in clustering 2. Thus to compute the matrix, we need to mapping clusters between the two clusterings which introduces complexity. The Lange et al. talked about the mapping problem in their paper. To simplify, we use Variation of Information (VI) as a measure of similarity between clusterings.

4.2.1 Compare Clustering by Variation of Information (VI)

VI is the criterion proposed by Marina Meila for comparing two clusterings (M. Meila 2003). Let's say we have dataset *X* containing *N* observations and a clustering *C* on *X* containing *k* clusters $X_1, X_2, ..., X_k$. If we are going to randomly pick up an observation from *X*, how much uncertainty is there about which cluster the picked point is in? Let the probability that the point is in cluster X_i be $P(i)$, which is reflected by the proportion of the size of cluster X_i to the size of the full data set *X*.

$$P(i) = \frac{|X_i|}{N}$$

Then we have a discrete random variable taking *k* values $P(1), P(2), ..., P(k)$, which is uniquely associated to clustering *C*. According to information theory, the entropy of this variable is measured by $E(C)$

$$E(C) = -\sum_{i=1}^{k} P(i) \log P(i)$$

$E(C)$ is a measure of how much information clusterings *C* contains about which cluster a randomly picked point should belong to.

Now suppose we have two clusterings on the same data: C_1 which contains k_1 clusters and C_2 which contains k_2 clusters to compare. Denote by $P_1(i), i = 1,2,...,k_1$ and $P_2(j), j = 1,2,...,k_2$ the random variables associated with two clusterings C_1 and C_2 respectively. Let $P(i, j)$ represent the probability that a point belongs to cluster X_i in clustering C_1 and X_j in clustering C_2, namely the joint distribution of the random

variables associated with the two clusterings. Then we define mutual information between the two clusterings C_1 and C_2 to be $I(C_1, C_2)$

$$I(C_1, C_2) = -\sum_{i=1}^{k_1}\sum_{j=1}^{k_2} P(i, j)\log\frac{P(i, j)}{P_1(i)P_2(j)}$$

$I(C_1, C_2)$ can be interpreted in the following way: for any randomly picked point in *X*, the uncertainty about which cluster it belongs to in C_2 is measured by $E(C_2)$. Suppose we are told which cluster the point belongs to in C_1. Will this reduce the uncertainty about which cluster it belongs to in C_2? It may, depending on how similar C_1 and C_2 are. Then how much uncertainty is reduced from $E(C_2)$? $I(C_1, C_2)$ measures how much this knowledge-which cluster the point is in C_1-reduces the uncertainty about where it belongs in C_2. For example, when the two clusterings C_1 and C_2 are equal, we have

$$I(C_1, C_2) = E(C_1) = E(C_2), \qquad (1)$$

which means once we know where the point is in one clustering, we know completely where it is in the other clustering. Based on this, VI is proposed as a criterion for comparing two clusterings which can be calculated as

$$VI(C_1, C_2) = E(C_1) + E(C_2) - 2 * I(C_1, C_2) \qquad (2)$$

Algorithm 10 describes the calculation of VI score for a specified *k*. To calculate the VI score for a given *k*, we obtain *r* clusterings on the same data set by repeating the same algorithm. These *r* clusterings are compared in a pairwise manner and the pairwise VI scores are averaged as the VI score for the *k*. In algorithm 5 we use K-means as the clustering algorithm to obtain the *r* clusterings. I would like to clarify that we use K-means here just for the reason that K-means generally runs faster than other algorithms.

Since we are repeating the clustering algorithm r times here, slow algorithms will definitely slow down HS-means. K-means can be substituted by any other clustering algorithm.

Note that the computed averaged VI score for the specified k actually reflects how instable the k is. The higher the VI score is, the more different the clusterings are and the more instable this k is. To be consistent, we calculate the stability of k by taking the negative of calculated VI score.

$$stability(k) = -VI(k)$$

Finally, the value r is chosen from experience. If r is too small, the averaged VI may not be stable and the calculated stability may be inaccurate. Based on our experiments, r = 10 is a suitable value most of the time.

Algorithm 10 Calculate VI(Dataset X, r, k)

Input: dataset X contains N observations on D attributes; r is the number of clusterings compared; k is number of clusters
Output: $VI(k)$ - VI score of this k

1: for $i = 1$ *to* r do
2: C_i = K-means(X, k)
3: end for

4: for $i = 1$ *to* r do
5: for $j = i$ *to* r do
6: $VI_{ij} = E(C_i) + E(C_j) - 2 * I(C_i, C_j)$
7: end for
8: end for

9: $VI(k)$ = average VI_{ij}
10: return $VI(k)$

4.2.2 Stability Analysis

Once we have calculated the stability score for each *k* under consideration, we can select the *k* for which the stability score is maximal. Algorithm 11 describes the procedure of our Stability Analysis.

Algorithm 11 Stability Analysis(Dataset *X*, *r*, *minK*, *maxK*)

Input: dataset *X* contains *N* observations on *D* attributes, *r* is the number of clusterings that will be compared to compute VI score, *minK* and *maxK* are the lower and upper bounds of the possible number of clusters under consideration
Output: the k with the highest stability score

1: for *k* = *minK* to *maxK* do
2: *Stability*(*k*) = – Calculate VI(*X*, *r*, *k*)
3: end for
4: return *k* that has maximum stability score

4.3 Unimodality Test

As mentioned above, our HS-means method recursively partitions the data set and performs stability analysis on subsets to find a stable model for each subset. It is critical that the method should stop splitting when it comes to a subset which is a single cluster. In other words, the method must be able to tell a mixture model from a single component model. However, stability analysis assumes that the data always contains at least two clusters. Back to our definition of VI score, if the data contains one cluster and if we cluster the data into one cluster, then the resulting clusterings C_1 and C_2 are the same. According to equation (1) and equation (2), the stability for *k*=1 will be 1. In other words, *k*=1 is always the most stable model for any data set. Thus we cannot use stability score to answer the question of whether the given data set contains only one single cluster or multiple clusters.

Stability analysis can not answer this single component versus mixture question. We need a unimodality test which can detect the single component model and help the recursive stability analysis terminate. Unimodality testing is a very difficult problem in the area of statistics, which has no common agreement upon good solutions. We have investigated several typical unimodality testing strategies as well as our own unimodality testing proposals. In this section, I will talk about the details of several unimodality test approaches, which we found in the experiments can produce acceptable results.

4.3.1 Dip Statistics

The Dip test of unimodality was first proposed by Hartigan (P.M. Hartigan 1985; J.A. Hartigan and P.M. Hartigan 1985). This method is based on the fact that a distribution function F that is unimodal with mode m if F is convex in $(-\infty, m]$ and concave in $[m, \infty)$. Denote by (x_L, x_U) the modal interval, that is, where the mode of F falls in. For a given data set *X*, there are $\frac{n(n-1)}{2}$ possible modal intervals (x_i, x_j) in total. Let F_n denote the empirical distribution function of *X*. For each modal interval (x_i, x_j), compute the greatest convex minorant (g.c.m.) of F_n in $(-\infty, x_i)$ and least concave majorant (l.c.m.) of F_n in (x_j, ∞). Denote by d_{ij} the maximum distance between F_n and these computed curves. Then the minimum d_{ij} divided by 2 will be the dip value. For details of the calculation of dip test, please refer Hartigan's paper.

To use dip statistics value to test the unimodality of X, we need some unimodal distribution as a reference distribution. Normally, the Gaussian distribution is a good choice. Compare the dip value of X with the dip value of the reference distributions to get

a p-value. This p-value is an indicator of how much X looks like a single component model. Also notice that dip statistics only works in one dimension. So during the unimodality testing, we perform principal components analysis on the given data and take the projected data on the principal component to apply the dip statistics.

4.3.2 Gap Statistics

Gap statistics was proposed by Tibshirani et al. for estimating the number of data clusters, that is, as a model selection algorithm. Our experience with Gap statistics show that this method only works well when the number of clusters is small. However, the idea behind Gap statistics can still be adapted to test whether the given data contains one single cluster or not. We do not really use Gap statistics to find how many clusters are there in the data. We just calculate the gap statistics of *k=1* and *k=2* respectively and compare the two value to test whether the data contains more than one cluster or not. This is a simple application of gap statistics.

Algorithm 12 Gap Unimodality Test(Dataset *X*)

Input: Data set *X*
Output: a Boolean value indicating whether *X* is a mixture or a single cluster

1: Calculate gap for *k=1*and *k=2* using the same method discussed in Chapter Two
2: if $k \equiv 1$ then
3: *X* is single component cluster, return true.
4: else
5: *X* is a mixture, return false.

4.3.3 χ^2 *Unimodality Test*

One of the most famous properties of Gaussian distribution is that the sum of squares of *d* independent Gaussian variables follows χ^2 distribution with degrees of freedom *d* (S. Kotz, N. Balakrishnan and N.L. Johnson 2000). This simple fact can be

used to test the unimodality of data set X with the assumption that X is from a multivariate Gaussian or a mixture of multivariate Gaussians. Assume the data set X which contains N observations on d attributes $(x_1, x_2, ..., x_d)$ is from a multivariate Gaussian distribution in d dimensional space, and further assume the covariance matrix is a identity matrix, if we sum the squares of each attribute we will have a χ^2 distribution with degrees of freedom d.

For each data objects x_i in X we can compute S_i as

$$S_i = [x_{i1} - \hat{\mu}_1]^2 + [x_{i2} - \hat{\mu}_2]^2 + ... + [x_{id} - \hat{\mu}_d]^2$$

Where $\hat{\mu}_j$ represents the estimated mean of X along dimension j.

If X is indeed approximately from the multivariate Gaussian distribution we mentioned above, $S = \{S_1, S_2, ..., S_n\}$ will be a sample drawn from a χ^2 distribution with freedom d. We also note the fact that when X is from a mixture of multivariate Gaussians, the sample S will not follow the χ^2 distribution with freedom d, as shown in Figure 6. Unfortunately, we do not know yet what the distribution of S is when X is from a mixture of multivariate Gaussians. Thus we can use a hypothesis test to check whether S follows a χ^2 distribution of freedom d. If yes, we consider X as a single component cluster and otherwise consider X as a mixture model.

Algorithm 13 describes the procedure our χ^2 unimodality test. In our test approach described above we assume that if X is from a single multivariate Gaussian, then the covariance matrix of that multivariate Gaussian is an identity matrix, that is, the d attributes are independent of each other. So before we calculate the S, we first need to normalize the given data set and force it to have an identity covariance matrix. Our

approach for normalization is centering the data to have sample mean of zero and projecting the data along each of the *d* eigenvectors and dividing by square root of the corresponding Eigen value. After normalization, the data set will look spherical and have identity matrix as the covariance.

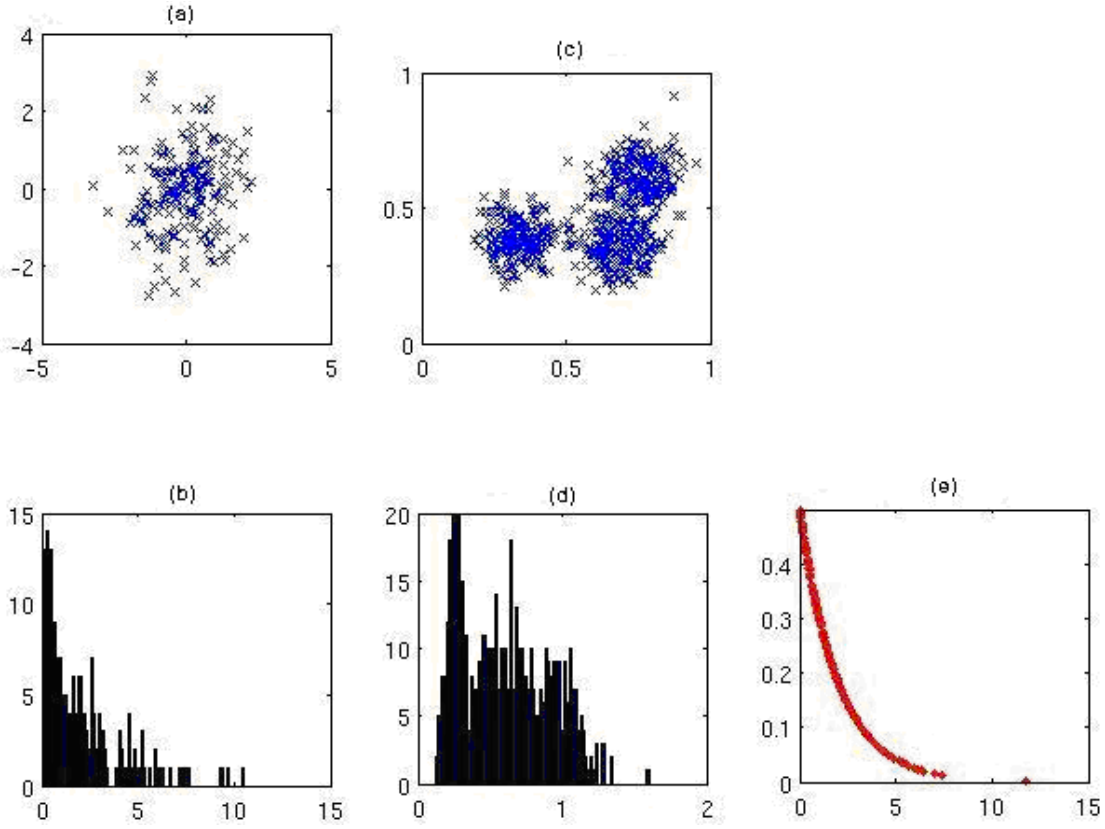

Figure 6: The Distribution of Sum of Squared Gaussians. Figure 6.a and figure 6.c are two data sets in two dimensions drawn from a single multivariate Gaussian and a mixture of multivariate Gaussians respectively. Figure 6.b and figure 6.d are the histogram of S respectively. It is clear that S of single multivariate Gaussian follows the expected χ^2 distribution shown in figure 6.e while the S of the mixture does not.

Algorithm 13 χ^2 unimodality Test(Dataset *X*)

Input: Data set *X*
Output: A Boolean value indicating whether *X* is from a single multivariate Gaussian or not

1: center and spherize the data
2: Calculate *S* for *X*.
3: Do a ks-test on whether *S* follows χ^2 distribution with freedom d
4: if ks-test accepts null hypothesis then
5: X is a single multivariate Gaussian.
6: else
7: X is from a mixture
8: end if

CHAPTER FIVE

Experiment Evaluation

We evaluated the performance of HS-means on several data sets and compared the performance with the competing methods we discussed in chapter four. The results shows that HS-means is competitive to PG-means and out-performs the existing stability based model selection algorithms, Gap statistics and G-means. We compared the performance on both generated data sets and two real world data sets.

5.1 Performance on Generated Data Sets

We first evaluate the performance of our algorithm on several generated data sets which vary in underlying statistical distribution, number of clusters, number of dimensions, and the degree of overlapping. The data sets are generated from both Gaussian mixtures and uniform mixtures in both low dimension and high dimension, including both well separated clusters and highly overlapped clusters. Especially, we reproduce the symmetric data problem and evaluate how well our HS-means and other competing methods perform on the symmetric data sets.

5.1.1 Generating the Data Sets

A single cluster of data points is generated from a single Gaussian distribution. Our generated Gaussian sample can be either spherical or elongated. To make the generated data set looks elongated, we make a $d \times d$ scaling matrix S which has certain eccentricity specified by a parameter *ecc* such that

$$\sqrt{\frac{\max(S)}{\min(S)}} = ecc$$

where $\max(S)$ and $\min(S)$ are the maximum and minimum variance (standard deviation) of S in any axis-aligned direction. The scaling matrix is equivalent to the square root of diagonal matrix of Eigen values. To rotate the generated Gaussian, we construct a random orthogonal basis as a $d \times d$ matrix. The generated single Gaussian sample is computed as

$$X = Y * S * R'$$

where Y is a random sample from a Gaussian with identity matrix as covariance matrix, S is the scaling matrix and R is the rotation basis.

When we generate a mixture of Gaussians, each Gaussian in the mixture is generated in the same way as above. To control the degree of overlapping between the Gaussian components in the mixture, we use the definition of *c-separation* (S. Dasgupta 2000). Two Gaussians $N(\mu_1, \sum_1)$ and $N(\mu_2, \sum_2)$ in $\Re^n$ are *c-separated* if

$$\| \mu_1 - \mu_2 \| \geq c\sqrt{\max\{trace(\sum_1), trace(\sum_2)\}}$$

A mixture of Gaussians is *c-separated* if its component Gaussians are pairwise *c-separated.*

In the similar way, we generate mixtures of Uniform clusters.

5.1.2 Algorithm Performance

Figure 7 is an illustration of how HS-means performs over a generated Gaussian mixture. The generated data, as shown in Figure 7.a contains six clusters in two dimensions with each cluster having 200 data points. The eccentricity of each Gaussian is 2. Figure 7.b is the stability curve of this whole mixture. In the stability curve, *k=3* and

k=6 are the two peak of the curve which means the mixture might contain three clusters or six clusters. However, the symmetric data problem happens, that is, *k=3* has the higher stability score than *k=6*. Thus from the stability curve, we choose k=3 as the best result of the top level stability analysis. Figure 7.c shows how the whole mixture is divided into three clusters. The subset with green color contains two clusters and the subset with red color contains three clusters. In the next lower level of stability analysis, these two bigger clusters will be divided into two small clusters and three small clusters, respectively. Together, six clusters are found after the second level of stability analysis. The unimodality test accepts these six clusters as single Gaussian clusters. Thus the original data are clustered into six clusters as the red crosses indicating in figure 7.a.

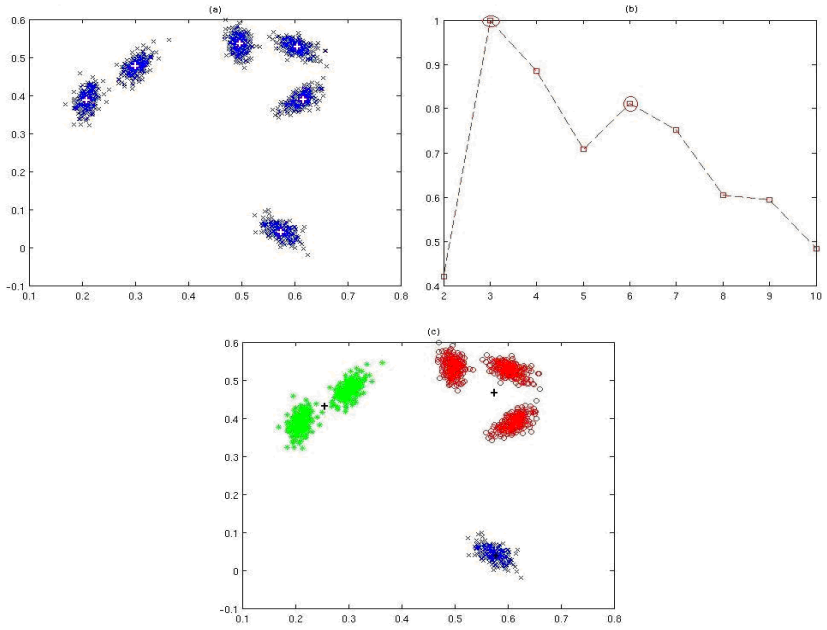

Figure 7: An Illustration of HS-means. The data set contains 6 generated clusters Gaussian mixture in two dimensions. Figure 7.a shows the six clusters found by HS-Means with the crosses indicating the centers of the clusters. Figure 7.b shows the stability curve in the first level of analysis. *k=3* has a better stability score than *k=6*. Figure 7.c shows how the Lange's algorithm will partition the data set based on only one level of stability analysis.

We also run the competing methods on several generated data sets and compared the results with our HS-means. Table 1 is a summary of the comparisons. In the generated data sets we used for these comparisons, each data set contains 10 clusters with each cluster containing 200 data points. The c-separation we used is 3. The data might contain overlapping clusters, but not highly overlapped. For each category of the test data, we run the test for 20 times.

Table 1: Learned *k* on Different Synthetically Generated Data Sets with Correct *k=10*

	Gaussians 3 dimensions	Gaussians 16 dimensions	Uniforms 3 dimensions	Uniforms 16 dimensions
G-means	10.0476 ± 0.8047	10 ± 0	20.25 ± 2.5521	14.3 ± 1.8972
PG-means	10 ± 0.6666	9.6 ± 1.3498	11.7 ± 1.2517	10.15 ± 0.7436
Gap	4.3 ± 3.1222	4.2 ± 2.6583	3.9 ± 2.846	4.7 ± 2.5976
Lange's	6 ± 3.559	10.8 ± 1.032	6.2 ± 2.8597	9.8 ± 1.025
Levin's	6.32 ± 3.258	10.5 ± 1.521	5.92 ± 2.942	8.8 ± 1.35
HS-means	10.01 ± 0.325	9.95 ± 0.263	10 ± 0.048	10.6 ± 0.52

As we can see from the table, roughly speaking, PG-means and our HS-means have equivalent performance, which are the two best methods among all the methods we tested. The learned *k* from the existing stability based model selection methods are smaller than the correct one due to the symmetric data structure. Note that our data generating schema generally does not produce symmetric data problem in high dimensions. This is why in the table, for Gaussian mixture and Uniform mixture in 16 dimensions, the existing stability based methods does not perform as bad as with data in low dimension. But in the real data set we tested later, the data symmetry does happen and those existing stability based algorithms does not perform well. As we said, G-means has a strong assumption that the data comes from Gaussians, so G-means performs very bad with the Uniform mixture data we generated.

5.1.3 Performance on the Symmetric Data Sets

Let's see how our HS-means handles the symmetric data set. To illustrate this, we generated two typical symmetric data sets as shown in Figure 8.a, where the data set contains four Gaussians grouped into two larger subsets, and Figure 8.d, where the data set contains nine Gaussians grouped into three larger subsets.

We compared our results with the stability based model selection algorithm proposed by Tilman Lange et al. Figure 8.a and Figure 8.d shows that our HS-means can find correctly four true clusters and nine true clusters respectively. However, as shown in Figure 8.b and Figure 8.e, Lange's algorithm can only find two clusters for data set in Figure 8.a and three clusters for data set in Figure 8.d. The stability curves in Figure 7c and Figure 8.f shows why this happens. In Figure 8.c, though $k=4$ does have a good stability score, $k=2$ has a better score because the data set looks like two clusters in the view of stability. Our algorithm will further analyze the two large cluster and discover a better model to fit the data while Lange's algorithm will just stop with $k=2$. Similarly, in Figure 8.f, $k=3$ is more preferred than $k=9$ by Lange's stability analysis while our algorithm can find exactly $k=9$.

This wrong behavior of single level stability analysis does not only happen in Lange's algorithm. After our investigation, we find that this wrong behavior happens in all the existing stability based model selection algorithms.

5.2 Performance on Real World Data Sets

5.2.1 Handwriting Digits Data Set

This data set contains the normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service (obtained from http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html). There are 7291 training examples and 2007 test examples, and each example is a 16 x 16 grayscale image. We merge the training set and testing set together as our data set for testing HS-means. The examples for each digit from 0 to 9 are distributed as follows:

Table 2: Distribution of Examples for Handwriting Data Set

data	0	1	2	3	4	5	6	7	8	9	Total
Train	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

The data set is a commonly used data for clustering and classification algorithms. It is not consistent that how many true clusters there are in the data. An intuitive answer is that the data contains 10 true clusters since it is a data set for 10 digits from 0 to 9. However, since different people have different way of writing the same digit, it is hard to have a common standard for how many clusters are exactly presented in the data. We investigated the images representing each digit respectively and discovered that, at least, interpreting the data as 11 clusters is better than interpreting the data as 10 clusters since there are, roughly speaking, two different style of writing digit 0: thin and tall 0 versus fat and round 0. Thus in our experiments, we use $k=11$ as the number of true clusters for the handwriting data set though there might be more true clusters in fact. Put it in one word,

our data set is a 256 dimensional data set containing 9298 data points in 11 clusters. We don't expect the data is from a mixture of Gaussians or a mixture of Uniforms.

5.2.2 Synthetic Control Curves Data Set

This dataset contains 600 examples in 60 dimensions of control charts(obtained from UCI machine learning repository) which are synthetically generated by the process in Alcock and Manolopoulos (R.J. Alcock and Y. Manolopoulos 1999). Each instance in the data set represents a time series. There are six different clusters of control charts: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift as shown in Figure 9. The data set contains six true clusters with each cluster contains 100 data points.

5.2.3 Data Projection

As we have seen in the description of the Handwriting digits data set and the synthetic control charts data set, the dimension of these data sets are extremely large. To reduce the time of running tests on these data sets we perform data projections to reduce the dimension and run the algorithms on the projected low dimensional data. Generally, two projection methods can be used here: random linear projection (S. Dasgupta 2000) and principal components analysis (I.T. Jolliffe 2002). Random linear projection is fast and easy. PCA is more expensive, but can give really good results which present the structure of the data very well.

For the Handwriting digits data set, we project the original 256 dimensional data into 16 dimensional data and for the synthetic control charts data we project the original 60 dimensional data into 10 dimensional data. Figure 10 shows how the data sets look

like in their first two principal components. As we can observe from this figure, both the two data sets have data symmetry. In figure 10, the observed two large cluster in Handwriting data set and the observed three large cluster in synthetic control charts data set all contain several clusters inside them. Those small clusters are not visible here because this figure just illustrates the first two principle components.

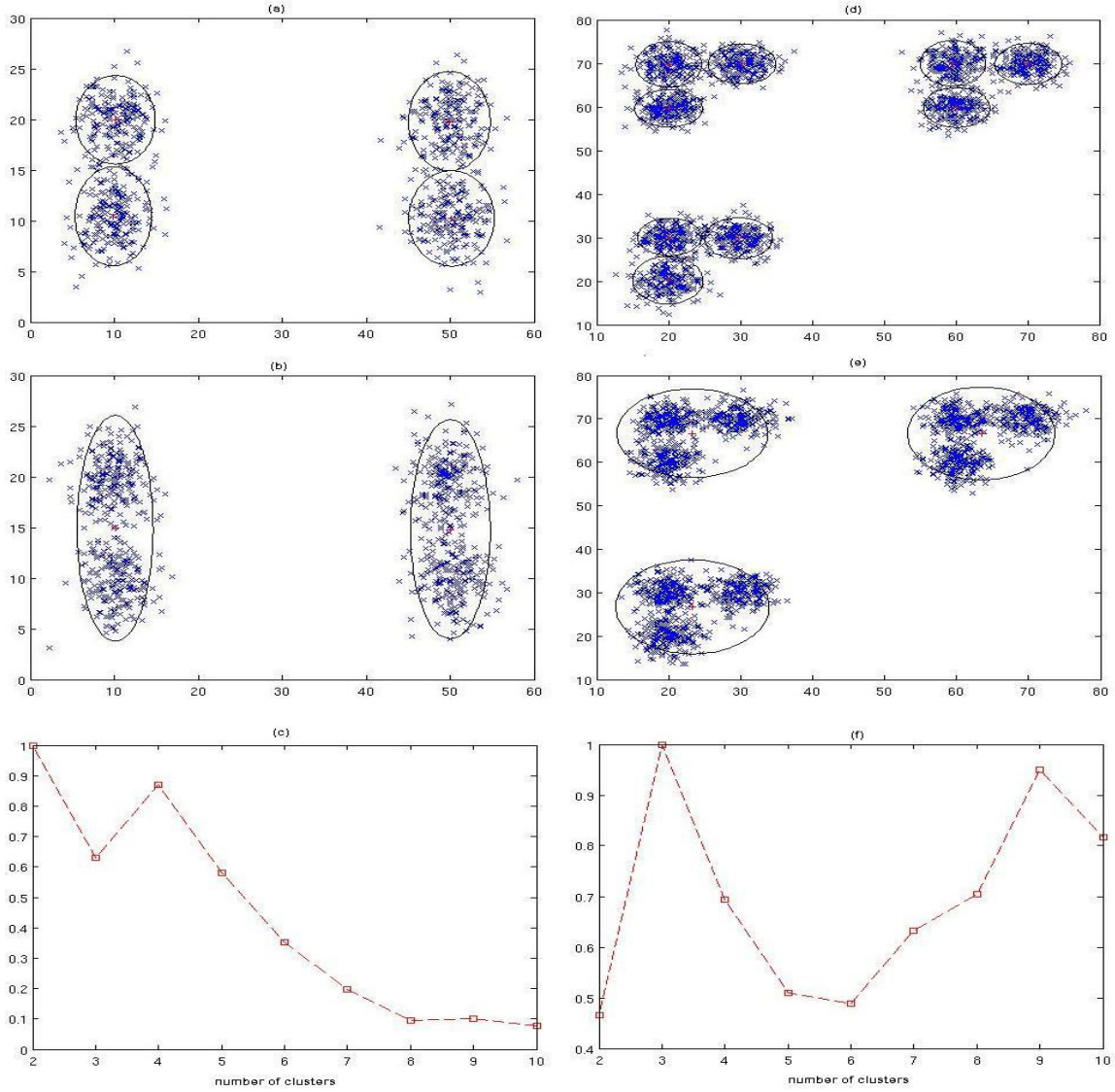

Figure 8: HS-means versus Lange's Algorithm on Symmetric Data Sets. Figure 8.a and Figure 8.d show the clustering results from HS-means. Figure 8.b and Figure 8.e show the clustering results from Lange's algorithm. The stability curves of Lange's algorithm are shown in Figure 8.e and Figure 8.f respectively.

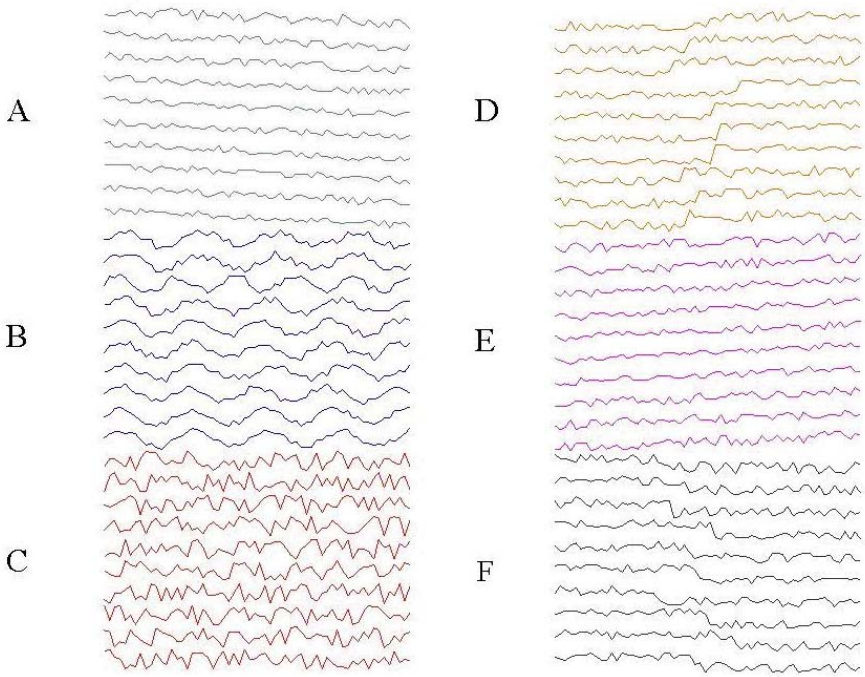

Figure 9: Six Clusters in Control Charts. From A to F are normal, cyclic, increasing trend, decreasing trend, upward shift and downward shift, respectively.

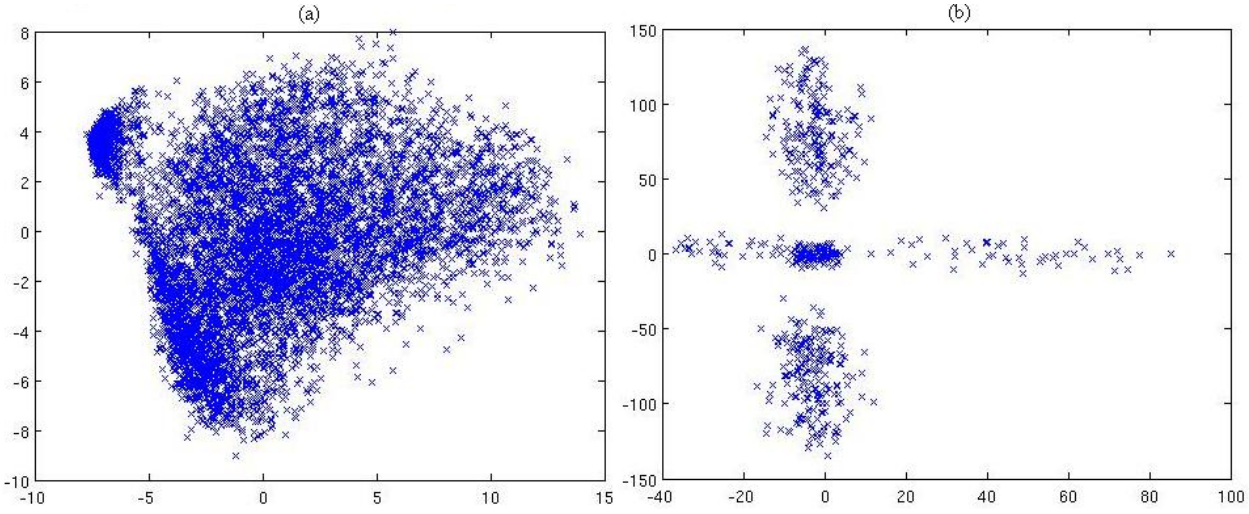

Figure 10: PCA Illustration on Handwriting Data and Control Charts Data. The first two principal components of Handwriting digits data set (figure a) and KDD synthetic control charts data set (figure b).

5.2.4 Algorithm Performance

As we can see from figure 10, both the two testing data set has symmetric data structure. The handwritten digits dataset forms two large clusters. Each of the two large clusters has several clusters in it, which are not visible due to projection (the true data lives in 256 dimensions). Similarly, the Synthetic Control Charts dataset contains three large clusters visible with the two principle components. This data symmetry leads Lange's algorithm to perform poorly. However, HS-means is able to find high quality clusterings that use more appropriate k values for the datasets at hand. We also verified that the algorithms both give reasonably good (low) VI scores relative to the true labels, but HS-means was significantly better on the handwritten digits dataset. As a result, running Lange's stability based model selection algorithm only finds two clusters in Handwriting Digits data set and three clusters in Synthetic Control Charts data set, as we expected. HS-means performs well on the Synthetic Control Charts data set, finding around eight clusters in the data which is the closest to the true number of clusters. PG-means does not work as well as HS-means on the Synthetic Control Charts data set which probably is because the clusters in the data set are not from Gaussian distribution. PG-means and HS-means have similar performance on the Handwriting Digits data set. The results on Handwriting Digits data set, though larger than the true number of clusters, are more acceptable than the result from Lange's algorithm. To better understand the learned clusterings, we calculate the VI score between the learned clustering and the true labels. These k and VI scores are calculated by running the experiments for 20 times. The lower the VI score, the better the learned clustering.

Table 3: Learned *k* on Handwriting Data and Control Charts Data

	Handwriting Digits		Synthetic Control Charts	
	learned k	VI	learned k	VI
HS-means	18.2 ± 2.023	2.23 ± 0.06	8.5 ± 1.5	1.85 ± 0.15
Lange's Stability	2.0 ± 0.0	3.02 ± 0.12	3.0 ± 0.0	2.56 ± 0.25
PG-means	17.2 ± 2.32	2.25 ± 0.02	15.2 ± 2.042	2.06 ± 0.36

5.3 Experiments on SimPoint

We also evaluated our algorithm using large program traces from SPEC CPU 2000. These come from our work on the SimPoint project, where the goal is to predict the performance of a program on a novel CPU without performing costly detailed simulation on the entire program. Instead, we profile the execution patterns (count basic blocks used over time intervals) of a program, and cluster this data to identify key regions for detailed simulation. By doing this, we can usually predict key performance statistics such as CPI and branch miss rates with less than 5% error, based on executing a very small portion of the program (e.g. less than 1%). This saves months of time when doing exploratory architecture comparisons which require a simulation across dozens of benchmarks for hundreds or thousands of potential CPU configurations. The current SimPoint approach uses k-means and a scoring system based on the Bayesian information criterion (BIC) to choose k; however, we desire a more statistically robust approach. Therefore, we applied HS-means to several benchmarks. We are willing to spend more time clustering benchmark trace data, since detailed simulation is extremely slow, and these clustering results will be used many times over many CPU configurations.

We compared HS-means and Lange's stability algorithm on the four benchmarks art-110, art-470, vpr-route, and gzip-graphic. These applications have been compiled for and profiled on an Alpha ISA processor. Each of these datasets is very high-dimensional

(from 2039 dimensions for art-110 to 11732 dimensions for vpr-route). We reduce the dimension of these datasets via random linear projection to 15 dimensions. The number of objects in each dataset ranges from 418 for art-110 to 1038 for gzip-graphic. Given the program traces, we use HS-means and Lange's algorithm to compute a clustering for each. We allow each algorithm to select k in the process. Then we use the clustering to label each example in the original dataset, and evaluate the clusterings using measurements on the CPI performance metric, which was obtained via slow, detailed simulation of these benchmarks.

In general, HS-means selects a larger k, indicating that it is detecting more fine-grained behavior in these programs. In Table 4 we see that in all cases, HS-means is able to reduce the within-cluster CPI variance by a larger amount, indicating that the clusters it is finding are more uniform in their CPI, which is a good indication of a quality clustering. The within-cluster variance is a weighted average of the variance within each of the clusters, where the weights come from the relative cluster sizes. In Table 5, we investigate the ability of each algorithm to correctly predict the true average CPI of the whole program based only on samples of k examples (one from each cluster). We compute the weighted average CPI of the chosen examples (which are called 'SimPoints'), where again the weights come from the cluster sizes, and then we calculate the error with respect to the true whole-program average CPI. Having accurate CPI predictions is important for making decisions on new CPU designs. In three of the four benchmarks, HS-means predicts CPIs much more accurately than Lange's algorithm. This is because Lange's algorithm cannot handle the data symmetry problem which

occurs in these datasets. As a result, Lange's algorithm cannot learn the k which better fits the data and obtain the clustering which better classifies the program execution data.

Table 4: SPEC CPU 2000 Benchmark Results on CPI Variance Reduction. For each benchmark we calculate the weighted average within-cluster CPI variance. The 'reduction' column is one minus the clustered CPI variance divided by the whole-program CPI variance. A better clustering will reduce the CPI variance more.

benchmark	whole-program CPI variance	HS-means k	HS-means Clustered CPI variance	HS-means reduction	Lange k	Lange Clustered CPI variance	Lange reduction
art-110	.01956	5	.00174	.91084	2	.00293	.85009
art-470	.01916	4	.00277	.85520	2	.00284	.85197
vpr-route	.12242	14	.05926	.51597	2	.08970	.26723
gzip-graph	.00363	6	.00320	.11820	3	.00319	.11792

Table 5: SPEC CPU 2000 CPI Prediction Results. We compute a weighted CPI prediction for each benchmark using the clusterings from each algorithm. Lower error is better.

benchmark	whole-program CPI	HS-means k	HS-means predicted CPI	HS-means error	Lange k	Lange predicted CPI	Lange error
art-110	2.156	5	1.158	.0011	2	2.190	.0155
art-470	2.145	4	2.146	.0090	2	2.167	.0100
vpr-route	1.455	14	1.315	.0960	2	1.432	.0156
gzip-graph	0.509	6	.506	.0049	3	.504	.0096

CHAPTER SIX

Conclusion and Future Work

The concept of clustering stability was proposed by statisticians and researchers in machine learning as a tool to automatically discover the number of clusters which best fits the given data. Several model selection methods based on this concept have been developed. However, these existing stability based model selection methods can not handle the data which contain symmetric data structure. We proposed a method which analyzes the clustering stability on multiple hierarchical levels to handle this problem. The experiments show that our proposed method solves the symmetric data problem well and out performs the existing stability based methods. The experiments on SimPoint data shows HS-Means is able to find reasonably good clustering which is comparable to current SimPoint algorithm. On the other hand, Lange's algorithm, one of the existing stability based model selection algorithms, performs less well on the data.

In the existing stability based model selection algorithms, most of them use resampling strategy to obtain different clusterings on the same data which are compared to compute the stability of the model. During the investigation in our research, we find that resampling is not necessary, at least not the only possible method, to achieve this. We use randomization of the initial positions of the cluster centers to introduce input randomization to the clusterings which works well in our algorithm.

One key part of our method is detecting the single cluster using unimodality testing techniques. The quality of these unimodality testing techniques is the bottle neck

of the method. The future improvements could be made to our method by developing a better unimodality testing method. We investigated several unimodality testing methods during our research. None of the methods we investigated works perfect in all situations. The unimodality testing still remains as a challenging problem in statistics.

BIBLIOGRAPHY

H. Akaike. A new look at the statistical model identification. IEEE Transactions on Automatic Control 19 (6): 716–723, 1974.

R.J. Alcock and Y. Manolopoulos Time-Series Similarity Queries Employing a Feature-Based Approach. 7th Hellenic Conference on Informatics. August 27-29. Ioannina,Greece 1999.

T.W. Anderson and D.A. Darling. Asymptotic theory of certain “goodness-of-fit” criteria based on stochastic process. Annals of Mathematical Statistics 23: 193-212, 1952.

D. Arthur and S. Vassilvitskii. K-means++: the advantage of careful seeding. In proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 1027-1035, 2007.

S. Ben-David, U. von Luxburg, and D. Pal. A sober look at clustering stability. COLT, pages 5–19, 2006.

S. Ben-David, S. Pal and H.U. Simon. Stability of k-means clustering. In COLT, Lecture Notes in Computer Science, pp. 20-34, 2007

A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. Pacific Symposium on Biocomputing, 7:6–17, 2002.

J.C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell, MA, 1981.

A. M. Dan Pelleg. X-means: Extending k-means with efficient estimation of the number of clusters. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

S. Dasgupta. Experiments with random projection. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), pages 143–151. Morgan Kaufmann Publishers, 2000.

A.P. Dempster, N.M. Laird and D.B. Rubin. Maximum Likelihood from incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society Series B 39(1):1-38, 1977.

S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. Genome Biology 3(7), 2002.

Y. Feng and G. Hamerly. PG-means: learning the number of clusters in data. In Proceedings of the twentieth annual conference on neural information processing systems (NIPS), December 2006.

E. Fowlkes and C. Mallows. A method for comparing two hierarchical clusterings. Journal of the American Statistical Association, 78:553-584, 1983.

G. Hamerly and C. Elkan. Learning the k in k-means. In Proceedings of the seventeenth annual conference on neural information processing systems (NIPS), pages 281–288, 2003.

G. Hamerly, E. Perelman, J. Lau, T. Sherwood, and B. Calder. Using machine learning to guide architecture simulation. Journal of Machine Learning Research, 7:343–378, 2006.

J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

J.A. Hartigan and P.M. Hartigan. The Dip Test of Unimodality. Annals of Statistics 13: 70-84, 1985.

P.M. Hartigan. Computation of the Dip Statistic to Test for Unimodality. Applied Statistics 34: 320-325, 1985.

T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference and prediction. New York: Springer-Verlag, 2001.

P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin del la Société Vaudoise des Sciences Naturelles 37, 547-579, 1901.

I.T. Jolliffe. Principal Component Ananlysis. Springer Series in Statistics Second Edition, Springer, NY, 2002.

S. Kotz, N. Balakrishnan and N.L. Johnson. Continuous Multivariate Distributions Volume 1: Models and Applications. Wiley-Interscience, 2000.

T. Lange, V. Roth, M. Braun, and J. Buhmann. Stability based validation of clustering solutions. Neural Computation, 16:1299–1323, 2004.

E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. Neural Computation, 13:2573–2593, 2001.

E. Levine. Unsupervised estimation of cluster validity—methods and applications. Master's thesis, Weizmann Institute of Science, 1999.

S.P. Lloyd. Least square quantization in PCM. IEEE Transactions on Information Theory 28(2):129-137, 1982.

M. Meila. Comparing clusterings by the variation of information. COLT, pages 173–187, 2003.

A. D. R. McQuarrie and C.-L. Tsai. *Regression and Time Series Model Selection*. World Scientific, 1998.

F. J. Massey, Jr. The Kolmogorov-Smirnov test for goodness of fit. Journal of the American Statistical Association, 46(253):68–78, 1951.

A.Y. Ng, M.I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. In advances in Neural Information Processing System 14: Proceeding of the 2001 Conference, pages 849-856, 2001.

W.M. Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66: 846-850, 1971.

G.E. Schwarz. Estimating the dimension of a model. Annals of Statistics 6(2): 461-464, 1978

R. Tibshirani, T. Hastie, and G. Walther. Estimating the number of data clusters via gap statistics. Journal of the Royal Statistical Society B, 63:411–423, 2001.

J.H. Ward Jr. Hierarchical Grouping to Optimize an Objective Function. Journal of the American Statistical Association 58 (301): 236-244, 1963.

M. Welling and K. Kurihara. Bayesian k-means as a ‘maximization-expectation’ algorithm. In SIAM conference on Data Mining, 2006.