วิธีสเกลลักษณะประจำสำหรับขั้นตอนวิธีค่าเฉลี่ยเคควบคุมด้วยความผิดพลาดจากการจำแนกประเภทของทุกกลุ่ม

นาย วัชรศักดิ์ ศิริเสรีวรรณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

ATTRIBUTES SCALING FOR K-MEANS ALGORITHM CONTROLLED BY MISCLASSIFICATION OF ALL CLUSTERS

Wacharasak Siriseriwan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computational Science
Department of Mathematics
Faculty of Science
Chulalongkorn University
Academic Year 2008

Thesis Title	ATTRIBUTES SCALING FOR K-MEANS ALGORITHM CONTROLLED BY MISCLASSIFICATION OF ALL CLUSTERS
By	Mr.Wacharasak Siriseriwan
Field of Study	Computational Science
Thesis Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat)

THESIS COMMITTEE

.....Chairman
(Assistant Professor Jaruloj Chongstitvatana, Ph.D.)

.....Advisor
(Assistant Professor Krung Sinapiromsaran, Ph.D.)

.....Examiner
(Associate Professor Pornchai Satravaha, Ph.D.)

.....Examiner
(Associate Professor Paisan Nakmahachalasint, Ph.D.)

วัชรศักดิ์ ศิริเสรีวรรณ : วิธีสเกลลักษณะประจำสำหรับขั้นตอนวิธีค่าเฉลี่ยเคควบคุมด้วยความผิดพลาดจากการจำแนกประเภทของทุกกลุ่ม. (ATTRIBUTES SCALING FOR K-MEANS ALGORITHM CONTROLLED BY MISCLASSIFICATION OF ALL CLUSTERS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. กรุง สินอภิรมย์สราญ, 95 หน้า.

การแบ่งกลุ่มค่าเฉลี่ยเค ซึ่งเป็นหนึ่งในวิธีที่รู้จักโดยทั่วไปในการแบ่งกลุ่มด้วยระยะทาง เป็นวิธีการเรียนรู้ของเครื่องแบบไม่มีผู้สอนที่เป็นที่นิยมมาก และใช้ในงานประยุกต์ต่างๆอย่างหลากหลาย มีงานวิจัยในอดีตที่จะรวมแนวคิดของการเรียนรู้แบบมีผู้สอนเข้าไปในการแบ่งกลุ่มแบบค่าเฉลี่ยเค โดยผ่านเวกเตอร์ที่สเกลลักษณะประจำ ด้วยการเพิ่มเวกเตอร์นี้ การแบ่งกลุ่มแบบค่าเฉลี่ยเคสามารถกำกับดูแลด้วยข้อมูลของชั้นเป้าหมายซึ่งได้จัดเตรียมไว้ในข้อมูลสอน ในวิทยานิพนธ์นี้ เราค้นและเสาะหาเวกเตอร์สเกลลักษณะประจำที่เหมาะสมที่มีความผิดพลาดจากการจำแนกประเภทจากชั้นเป้าหมายต่ำที่สุด วิทยานิพนธ์นี้ใช้เทคนิคการหาค่าเหมาะที่สุดไม่เชิงเส้นแบบไม่มีเงื่อนไขบังคับในปริภูมิของสเกลลักษณะประจำสองแบบคือ วิธีการพิกัดวัฏจักรและวิธีของฮุคและจีพส์ การทดลองของเราแสดงให้เห็นว่าทั้งสองวิธีให้เวกเตอร์สเกลที่เหมาะสม ซึ่งสามารถลดความผิดพลาดจากการจำแนกประเภทของการแบ่งกลุ่มค่าเฉลี่ยเคแบบมีผู้สอนได้อย่างมีประสิทธิภาพ และยังนำไปสู่การจัดกลุ่มแบบมีผู้สอนที่ได้ผลกับข้อมูลบางชุด แต่ในบางชุดข้อมูลความผิดพลาดไม่เปลี่ยนไปอย่างมีนัยสำคัญ ซึ่งสะท้อนให้เห็นว่าชุดข้อมูลดังกล่าว ไม่เหมาะสมในการที่จะทำการแบ่งกลุ่มแบบมีผู้สอน

ภาควิชา...คณิตศาสตร์..... ลายมือชื่อนิสิต.....

สาขาวิชา..วิทยาการคณนา.....ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา ...2551.....

4872454023 : MAJOR COMPUTATIONAL SCIENCE
KEYWORDS : K-MEANS CLUSTERING / ATTRIBUTE-SCALED SPACE / UNCONSTRAINED OPTIMIZATION / SUPERVISED CLUSTERING / MISCLASSIFICATION RATE

WACHARASAK SIRISERIWAN : ATTRIBUTES SCALING FOR K-MEANS ALGORITHM CONTROLLED BY MISCLASSIFICATION OF ALL CLUSTERS. ADVISOR : ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., 95 pp.

K-means clustering, one of the well-known distance-based clustering methods, is a very popular unsupervised machine learning using in various applications. Researchers try to integrate the concept of supervised learning to K-means clustering via attribute-scaling vector. With addition of this vector, K-means clustering can be supervised by the information of target class provided in the training set. In this thesis, we explore and determine the optimal attribute-scaled vector that minimizes the misclassification rate of the target class. This thesis uses the non-linear unconstrained optimization techniques in attribute-scaled space, called cyclic coordinate method and Hooke and Jeeves method. Our experiments show that both methods can provide the optimal scaling vectors which effectively reduce the misclassification error of supervised K-means clustering and lead to the effective supervised clustering in some data sets. For other data sets, the improvement of misclassification error is still achievable, but the error is too high suggesting that those datasets are not suitable to apply supervised clustering.

Department :.....Mathematics..... Student's Signature……
Field of Study:.....Computational Science..... Advisor's Signature……
Academic Year : .....2008.....

ACKNOWLEDGMENT

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank Development and Promotion of Science and technology Talents project (DPST) for financial support during my seven years in the university, and every teacher and professor in the department of mathematics, Chulalongkorn University and my previous academies for their contribution of knowledge and guidance. I am deeply and specially indebted to my advisor, Assistant Professor Dr. Krung Sinapiromsaran who gives me all the support, suggestions, supervision, challenges and encouragement helping me through all the time of research and writing of this thesis.

I am grateful to the thesis committee, Assistant Professor Jaruloj Chongstitvatana, Ph.D., Associate Professor Pornchai Satravaha, Ph.D. and Associate Professor Paisan Nakmahachalasint, Ph.D, for their invaluable criticism and advice.

I would like to thank all friends and colleagues for all support and relaxation. And I would give a big thank and love to my family; my mother, my father and my sister.

CONTENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION

1.1 Motivation and Problem Description

Current progress in digital data acquisition and storage technology has initiated the growth of huge data collections [1]. This emerges in all areas, from the regular activities (such as supermarket transaction data, credit card usage records [2], telephone call marketing, and government statistics) to the more exotic applications (such as images of astronomical bodies [3], molecular databases [4], and medical records). The interest on extracting useful information from these data is increasing. The discipline concerned with this task is known as *data mining*.

Data mining is the analysis of (often large) observational data sets to discover unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. The relationships and summaries derived through data mining progresses are often referred to as *models* or *patterns*. The examples include linear equations, rules, clusters, graphs, tree structures, and recurrent patterns in time series.

Data mining is often fit in the broader context of *knowledge discovery in databases,* or KDD [5]. The KDD process involves several stages: selecting the target data, preprocessing the data, transforming, performing data mining to extract patterns and relationships, and then interpreting and assessing the discovered structures. The most time-consuming stage in KDD is data pre-processing [6] or data preparation. Since real world data contain some difficulties such as missing values, errors and outliers, unequal range of data values, and the mixture of numerical and nominal values, the data analyst needs to find the suitable data preparation and help choosing an effective data mining model to apply.

The data normalization [7], as we discuss in this thesis, is a part of data preparation which is used to transform numerical data values into a user-defined range.(usually 0-1) [2] There are evidences that this method help improving the

accuracy and efficiency on mining algorithms such as the neural network, nearest neighbor and clustering algorithms. Such methods provide better results if data to be analyzed are scaled to some specific ranges. In the neural network for classification mining [8], normalizing the input values for each attribute measured in the training samples helps speeding up the learning phase. For distance-based methods, normalization helps prevent attributes with initially large ranges from outweighing attributes with initially smaller ranges. There are three well-known methods for data normalization include min-max normalization, z-score normalization, and normalization by decimal scaling.

The other scaling technique is called the user-defined scaling. Some datasets and mining algorithms will achieve a better accuracy and running time if the right user-defined scale is used. However, the effort and time to determine the appropriate scaling are tremendous even for a few attributes. This leads to our main motivation which is to find the alternative method to have machine automatically determine the suitable scale for a dataset with a specific mining algorithm.

After data preparation process is done, the effective data mining model is guided by various factors such as the objective of the problem, the characteristic of results. In data mining, there are many types of models and patterns offered such as linear equations, rules, clusters in the form of their representatives, graphs, tree structures, and recurrent patterns in time series which have different strategies, methods and techniques to achieve the desired and effective results. This requires machine learning knowledge [9] to develop algorithms which allow computers automatically derive these models and patterns.

Machine learning algorithms can be categorized in various forms based on desired outcomes. Common forms are supervised learning and unsupervised learning. Supervised learning is a machine learning technique for learning a model or pattern from training data. The training data consist of pairs of input vectors or data points and the desired output. The task is to predict the outcome for any valid data points after trained by training examples. On the other hand, unsupervised learning is a machine learning technique that determines or builds the representatives of input that

can be used to group a new input or represent the inputs to other machines. Unlike supervised learning, unsupervised learning does not require the pre-defined outputs of training set nor rewards from environments. Two very simple classic examples of unsupervised learning are clustering and dimensionality reduction [10].

Clustering [11] is one of unsupervised learning techniques which groups a set of physical or abstract data points into classes of similar points. The similarity used to dictate which cluster each data point belongs to can be any of defined distance metric or the density of data points in a defined area. It has been widely used in numerous applications, including pattern recognition [12], data analysis [13], image processing [14], and market research [15]. In general, clustering methods can be classified into the following categories [16]; partitioning methods, hierarchical methods, density-based methods, grid-based methods and model-based methods. However, the simplest and well-known clustering algorithm is *K*-means clustering algorithm.

K-means clustering is an example of a unique partitioning method, where each data point is assigned to only one cluster. The objective that this method tries to achieve is to minimize total intra-cluster variance in term of the squared error function

$$V = \sum_{i=1}^{k} \sum_{X \in S_i} (d(X, C_i))^2$$

where X is a data point, Si is the set of data points in the cluster i with Ci as its centroid. k is the number of cluster, d is the distance function and V is called the total intra-cluster variance.

In many applications including this thesis, the Lloyd's algorithm is adopted for *K*-means instead of minimizing total intra-cluster variance. The Lloyd's algorithm starts from randomly selecting k centroids or cluster centers. All data points are assigned to their closest centroids according to the distance metric. The new centroid for each cluster is calculated by the average values of data points in that cluster. Then all data points are re-assigned to their closest centroids again. The whole process is repeated until there is no change on assigning data points into clusters. The advantage of this method is its simplicity and effectiveness. However, the solution is guaranteed

only a local minimum and can be found if the mean of a cluster is defined. This may not be the case when data points have categorical attributes. The necessity for users to specify the number of clusters, in advance can also be considered as a disadvantage. Moreover, the *K*-means method is not suitable for discovering clusters with non-convex shapes [17]. It is sensitive to noise data points which can influence the average value despite of a small number of data points.

There are some factors that affect the results of clustering. The selection of initial data points plays a big role on the final solution as well as the defined distance metric [18]. The convergence of an algorithm can only be applied if the initial starting point is chosen wisely. Different distance metric also causes the different results for selecting centroids and clustering. To create a new and different distance metric, researcher adapted by adding scaling or weight vector to modify values in each attribute.

Even though clustering is considered as an unsupervised learning method, there are many researchers that tried to combine the concept of supervised learning into clustering such as constraint clustering [19] which is clustering with the conditions, semi-supervised clustering [20] which uses initial labeled data for seeding and weighting the clustering of unlabeled data and supervised clustering. Supervised clustering is a combined concept of supervised learning and clustering. The idea is to identify factors that affect clustering and to find the right configuration for clustering based on the training set which contains input vectors and desired output set. Al-Harbi and Rayward-Smith [21] suggested the multiplication of a weight vector, we called scaling vector, to the distance metrics and optimize the matrix as multivariate optimization looking for the optimal solution that gives the best clustering according to the given training set. To evaluate the goodness in this supervised clustering, the information about desired outputs or target classes is included and used as the measure.

There are several benefits of supervised clustering reported by Eick and Zeidat [22]. First, supervised clustering can be applied to create summary and background knowledge. The settings of clustering on each data can be presented and interpreted as

the information. Next benefit is to be used as the automatically buildimg tool which modifies the data value in data-preprocessing process. Then, the accuracy of the modified dataset will be improved for the distance-based classifiers such as Nearest Neighbor. Moreover, the supervised clustering might be used to directly enhance the performance of simple classifiers such as Naïve Bayes.

The knowledge of optimization techniques is required in order to find the optimal scaling vector. Even though Al-Harbi and Rayward-Smith introduced simulated annealing for solving this optimization problem, we use the unconstrained optimization techniques [23] due to the absence of any restrictions. With each scaling vector, the goodness of each clustering is measured by its misclassification error. Unconstrained optimization proceeds by optimizing along the direction that minimizes misclassification error.

There are many different kinds of unconstrained optimization [23]. Since the misclassification error of clustering cannot be post as the continuous function, this function to optimize uses a derivative-free method. This excludes using a very popular method such as Newton method while the techniques using a direct search are more suitable for this problem. Direct search techniques look for the optimal solution along the given direction via some simple line search methods. For multivariate problems, the simplest direct search is called Cyclic coordinate.

Cyclic coordinate is a basic technique for multivariate unconstrained optimization. The aim is to optimize the value of multivariate function via the standard coordinate direction. This transform multivariate optimization problem into a series of univariate optimization problems. Another multivariate unconstrained optimization is Hooke and Jeeves method which improves cyclic coordinate by adding a search direction in order to avoid a premature termination that might occurs on coordinate directions.

These techniques use the concept of transforming a multivariate problem to a series of univariate problems which require a line search method. The widely known line search method is golden section line search which use the golden ratio (around

0.618) to reduce the interval of uncertainty (interval that contains optimal value) of the search.

So, the contribution of this thesis is to introduce automatic attribute-scaling algorithm based on the knowledge of supervised clustering. Supervised clustering is used to automatically find a scaling vector which is trained and searched for the optimal solution via techniques of unconstrained optimization; cyclic coordinate and Hooke and Jeeves method based on the golden section line search.

This thesis is divided into 5 chapters. The next chapter presents background knowledge and related researches required for this thesis. Then the details and process of the works are discussed in chapter 3. The experiments and results are described in chapter 4. And finally, chapter 5 draws conclusion and suggests possible future work and improvement.

CHAPTER II
BACKGROUND KNOWLEDGE

2.1 Data Mining Concept

Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, and potentially useful patterns in data. In this context, data mining is a step in the KDD process that centers on the automated discovery of new facts and relationships in data [1].

Data are usually collected from various sources and stored in a data warehouse and may include the different data scheme. A good data preparation is required to shape up the selected data for the effective model used for data mining. Data preparation is considered as a time consuming step of the KDD process, requiring most time and resources to be completed. There are many types of data preparation processes which depend on data and selected data mining models.

2.1.1 Data Normalization

For a raw data in a practical situation, data integration and transformation are required to help the mining process on dealing with the data values. Data transformation such as normalization can improve an accuracy and efficiency of mining algorithms involving neural networks, nearest neighbor, and clustering classifiers. Such methods will provide a better result if the data to be analyzed has been normalized, see [7]. For empirical study, using the neural network back propagation algorithm for classification mining [24] with normalized attributes will help speed up the learning phase. For distanced-based methods, normalization helps prevent attributes with initially large ranges from outweighing attributes with other smaller ranges [7]. There are many methods for data normalization including min-max normalization, z-score normalization, and normalization by decimal scaling.

Min-max normalization performs a linear transformation on the original data. Suppose that a and b are the minimum and the maximum values for attribute A. Min-max normalization maps a value v of A to v' in the range $[c, d]$ by computing:

$$v' = \left(\frac{v - a}{b - a} \right) \cdot (d - c) + c$$

In z-score normalization, the values of an attribute *A* are normalized based on the mean and standard deviation of *A*. A value *v* of *A* is normalized to *v*′ by computing:

$$v' = \frac{v - \overline{x}_A}{\sigma_A}$$

where $\overline{x}_A$ and σ_A are the mean and the standard deviation of attribute *A*, respectively. This method of normalization is useful when the actual minimum and maximum of attribute *A* are unknown.

Normalization by a decimal scaling deals with changing the decimal place of values of attribute *A*. The number of selected decimal places depends on the maximum absolute value of *A*. A value *v* of *A* is normalized to *v*′ by computing:

$$v' = \frac{v}{10^j}$$

where *j* is the smallest integer such that Max(| v' |) < 1

Normalization changes the original data and it is necessary to save the normalization parameters (the mean and the standard deviation if using the z-score normalization, the minimum and the maximum values if using the min-max normalization and the power index of 10 if using the decimal scaling normalization) so that future data can be normalized in the same manner.

2.1.2 *K*-means Clustering

Clustering problems arise in many different applications, such as data mining and knowledge discovery, data compression, vector quantization, pattern recognition and pattern classification. The notion of a good cluster depends on the application and there are many methods for finding clusters subject to various criteria, both ad hoc

and systematic. These include approaches based on splitting and merging such as ISODATA[10], randomized approaches such as CLARA [25], CLARANS [26], methods based on neural networks, and methods designed to scale to large databases, including DBSCAN, BIRCH, and ScaleKM [27, 28].

Among clustering formulations that are based on minimizing various objective functions, the most widely used and studied is ***K*-means clustering**. Invented in 1956, the *K*-means algorithm starts by given a set of data points in real *n*-dimensional space, $\mathbb{R}^n$, and an integer *k*. The problem is to determine a set of *k* points in $\mathbb{R}^n$ called centers or centroids, that minimize the mean squared distance from each data point to its nearest center. This measure is often called the squared-error distortion

$$V = \sum_{i=1}^{k} \sum_{X \in S_i} (d(X, C_i))^2$$

where X is a data point, S_i is the set of data points in the cluster i with C_i as its centroid. k is the number of cluster, d is the distance function and V is called the total intra-cluster variance.

This type of clustering falls into the general category of variance based clustering. Clustering based on *K*-means is closely related to a number of other clustering and location problems. These include the Euclidean *K*-medians, the multi-source Weber problem [29], in which the objective is to minimize the sum of distances to the nearest center and the geometric *K*-center problem in which the objective is to minimize the maximum distance from every point to its closest center.

There are no efficient solutions known to these problems and some formulations are NP-hard [30]. An asymptotically efficient approximation for the *K*-means clustering problem has been presented by Matousek [31], but the large constant factors found in his work makes his method to be impractical in the real situation.

One of the most popular heuristics for solving the *K*-means problem is based on a simple iterative scheme for finding a locally minimal solution. This algorithm is

often called the *K*-means algorithm. There are a number of variants to this algorithm. One of the popular algorithms is a generalized Lloyd's algorithm [32].

Lloyd's algorithm is based on the simple observation that the optimal placement of a center is at the centroid of their associated clusters. Given any set of k centers of Z, for each center $z \in Z$, let $N(z)$ denote its neighborhood, that is, the set of data points for which z is the nearest neighbor. Each stage of Lloyd's algorithm moves every center point z to the centroid of $N(z)$ and then updates $N(z)$ by re-computing the distance from each point to its nearest center. These steps are repeated until a convergent condition is met which normally is to measure unchanges of assigning data points into groups. Here is the step in Lloyd's algorithm.

Initialization Step: Select the k representatives as the starting points. Calculate the distance between each data points to all representatives. Assign data points to the clusters which the distance between itself and the group representative is the least.

Main Step:

1. Calculate the mean value of the members in the same clusters. Use this value as the new representative of each cluster.
2. Calculate the distance from all data points to these new representatives.
3. Check every data points whether it belongs to the same representative. If yes, done. Otherwise change its group to the new one. Back to step 1.

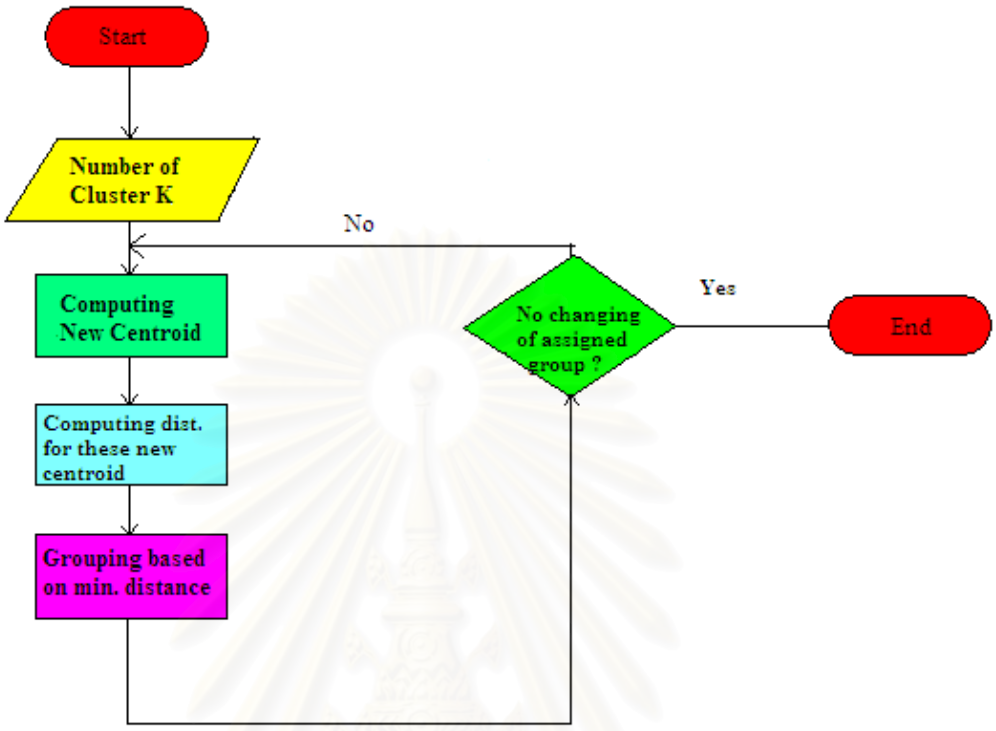

Fig.2.1 Flow diagram of the *K*-means clustering by Lloyd's algorithm

For data points in a general position (in particular, if no data point is equidistant from two centers), the algorithm eventually converges to a point that is a local minimum for the distortion. However, the result is not necessarily a global minimum. Bradley et al. [18] have shown how to scale *K*-means clustering to very large data sets through sampling and pruning. Note that Lloyd's algorithm does not specify the initial placement of centers.

Because of its simplicity and flexibility, Lloyd's algorithm is very popular in statistical analysis. However, a straightforward implementation of Lloyd's algorithm can be quite slow due to the cost of computing nearest neighbors.

In summary, *K*-means algorithm has the following important properties.

a) It works only on numerical values;

b) It usually uses the Euclidean metric and hence, the centroid of each cluster is the mean of the points in that cluster;

c) It is efficient in processing large data sets. The computational complexity of the algorithm is $O(nkt)$, where n is the total number of data points, k is the number of clusters and t is the number of iterations. In clustering large data sets the *K*-means algorithm is much faster than hierarchical clustering algorithms, whose computational complexity is generally $O(n^2)$;
d) It terminates at a local minimum.

In this thesis, we are interested in supervised *K*-means clustering, which uses side information of datasets about the output field or the pre-defined target class to help algorithm groups similar data points in the same cluster. The aim is to generate a cluster that has a strong tendency to partition data points into the same class. Any new data point assigned to a cluster is assumed to have the same class as the majority of data points in that cluster. The target class itself is used to aid and bias the constructing of a suitable metric defined on other attributes.

The idea of using supervised learning together with clustering algorithms has appeared quite recently. In [33], for example, the authors explored the use of a small amount of labeled data to aid and bias the clustering of unlabelled data. This paper also introduced a semi-supervised variant of *k*-means that uses initial labeled data for seeding. These seeds were kept unchanged throughout the algorithm. The authors claimed to improve the objective function (i.e maximizing the within-cluster similarity) that was used in the *K*-means algorithm. Cohn et al. [34] allowed the user to iteratively provide feedback (which may also be called supervision) to a clustering algorithm. The feedback was incorporated in the form of constraints which the clustering algorithm attempted to satisfy in future iterations. These constraints guided the clustering to become more relevant and useful. This algorithm attempted to give the user a way to interact with the data, however this required much human effort and may be impractical with high-dimensional data sets. Similarly, Wagstaff and Rogers [19] developed another variant of the *k*-means algorithm which incorporated background knowledge in the form of instance-level constraints. These instance-level constraints presented a priori knowledge about which data points should be grouped together. The variant algorithm added an if-statement to the updating cluster assignments. This conditional statement ensured that none of the specific constraints

were violated when the *K*-means algorithm attempted to assign each data point to its closest cluster. However, the major drawback of this algorithm was that the user needed to construct these constraints manually. Demiriz and Bennett [20] proposed a genetic cluster algorithm that used labeled data in addition to unlabelled data in the training data set, and the aim of using labeled data was to cluster the unlabelled data. This algorithm synergized the benefits of supervised and unsupervised learning methods. The proposed algorithm used varying values of *k* (7, 11 and 15, for small, medium and large data sets respectively). A similar technique had been proposed by Jourdan [35], though in this case the authors extended the algorithm in order to deal with categorical data. Al-Harbi and Rayward-Smith [21] introduced another concept of the supervised *K*-means algorithm which had slightly different techniques from the above algorithms. They adapted the traditional *k*-means algorithm to be used as a classification technique in order to predict the class label of unlabelled data. The Euclidean metric and the objective function of the traditional *K*-means algorithm were modified by the multiplication of a weight metric. In order to achieve this, all data in the training data set must be labeled and the value of *k* was predetermined. Moreover, they introduced the measure which is also used on this thesis and their searching algorithm was based on simulated annealing.

2.1.3 Distance function

The performance of distanced-based clustering, especially *K*-means clustering depend critically on a metric or distance function over the input space. Metrics should reflect reasonably well the important relationships between each data points and attributes. The distance is a numerical description of how far between data points. In a mathematical term, a **metric** or **distance function** is a function which defines a distance between elements of a set.

A **metric** *D* on a set *X* is a function mapping *X* into the set of real values. For all *x*, *y*, *z* in *X*, this function is required to satisfy the following conditions:

1. $D(x, y) \geq 0$

2. $D(x,y)=0 \leftrightarrow x=y$

3. $D(x,y)=D(y,x)$

4. $D(x,z) \le D(x,y)+D(y,z)$ "Triangle Inequality"

2.1.3.1 Minkowski Distance

In the Euclidean space ^{n}, the distance between two points is usually given by the Euclidean distance (2-norm distance).

For a point X (x_1, x_2, ...,x_n) and a point Y (y_1, y_2, ...,y_n), the **Minkowski distance *D*** of order p (***p*-norm distance**) is defined as:

$$D_p(X,Y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$$

and other norms are

1-norm distance (taxicab norm or Manhattan distance)

$$D_1(X,Y)=\sum_{i=1}^{n}|x_i-y_i|$$

2-norm distance (Euclidean distance)

$$D_2(X,Y)=\left(\sum_{i=1}^{n}(x_i-y_i)^2\right)^{\frac{1}{2}}$$

∞-norm distance (*Chebyshev distance*)

$$D_\infty(X,Y)=\lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}=\max_{i=1,..n}\left(|x_i-y_i|\right)$$

2.1.4 Attribute-scaling space

From Euclidean distance function $D(X,Y)=(\sum_{i=1}^{n}(x_i-y_i)^2)^{\frac{1}{2}}$, this thesis will modify the function by multiplying a coefficient to each dimension. We define the collection of these multipliers as the attribute-scaling vector. Note that, the original Euclidean distance function for *n*-dimension space has an attribute-scaling vector as the vector that contains every element equal to one.

The value of each element in this vector can be interpreted as the influence of each attribute to the performance of the clustering. The attribute which has the largest value relatively to the others is the most important attribute for clustering process. The vector which is the multiple of another vector gives the same results after the clustering is applied.

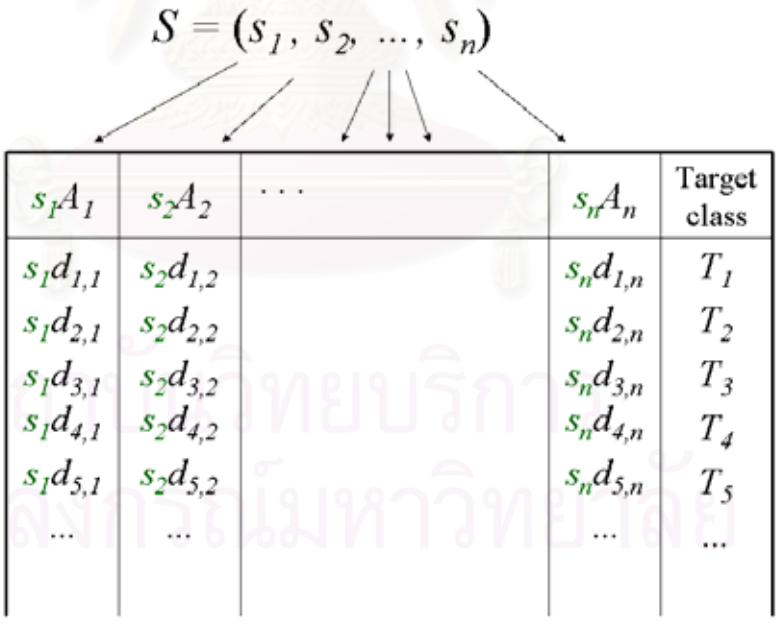

Fig.2.2 Illustration of data transformation applying scaling vectors to data for clustering, where $d_{i,j}$ is the element of i^{th} data point in j^{th} attribute.

Al-Harbi and Rayward-Smith used the similar notation with a different name called ‘weight metric’. They also explained that natural clustering regards all fields to

be equally important when determining the best partitions for a data set. However, when there were a priori hypotheses about the output of a clustering algorithm, then this natural assumption was violated; the clusters were needed based upon these hypotheses.

Our aim for scaling vector is to help the algorithm to partition the data into its different class labels based on assumption that all data points within a close proximity will be in the same class. Consequently, when measuring the distances between attributes, it may be necessary to assign greater significance to one attribute over another. This can be achieved by assigning greater weight to those attribute values which have a more significant relationship with a class label.

2.1.5 Misclassification Error

As previously mentioned, this thesis focuses on supervising *K*-means algorithm by adjusting a scaling vector to find the proper metric and eventually the proper clustering for data. To determine a suitable vector for *K*-means clustering, it requires a measure to evaluate and judge its result. In this work, we use the standard measure for classification techniques, called the misclassification error.

The misclassification error is the error introduced under the assumption that after perfect clustering, the data points which are contained in the same cluster should have the same target class. However in practice, each cluster will contain some data points from other classes. So we define these misclassified data points as the error. We denote the error in term of function as

$$Error_S = \frac{\sum_{i=1}^{K} \left| \{ d \in C_i \,|\, t_d \neq TC_i \} \right|}{m}$$

where the data point d with the target class t_d is on the cluster C_i. For each cluster i, it contains TC_i as the target class of each cluster which is the most common class of data points in their respective cluster. This dataset contains m data points. We also define the cardinality$|A|$ as the number of member in a set A.

We illustrate the concept as in the figure 2.3 :

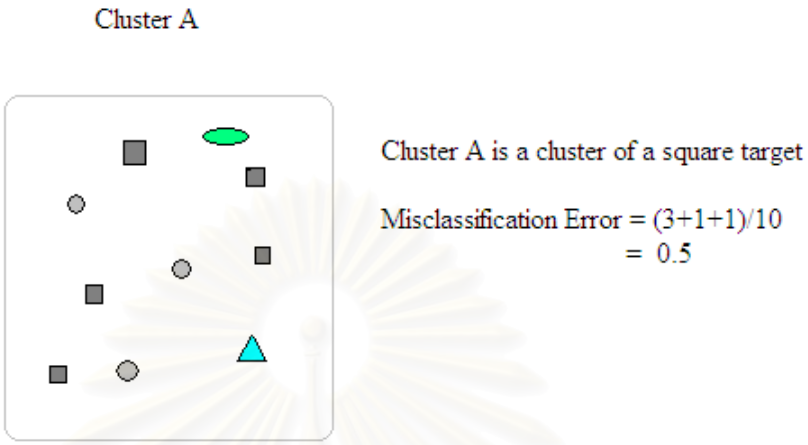

Fig.2.3 The calculation of misclassification error

In this thesis we will supervises the *K*-means algorithm by adjusting the attribute-scaling vectors. With the dataset of *n*-dimension input space, the most trivial algortihm for finding a solution is to use a brute-force search which enumerates every single possible candidate for solution looking for the minimum misclassification error. It guarantees a global solution but consumes enormous time and efforts. Another trivial one is a random search which spends less resource but give unpredictable performance. This work applies a concept of optimization to find an optimal scaling vector as the heuristic local search.

2.2 Optimization Concept

Optimization is a study of problems in which one seeks the optimal solution of the problem [18]. It is one of major fields in applied mathematics containing many subfields divided by characteristics of problems. The important characteristics which require different approaches on finding solution include types of function which is needed to optimize, types of solutions and their dimensions, the number and

description of conditions given in the problems. For choosing a suitable method for each problem, we have to identify these characteristics.

Unconstrained optimization is the optimization that deals with the problem of minimizing or maximizing a function in the absence of any restrictions. With the techniques of unconstrained optimization, only some side restrictions are still required. Most methods proceed by finding direction and minimizing along this direction by the procedures called the line search.

In this thesis, we focus on two elements of unconstrained optimizations. Since the objective of this work is to find the attribute-scaling vector that provides the minimum misclassification error, the problems will be considered as the multivariate minimization problems. These misclassification errors required the optimization methods that do not involve derivatives. These methods are called the direct method. For this work, we will adapt one of the simple direct methods, cyclic coordinate. It is the method which minimizes the function value along each of the coordinate directions for finding optimal solutions. In order to minimize on each direction, golden section line search will be applied for this propose. In next two sections, we will go through the details of these two techniques.

2.2.1 The Golden section line search

One-dimensional search is the key of many algorithms for solving non-linear programming problems [23]. To minimize the function with one variable, one of the popular approaches is to set the derivative of the function equal to 0 or to determine the critical point. Note that it becomes impossible if the function is not differentiable. Some which are differentiable may become unstable when searching near the critical point. Furthermore, this provides only the candidates that could be just the local optimal or the saddle points. So for these reasons, this approach will be avoided and the numerical techniques will be used instead.

In order to minimize a univariate function over a closed and bounded interval without using derivatives, the concept of sequential line search will be used to find the interval that the local minimum point is contained. This kind of method uses the values of the function at the previous iterations to determine the succeeding points

and continuously reduce the length of the interval which contains local optimal point until we reach the acceptable length.

The interval of uncertainty

Consider the line search problem to minimize $\theta(\lambda)$ subject to $a \leq \lambda \leq b$ [23]. Since the exact location of the minimum is unknown, this interval is called the interval of uncertainty. During the search procedure if we can exclude portions that do not contain minimum, then the interval of uncertainty is reduced.

The following theorem shows that if the function θ is strictly quasi-convex [18], then the interval of uncertainty can be reduced by evaluating θ at two points within the interval.

Theorem 1 Let $\theta : \rightarrow$ be strictly quasi-convex over the interval $[a, b]$ be such that $\lambda < \mu$. If $\theta(\lambda) \geq \theta(\mu)$, then $\theta(z) \geq \theta(\mu)$ for all $z \in [a, \lambda)$ If $\theta(\lambda) < \theta(\mu)$, then $\theta(z) \geq \theta(\lambda)$ for all $z \in (\mu, b]$.

From the above theorem, under strict quasi-convexity if $\theta(\lambda) \geq \theta(\mu)$, the new interval of uncertainty is $[\lambda, b]$. On the other hand, if $\theta(\lambda) < \theta(\mu)$, then the new interval of uncertainty is $[a, \mu]$.

Now we present the golden section method which is one of the procedures for minimizing a strictly quasi-convex function over a closed and bounded interval by iteratively reducing the interval of uncertainty. The golden section line search is classified as the sequential search; the search that utilizes the information generated at the previous iterations. Normally the interval of uncertainty of the line search has the reduction ratio that can be calculated by the ratio of the length of the interval of uncertainty after v observations and the length before taking the observations. The golden section gives the effective number of reduction ratio at $(0.618)^{v-1}$.

At an iteration k of the golden section method, let the interval of uncertainty be $[a^k, b^k]$. By theorem 1, the new interval of uncertainty $[a^{k+1}, b^{k+1}]$ is given by $[\lambda^k, b^k]$

if $f(\lambda^k) \geq f(\mu^k)$ and by $[a^k, \mu^k]$ if $f(\lambda^k) < f(\mu^k)$. The points λ^k and μ^k are selected such that

1. The length of the new interval of uncertainty $b^{k+1} - a^{k+1}$ does not depend upon the outcome of the k^{th} iteration. Therefore, we want $\mu^k - a^k = b^k - \lambda^k$. Thus, if λ^k is of the form

$$\lambda^k = a^k + (1 - \alpha)(b^k - a^k) \tag{1}$$

where $\alpha \in (0, 1)$, then μ^k must be of the form

$$\mu^k = a^k + \alpha(b^k - a^k) \tag{2}$$

so that

$$(b^{k+1} - a^{k+1}) = \alpha(b^k - a^k)$$

2. As λ^{k+1} and μ^{k+1} are selected for the purpose of a new iteration, either λ^{k+1} coincides with μ^k or μ^{k+1} coincides with λ^k. If this can be realized, then during iteration $k + 1$, only one extra observation is needed. To illustration, consider the figure 2.4 and the following two cases.

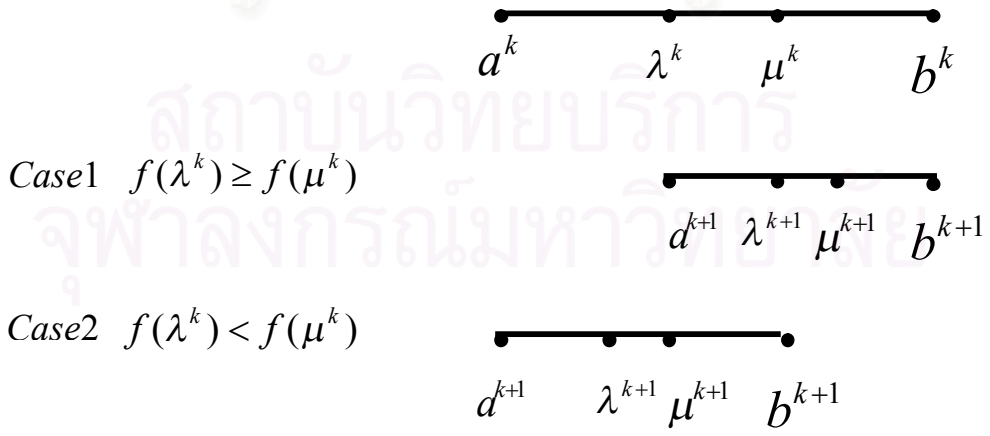

Fig.2.4 The reduction of interval in golden section line search

Case 1: $f(\lambda^k) \geq f(\mu^k)$

In this case, $a^{k+1} = \lambda^k$ and $b^{k+1} = b^k$. To satisfy $\lambda^{k+1} = \mu^k$, and applying (1) with k replaced by $k + 1$, we get

$$\mu^k = \lambda^{k+1} = a^{k+1} + (1-\alpha)(b^{k+1} - a^{k+1}) = \lambda^k + (1-\alpha)(b^k - \lambda^k)$$

Substituting the expressions of λ^k and μ^k from (1) and (2) into the above equation, we get $\alpha^2 + \alpha - 1 = 0$.

Case 2: $f(\lambda^k) < f(\mu^k)$.

In this case, $a^{k+1} = a^k$ and $b^{k+1} = \mu^k$. To satisfy $\mu^{k+1} = \lambda^k$, and applying (2) with k replaced by $k + 1$, we get

$$\lambda^k = \mu^{k+1} = a^{k+1} + \alpha(b^{k+1} - a^{k+1}) = a^k + \alpha(\mu^k - a^k)$$

The above also gives $\alpha^2 + \alpha - 1 = 0$. The positive root of this equation is approximately equal to 0.618 which is in the interval (0, 1). To summarize, if at iteration k, μ^k and λ^k are chosen according to (1) and (2), where $\alpha = 0.618$, then the interval of uncertainty is reduced by the factor of 0.618. At the first iteration, it requires two observations at μ^1 and λ^1 but at other subsequent iterations, requires only one function evaluation.

Summary of the golden section method

This is a summary of the golden section method for minimizing a strictly quasi-convex function over the interval $[a^1, b^1]$.

Initialization Step

Choose an acceptable final length of uncertainty $l > 0$. Let $[a^1, b^1]$ be the initial interval of uncertainty, let $\lambda^1 = a^1 + (1 - \alpha)(b^1 - a^1)$ and $\mu^1 = a^1 + \alpha(b^1 - a^1)$, where $\alpha = 0.618$. Evaluate $f(\lambda^1)$ and $f(\mu^1)$, let $k = 1$, and go to the main step.

Main Step

1. If $b^k - a^k < l$, stop; the optimal solution lies in the interval $[a^k, b^k]$. Otherwise, if $f(\lambda^k) \geq f(\mu^k)$, go to step 2; and if $f(\lambda^k) < f(\mu^k)$ go to step 3.
2. Let $a^{k+1} = \lambda^k$ and $b^{k+1} = b^k$. Furthermore, let $\lambda^{k+1} = \mu^k$, and let $\mu^{k+1} = a^{k+1} + \alpha(b^{k+1} - a^{k+1})$. Evaluate $f(\mu^{k+1})$ and go to step 4.
3. Let $a^{k+1} = a^k$ and $b^{k+1} = \mu^k$. Furthermore, let $\mu^{k+1} = \lambda^k$ and let $\lambda^{k+1} = a^{k+1} + (1 - \alpha)(b^{k+1} - a^{k+1})$. Evaluate $f(\lambda^{k+1})$ and go to step 4.
4. Replace k by $k + 1$ and go to step 1.

2.2.2 The cyclic coordinate method

When we consider the problem of minimizing a function f of multivariable without using derivatives [23], the straight forward concept to evaluate is given a vector $\boldsymbol{X}$ and determines a suitable direction $\boldsymbol{D}$. And then f is minimized from $\boldsymbol{X}$ along the direction $\boldsymbol{D}$ by a line search method. For the purpose of simplicity, we will assume the existence of a minimum point ρ.

Cyclic coordinate method uses the coordinate axes $\boldsymbol{D_j} = (d_1, d_2, \ldots, d_n)$ which d_j equal to 1 at index j, while the rest is zero. More specifically, the method searches along one direction (one attribute) while the other ones are fixed. The method is illustrated schematically in figure 2.5.

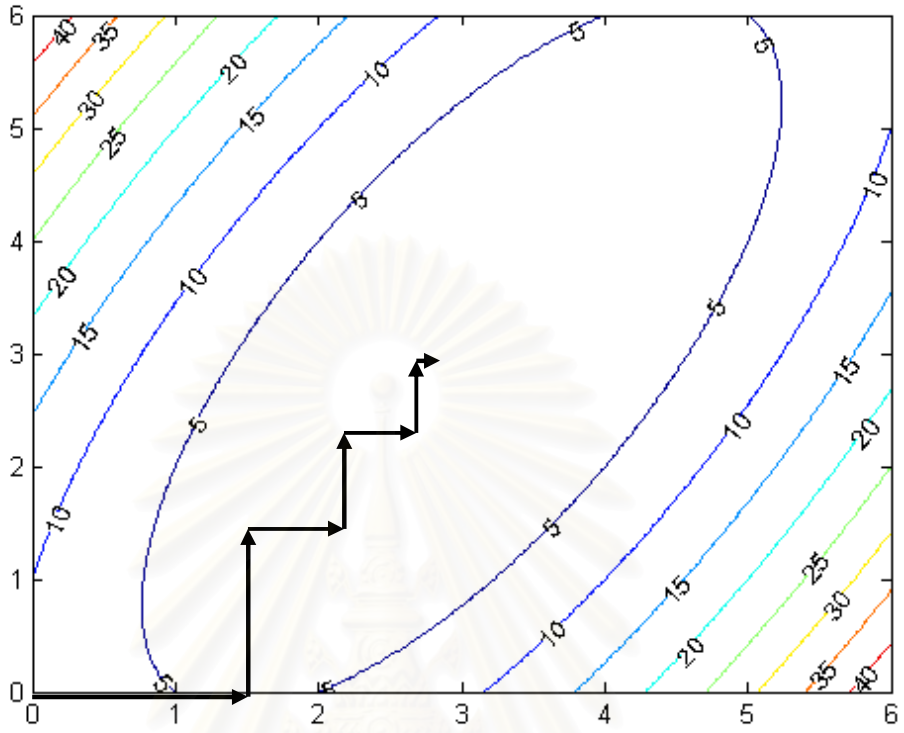

Fig.2.5 The search pattern of the cyclic coordinate method.

Summary of the cyclic coordinate method

We summarize below the cyclic coordinate method for minimizing a function of several variables without using any derivative information. In order to terminate the process, we use $\| X_{k+1} - X_k \| < \varepsilon$ but any other criteria can also be used.

Initialization step

Choose a scalar $\varepsilon > 0$ as a termination criterion for the algorithm. Let D_1, ..., D_n be the coordinate directions. Choose an initial point X_1, let $Y_1 = X_1$, let $k = j = 1$ and go to the main step.

Main step

1. Let ρ_j^* be an optimal solution to the problem to minimize $f(Y_j + \rho_j D_j)$ subject to ρ_j in the closed interval and let $Y_{j+1} = Y_j + \rho_j^* D_j$. If $j < n$, replace j by $j + 1$, and repeat step 1. Otherwise, if $j = n$, go to step 2.

2. Let $X_{k+1} = Y_{n+1}$. If $\| X_{k+1} - X_k \| < \varepsilon$ then stop. Otherwise, let $Y_1 = X_{k+1}$, let $j = 1$, replace k with $k + 1$, and repeat step 1.

Sometimes cyclic coordinate method can stall at a non-optimal point. Searching along any coordinate axes may leads to no improvement of the function and results in premature termination. The reason behind this is the presence of a valley caused by the non-differentiability of f. However, it can possibly overcome by searching along the direction $X_{k+1} - X_k$.

A search along the direction $X_{k+1} - X_k$ is frequently used in applying the cyclic coordinate. This modification frequently accelerates convergence, particularly when the sequence of points generated zigzags along a valley. Such a step is usually referred to as an acceleration step, or a pattern search step.

2.2.3 The method of Hooke and Jeeves

The method of Hooke and Jeeves [23] performs two types of search – exploratory search and pattern search. Given X_1, an exploratory search along the coordinate directions produces the point X_2. Now a pattern search along the direction $X_2 - X_1$ leads to the point Y, Another exploratory search starting from Y give the point X_3. The next pattern search is along the direction $X_3 - X_2$, yielding the new point. The process is then repeated, see figure 2.6.

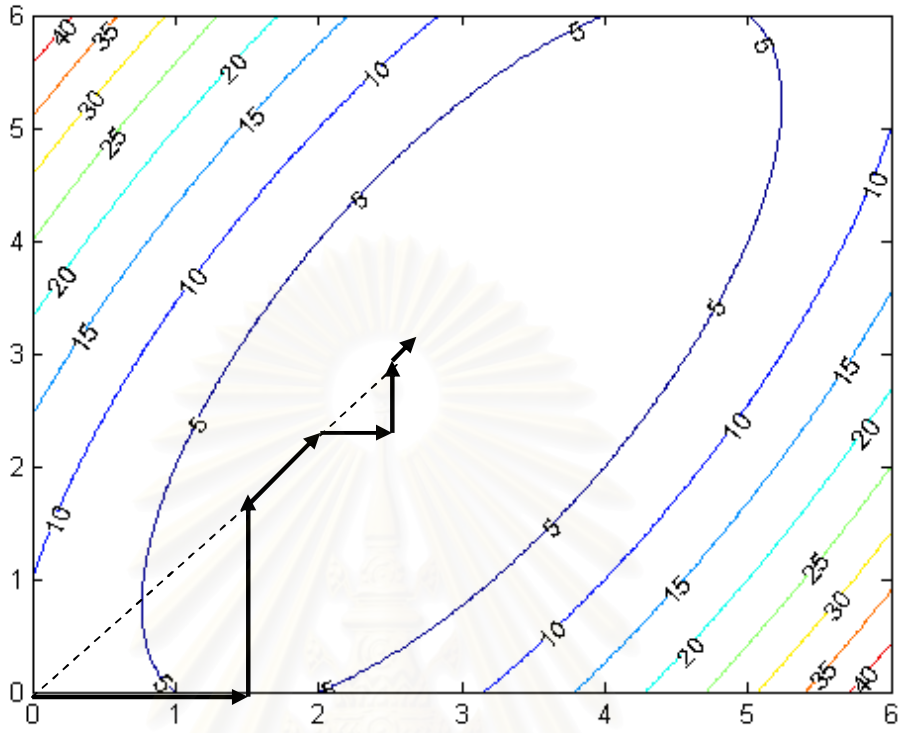

Fig.2.6 The search pattern of Hooke and Jeeves method.

Summary of the method of Hooke and Jeeves

Initialization step

Choose a scalar $\varepsilon > 0$ as a termination criterion for the algorithm. Let D_1, ..., D_n be the coordinate directions. Choose an initial point X_1, let $Y_1 = X_1$, let $k = j = 1$ and go to the main step.

Main step

1. Let ρ_j^* be an optimal solution to the problem to minimize $f(Y_j + \rho_j D_j)$ subject to ρ_j in the closed interval and let $Y_{j+1} = Y_j + \rho_j^* D_j$. If $j < n$, replace j by $j + 1$, and repeat step 1. Otherwise, if $j = n$, let $X_{k+1} = Y_{n+1}$. If $\| X_{k+1} - X_k \| < \varepsilon$, then stop; Otherwise, go to step 2.

2. Let $D = X_{k+1} - X_k$ and let σ^* be an optimal solution to the problem to minimize $f(Y_{n+1} + \sigma D)$ subject to σ in the closed interval. Let $Y_1 = X_{k+1} + \sigma^* D$, let $j = 1$, replace k with $k + 1$, and repeat step 1.

Combining these two concepts, we construct the supervised *K*-means algorithm with the attribute-scaling vector and minimize the misclassification error from the *K*-means clustering with the cyclic coordinate method and Hooke and Jeeves method to find the optimal vector that give the minimum error. The detail of the work is described in next chapter.

CHAPTER III
K-MEANS SCALING ALGORITHM CONTROLLED BY MISCLASSIFICATION ERROR OF ALL CLUSTERS

In this chapter, we introduce the detail of the thesis work. This work is the combination of data mining and optimization. It is divided in two parts; applying attribute-scaling vector to *K*-means algorithm and the optimization of misclassification error. The fitness of an attribute-scaling vector is obtained by first running the *K*-means algorithm with the corresponding weighted Euclidean metric. The algorithm iteratively creates clusters as it groups together data points which have the same class label. After clustering, each member of a cluster is compared with its class label or target class.

3.1 Applying attribute-scaling vector to *K*-means algorithm

For *K*-means algorithm, we need to define the distance function used for the process. This work adjusts the standard Euclidean distance using a weight vector. Some attributes that we weight more than the others easily affect the closeness of each component and lead to the different clustering results. With this effect, we determine the weight on each element of the distance function that gives the best classification result. This weight is considered as the multidimensional vector lying on the attribute-scaling space.

The standard Euclidean distance from the previous chapter is represented as the vector of one, while the standard data normalizations can also be written as the vectors in this attribute-scaling space as we show on the table 3.1

The type of scaling or distance function	The formula	The attribute-scaling vector
The Euclidean distance function	$D(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$	$(1, 1, \ldots, 1)$
Min-max normalization	$v_i'=\left(\frac{v_i-a_i}{b_i-a_i}\right)\cdot(d_i-c_i)+c_i$	$\left(\frac{d_1-c_1}{b_1-a_1},\frac{d_2-c_2}{b_2-a_2},\ldots,\frac{d_n-c_n}{b_n-a_n}\right)$
Z-score normalization	$v_i'=\frac{v_i-\bar{x}_i}{\sigma_i}$	$\left(\frac{1}{\sigma_1},\frac{1}{\sigma_2},\ldots,\frac{1}{\sigma_n}\right)$

Table 3.1 Distance function and its attribute-scaling vector formula

These vectors belong to the attribute-scaling space. We aim to find the vector that gives the best results. The results we consider as the best will be determined by the misclassification error.

In Euclidean space $\mathbb{R}^n$, We use $D = (d_1, d_2, \ldots, d_n)$ to represent a data point in a dataset with m data points and define the attribute-scaling vector $S = (s_1, s_2, \ldots, s_n)$. For each D, it has a target class t_D for a data point D, and there is T as a set of target classes for this dataset. For any the attribute-scaling vector S, define the modified metric as

$$\delta_S(p,q)=\sqrt{\sum_{i=1}^{n}s_i^{\,2}(p_i-q_i)^2}$$

We called δ_S an attribute-scaling metric.

From definitions, we apply K-means algorithm based on the attribute-scaling metric. After each cluster has been assign the class, we measure the misclassification error by

$$Error_S=\frac{\sum_{i=1}^{K}\left|\left\{d\in C_i \,|\, t_d\neq TC_i\right\}\right|}{m}$$

where the data point d with the target class t_d is on the cluster C_i. For each cluster i, it contains TC_i as the target class of each cluster which is the most common class of data points in their respective cluster. This dataset contains m data points. We also define the cardinality$|A|$ as the number of members in a set A.

The error can equal to zero if every data points in the same cluster is in the same class, while the maximum error never reaches to one. For each attribute-scaling vector, this generates one value of misclassification error. In this thesis we will treat this as the function to minimize.

3.2 The optimization of attribute-scaling vector

To determine the vector S, we measure the misclassification error occurring from K-means algorithm as the objective function.

Since the attribute-scaling vector is multi-dimensional, the techniques of multivariate optimization are adopted to find the solution. As mentioned before, we treat the misclassification error from the scaled K-means algorithm as the function from the attribute-scaling space. So the derivative of this function becomes impossible to gather. These obstacles lead us to choose direct method for the unconstrained optimization. In this thesis, we show results from two techniques, the cyclic coordinate method and the Hooke and Jeeves method.

3.2.1 The cyclic coordinate method

The cyclic coordinate is the simplest direct method. It minimizes the function value along the standard coordinate directions for finding optimal solutions. To look for each coordinate direction, it will transform the multivariate optimization to univariate optimization which can be solved by the line search.

Generally, the effective of the line search is based on the reduction rate of its uncertainty interval. The higher the reduction rate is, the quicker the line search leads to the optimal solution. The golden section line search is one of the effective line searches due to its reasonable reduction ratio which is around 0.618^{v-1} where v is the number of function evaluation. Moreover, for each search iteration, it requires only

one function calculation. With both advantages, the golden section line search is chosen.

The controlled factors that we used are the number of clusters used for *K*-means, the searching ranges of each dimension and the terminal threshold of the optimization. In this work we varied the number of the clusters to investigate the effect of the number of the clusters with respect to the algorithm, while other configurations will be discussed later in chapter 4.

Algorithm

Input: Dataset, the number of clusters
Output: The local minimum scaling vector

Initial Step

1. Define function to minimize *f*, the value of *f* at the vector *S* or *f*(*S*) is the misclassification error from *K*-means clustering algorithm with Euclidean distance.

$$D_S(X,Y) = \sqrt{\sum_{i=1}^{n} s_i^2 (x_i - y_i)^2}$$

2. Start with initial S_0, defined by giving s_i equal to the reciprocal of the range of the i^{th} attribute for $i = 1, ..., n$ (the number of attributes)
3. Let $d_1, d_2, ...d_n$ be the coordinate directions, j be the index of direction starting from 1 and k be the index of iteration, then go to main step

Main Step

4. Let ρ_j^* be the optimal solution to minimize $f(S_{k-1} + \rho_j d_j)$ in the defined range (in this work, we use [0, 10]). To find the optimal solution for each j, we use golden section line section along the direction d_j until j equal to n. After that go to step 5.
5. For $j = 1, 2, \ldots, n$, choose the vector $S_{k-1} + \rho_j^* d_j$ that give the minimum $f(S_{k-1} + \rho_j d_j)$ as S_k .
6. If $f(S_k) > f(S_{k-1})$, then stop. Otherwise, let $j = 1$, update k and repeat step 4. Our solution will be the vector S_{k-1}.

3.2.2 The method of Hooke and Jeeves

The drawback of searching along only coordinate directions which occasionally occur when we deal with a non-differentiable function is that the search can be terminated prematurely due to a valley of the function. In order to overcome this difficulty, Hooke and Jeeves method suggested the addition of a pattern search $X_{k+1} - X_k$. With this modification, the method often accelerates its convergence.

In this thesis, we apply the method of Hooke and Jeeves hoping to improve the result we achieve from the cyclic coordinate method. With the assumption that our supervised *K*-means algorithm may have some valleys on the attribute spaces, this method will avoid these obstacles and find the better optimal solution.

For this method, the algorithm is slightly changed; another pattern search will be added while other codes will stay the same.

Algorithm

Input: Dataset, the number of clusters
Output: The local minimum scaling vector

Initial Step

1. Define function to minimize *f*, the value of *f* at the vector *S* or *f(S)* is the misclassification error from *K*-means clustering algorithm with distance measure

$$D_S(X,Y) = \sqrt{\sum_{i=1}^{n} s_i^2 (x_i - y_i)^2}$$

2. Start with initial S_0, defined by giving s_i equal to the reciprocal of the range of the i^{th} attribute for $i = 1, ..., n$ (the number of attributes)
3. Let $d_1, d_2, ...d_n$ be the coordinate directions, j be the index of direction starting from 1 and k be the index of iteration, then go to main step

Main Step

4. Let ρ_j^* be the optimal solution to minimize $f(S_{k-1}+\rho_j d_j)$ in the defined range (in this work, we use [0, 10]). To find the optimal solution for each j, we use golden section line section along the direction d_j until j equal to n. After that, go to step 5.
5. For $j = 1, 2,\ldots, n$ choose the vector $S_{k-1} + \rho_j^* d_j$ that give the minimum $f(S_{k-1} + \rho_j d_j)$ as S^k .
6. Let d_s be the direction of the pattern search $S_k - S_{k-1}$, minimize $f(S_k + \rho d_s)$. If $f(S_k) > f(S_{k-1})$, then stop. Otherwise, let $j = 1$, update k and repeat step 4. Our solution will be the vector S_{k-1}.

We perform the experiments to test algorithms with the real world datasets. In the next chapter, we will introduce the datasets using for our experiments, then analyze the result from experiments that we have found.

CHAPTER IV
EXPERIMENTS AND RESULT

We use MATLAB 7.0 as a tool to implement thanks to the existing *K*-means algorithm on its statistic toolbox. It becomes easier to apply the attribute-scaling vector using MATLAB 7.0. The experiments are performed through Pentium core2duo 2.13 GHz processor with 1 GB RAM memory.

Databases we use for this thesis are gathering from UCI repository. We select 5 datasets; iris data, pima-Indian diabetes data, glass data, abalone data and ecoli data. All of these datasets are numerical and have no missing value. The other reason that these datasets are chosen is their low number of data points and attributes.

4.1 The description of dataset

Iris data contains 150 records and 4 attributes. Each record represents the characteristic of the sepals and petals which can use to determine the types of iris. The descriptive statistics of all attributes are shown in Table 4.1.

Attribute Information:	Mean	Standard Deviation	Sample Variance	Range	Minimum	Maximum
1. Sepal length in cm	5.84330	0.82807	0.68569	3.60	4.30	7.90
2. Sepal width in cm	3.05400	0.43359	0.18800	2.40	2.00	4.40
3. Petal length in cm	3.75870	1.76440	3.11320	5.90	1.00	6.90
4. Petal width in cm	1.19870	0.76316	0.58241	2.40	0.10	2.50

Table 4.1 The descriptive statistics of all attributes in Iris data set

Iris data has 3 target classes, Iris Setosa, Iris Versicolour and Iris Virginica, The distribution of each class is equally divided, 50 records per class or 33.33% per class.

Glass data contains 214 records of glass described by the quantity of each component. The descriptive statistics of all attributes are shown in Table 4.2.

Attribute Information:	Mean	Standard Deviation	Sample Variance	Range	Minimum	Maximum
1. RI: Refractive index	1.51840	0.00304	0.00001	0.023	1.511	1.534
2. Na: Sodium	13.40800	0.81660	0.66684	6.650	10.730	17.380
3. Mg: Magnesium	2.68450	1.44240	2.08050	4.490	0.000	4.490
4. Al: Aluminum	1.44490	0.49927	0.24927	3.210	0.290	3.500
5. Si: Silicon	72.65100	0.77455	0.59992	5.600	69.810	75.410
6. K: Potassium	0.49706	0.65219	0.42535	6.210	0.000	6.210
7. Ca: Calcium	8.95700	1.42320	2.02540	10.760	5.430	16.190
8. Ba: Barium	0.17505	0.49722	0.24723	3.150	0.000	3.150
9. Fe: Iron	0.05701	0.09744	0.00949	0.510	0.000	0.510

Table 4.2 The descriptive statistics of all attributes in Glass data set

For the second to last attributes, the value is in the unit measurement: weight percent in corresponding oxide.

The data is divided into 7 classes or types of glass based on its usage. The target class is the usage types of glass whose distribution is shown in Table 4.3.

Class : The usage types of glass	The number of instances
1. building windows float processed	70 (33%)
2. building window non-float processed	76 (35%)
3. vehicle window float processed	17 (8%)
4. vehicle window non-float processed	0 (0%)
5. containers	13 (6%)
6. tableware	9 (4%)
7. headlamps	29 (14%)

Table 4.3 The target class distribution in Glass data set

Pima Indians diabetes data contains the diabetes-related information of the total 768 Pima and Indian people. There are 8 physical attributes collected from the test group of Pima and Indians all in numeric value. A list of attributes is :

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mmHg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (muU/ml)
6. Body mass index (weight in kg/(height in $m)^2$)
7. Diabetes pedigree function
8. Age (years)

The descriptive statistics of all attributes are shown in Table 4.4.

Attribute Information:	Mean	Standard Deviation	Sample Variance	Range	Minimum	Maximum
1. Number of times pregnant	3.84510	3.370	11.354	17.00	0.00	17.00
2. Plasma glucose concentration	120.89000	31.973	1,022.200	199.00	0.00	199.00
3. Diastolic blood pressure	69.10500	19.356	374.650	122.00	0.00	122.00
4. Triceps skin fold thickness	20.53600	15.952	254.470	99.00	0.00	99.00
5. 2-Hour serum insulin	79.79900	115.240	13,281.000	846.00	0.00	846.00
6. Body mass index	31.99300	7.884	62.160	67.10	0.00	67.10
7. Diabetes pedigree function	0.47188	0.331	0.110	2.34	0.08	2.42
8. Ages	33.24100	11.760	138.300	60.00	21.00	81.00

Table 4.4 The descriptive statistics of all attributes in Pima-Indians diabetes data set

The target class is the result of the diabetes test, 1 for positive and 0 for negative. Class Distribution is

Class	Number of instances
1	500 (65%)
0	268 (35%)

Table 4.5 The target class distribution in Pima-Indians diabetes data set

Abalone data is used for predicting the age of abalone from physical measurements. So the value of each attribute will be the value of physical appearance

of abalone. The abalone data record contains 4177 records with 8 attributes, which are listed below:

1. Sex
2. Length
3. Diameter
4. Height
5. Whole weight
6. Shucked weight
7. Viscera weight
8. Shell weight

However we exclude the first attribute which is a nominal value. So we use only 7 attributes. The descriptive statistics of all attributes are shown in Table 4.6.

Attribute Information:	Mean	Standard Deviation	Sample Variance	Range	Minimum	Maximum
1. Length	0.52399	0.12009	0.01442	0.7400	0.0750	0.8150
2. Diameter	0.40788	0.09924	0.00985	0.5950	0.0550	0.6500
3. Height	0.13952	0.04183	0.00175	1.1300	0.0000	1.1300
4. Whole weight	0.82874	0.49039	0.24048	2.8235	0.0020	2.8255
5. Shucked weight	0.35937	0.22196	0.04927	1.4870	0.0010	1.4880
6. Viscera weight	0.18059	0.10961	0.01202	0.7595	0.0005	0.7600
7. Shell weight	0.23883	0.13920	0.01938	1.0035	0.0015	1.0050

Table 4.6 The descriptive statistics of all attributes in abalone data set

This dataset contain a target class as ages which is the integer value varied from 1-29. This leads to a great number of classes. We modified this column into four

distinct values via its quartile to prevent the imbalance of a class distribution. The class distribution after modifying is shown in the following table.

Class (Quartile)	Number of instances
1 (within 25%)	1407 (34%)
2 (from 25% upto 50%)	689 (16%)
3 (from 50% upto 75%)	1121 (27%)
4 (above 75%)	960 (23%)

Table 4.7 The target class distribution in abalone data set

Ecoli data shows the localization site of proteins. It contains 336 records and 7 attributes. The detail of all attributes is listed below

Attribute Information

1. mcg: McGeoch's method for signal sequence recognition.
2. gvh: von Heijne's method for signal sequence recognition.
3. lip: von Heijne's Signal Peptidase II consensus sequence score.
4. chg: Presence of charge on N-terminus of predicted lipoproteins.
5. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
6. alm1: score of the ALOM membrane spanning region prediction program.
7. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence.

The descriptive statistics of all attributes are shown on Table 4.8.

Attribute Information:	Mean	Standard Deviation	Sample Variance	Range	Minimum	Maximum
1. mcg	0.50006	0.19463	0.03788	0.89	0.00	0.89
2. gvh	0.50000	0.14816	0.02195	0.84	0.16	1.00
3. lip	0.49548	0.08850	0.00783	0.52	0.48	1.00
4. chg	0.50149	0.02728	0.00074	0.50	0.50	1.00
5. aac	0.50003	0.12238	0.01498	0.88	0.00	0.88
6. alm1	0.50018	0.21575	0.04655	0.97	0.03	1.00
7. alm2	0.49973	0.20941	0.04385	0.99	0.00	0.99

Table 4.8 The descriptive statistics of all attributes in ecoli data set

The target class is the localization of proteins, which has a class distribution as in the table 4.9

Class (Quartile)	Number of instances
1. cp (cytoplasm)	143 (43%)
2. im (inner membrane without signal sequence)	77 (23%)
3. pp (perisplasm)	52 (15%)
4. imU (inner membrane, uncleavable signal sequence)	35 (10%)
5. om (outer membrane)	20 (6%)
6. omL (outer membrane lipoprotein)	5 (1%)
7. imL (inner membrane lipoprotein)	2 (1%)
8. imS (inner membrane, cleavable signal sequence)	2 (1%)

Table 4.9 The target class distribution in ecoli data set

Similar to Al-Harbi and Rayward-Smith's work, a solution for each case is a vector of weights which we called the attribute-scaling vector. From Al-Harbi's work, the initial point of the search was randomly chosen. Instead of unreliable random choices, we consider three candidates for initial vectors and do supervised *K*-means clustering with these vectors in various numbers of clusters. After several experiments of clustering, we choose the one that give the least average error for each data set.

The average error we got from each candidate of an initial scaling vector is shown in Table 4.10.

	$(1, 1, \ldots, 1)$	$\left(\frac{d_1-c_1}{b_1-a_1}, \frac{d_2-c_2}{b_2-a_2}, \ldots, \frac{d_n-c_n}{b_n-a_n}\right)$	$\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_n}\right)$
Iris data	0.070001	0.071667	0.074167
Glass data	0.455396	0.456245	0.459219
Pima data	0.311199	0.311849	0.312500
Abalone data	0.504635	0.496262	0.498206
Ecoli data	0.158278	0.158279	0.196428

Table 4.10 The average misclassification error on the candidates of initial vector.

The table suggests that the vector of one is the most suitable candidate to be used as an initial point in all datasets except Abalone data which the reciprocal of range is the better choice.

Data is partitioned with the ratio 80 : 20, using 80% of data set as the training set to train the scaling vector and build the model. The rest of data set is the test set which is used to evaluate the model. We also evaluate the whole data based on the trained model which the results will be shown in Appendix B. The experiments are then repeated for 5 different partitions with uniform sampling.

The number of clusters for each dataset is another parameter for supervised *K*-means clustering to be considered. In Al-Harbi and Rayward-Smith work, it was set to

be equal to the number of target class (2 for Pima-Indians diabetes). However in this thesis, we investigate the effect of the number of clusters on our method. More experiments with various numbers of clusters are performed. For example the iris dataset is tested from k = 3 up to 10, 7-14 for glass dataset, 2-9 for Pima-Indians diabetes data, 4-10 for abalone data and 8-15 for ecoli data.

Other settings are the length of initial interval and the minimum tolerance of final interval which both are set at 20 and 0.0001 respectively. Since the reduction ratio of the golden section line search which is $(0.618)^{v-1}$ with v as the number of iterations, each line search needs at least 26 iterations until reaching the final interval.

4.2 Result

This section contains the result discussions from our experiments. In overall, our algorithms, both cyclic coordinate and Hooke and Jeeves, are able to reduce the misclassification error in the training set on all datasets. But there are some differences on the evaluation and how suitable each dataset is to this supervised *K*-means clustering.

First, the iris dataset gives 10.67% misclassification error when we apply *K*-means clustering. After applying attribute-scaling vector optimization by cyclic coordinate method and Hooke and Jeeves method, the algorithms give the vector that provides the error as shown in the figure 4.1 along with the error before applying scaling vector.

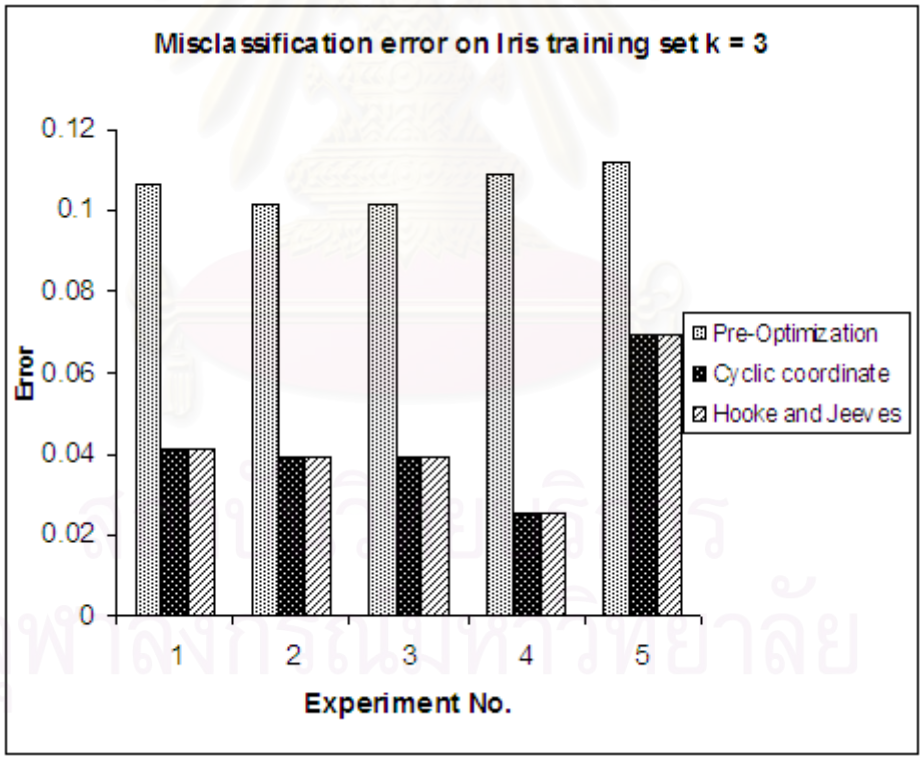

Fig.4.1 Misclassification error on Iris training set with $k = 3$

The result on this figure comes from the experiment which the number of clusters is set at $k = 3$. The error from clustering is significantly reduced by our algorithms in a significance rate which is 200-300 %.

In order to determine that our attribute-scaling vector from our algorithms is suitable for the classification, we evaluate it with the test set from previously partitioning. The result from the scaling vector with the same experiment as in figure 4.1 is shown in figure 4.2.

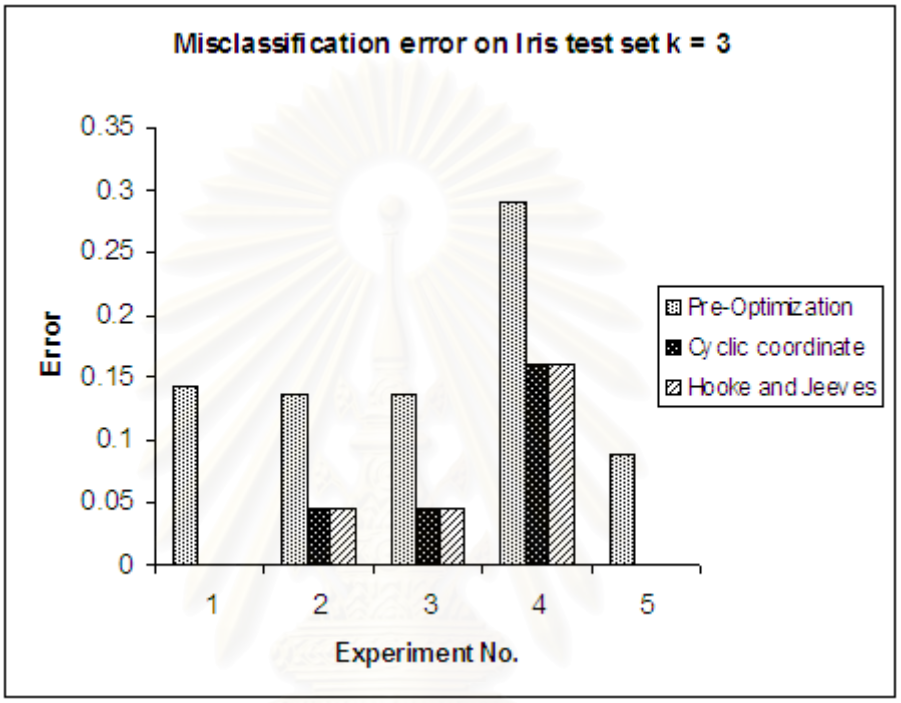

Fig.4.2 Misclassification error on Iris test set with $k = 3$

From the figure 4.2, Scaling vectors from experiment no.1 and no.5 give the error-free result. Even the case that gives the error about 16% (Experiment no.4) can be considered a good result since there are only 4-5 misclassified data points from total 30 data points in the test set. This shows that our algorithms work very well on iris data set.

Next, we consider the attribute-scaling vector we get from algorithms and see how it affects the clustering. We choose the example from experiment no.1 which has the attribute-scaling vector as [0, 0.0001, 0.6086, 1.8103]. Notice that very low values on the first and second elements suggest that the first and second attributes are not significant on classifying by clustering compared with the other two attributes. We observe the clustering in the axis of latter pair of attributes (petal length and petal width) and investigate the difference between the clustering without applying scaling

vector as in the figure 4.3 and the clustering with applying scaling vector as in the figure 4.4.

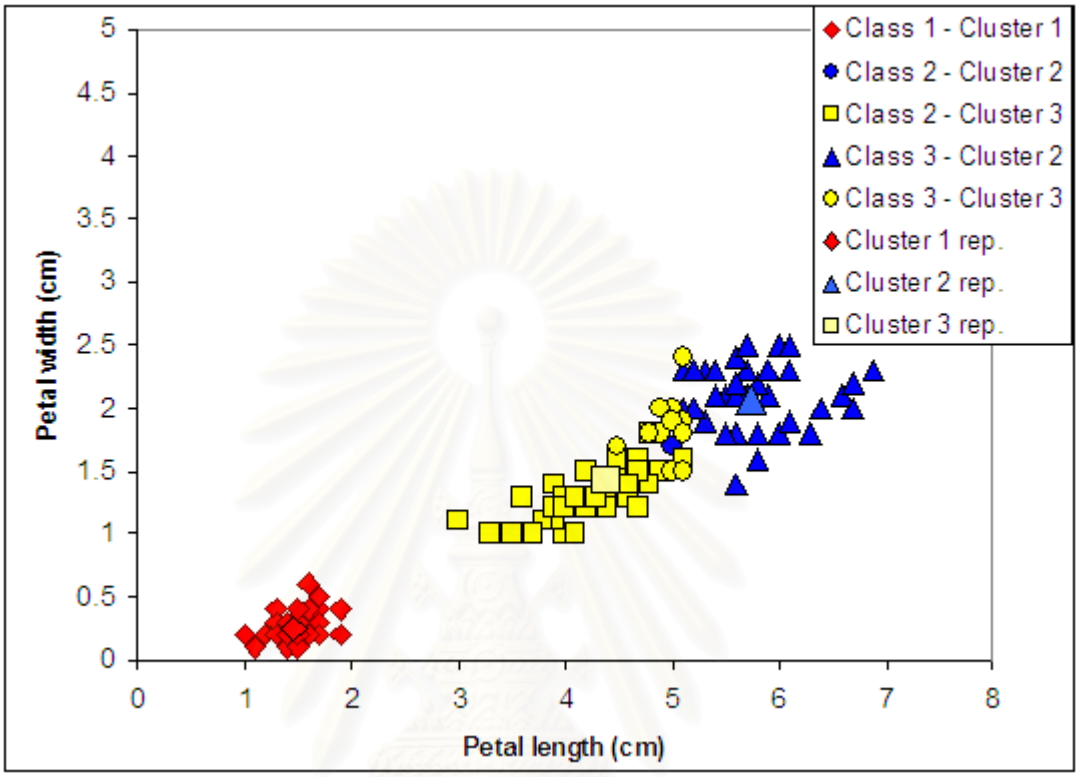

Fig.4.3 The graph of the clustering of iris data points without scaling vector

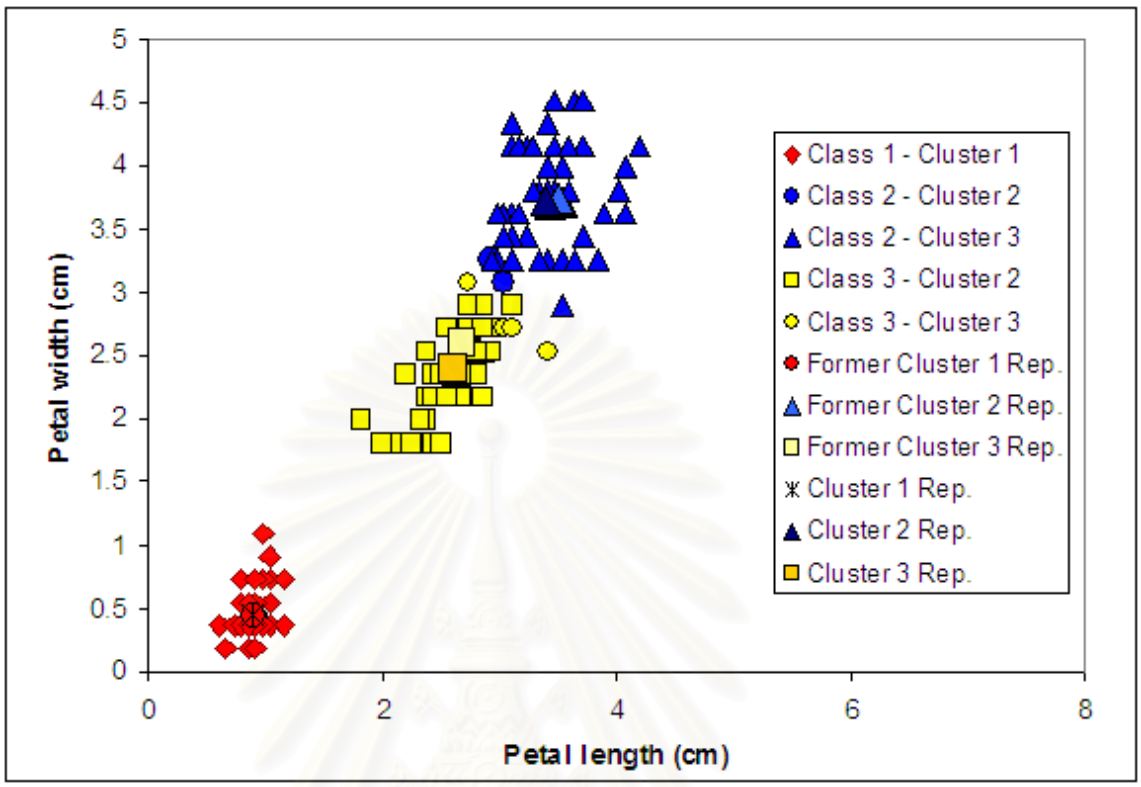

Fig.4.4 The graph of the clustering of iris data points with scaling vector

For both figures, their clusters and misclassified points are separated by the shape; diamond points for cluster 1, square points for cluster 2, triangle points for cluster 3 and circle points for misclassified points.

These two figures show the effect of scaling vector on clustering iris dataset. The first cluster which gathers at the bottom-left of the figure represents the diamond class (class 1), the second cluster in the middle represents the square class (class 2) and the last cluster which is on the top represents the triangle class (class 3). In the figure 4.3, there are 16 points that are misclassified (circle points) or around 10.67%. While after applying scaling vector as shown in the figure 4.4, the location and ratio of each data point has changed and also affected the clustering. The centroids have been moved and some elements change the cluster. We find that there are only 6 points (which is 4%) that are still misclassified and the error is effectively decreasing.

Another factor that we analyze is the difference between cyclic coordinate method and Hooke and Jeeves method which are used to optimizing attribute-scaling vectors. Both give the same optimal vector and the misclassification error. There is

only one difference between two algorithms which is the time using for both algorithms. Hooke and Jeeves method adds another search called pattern search. However after the experiment on iris dataset, the result shows slower running time as we see in the figure 4.5, while the optimal solution is the same.

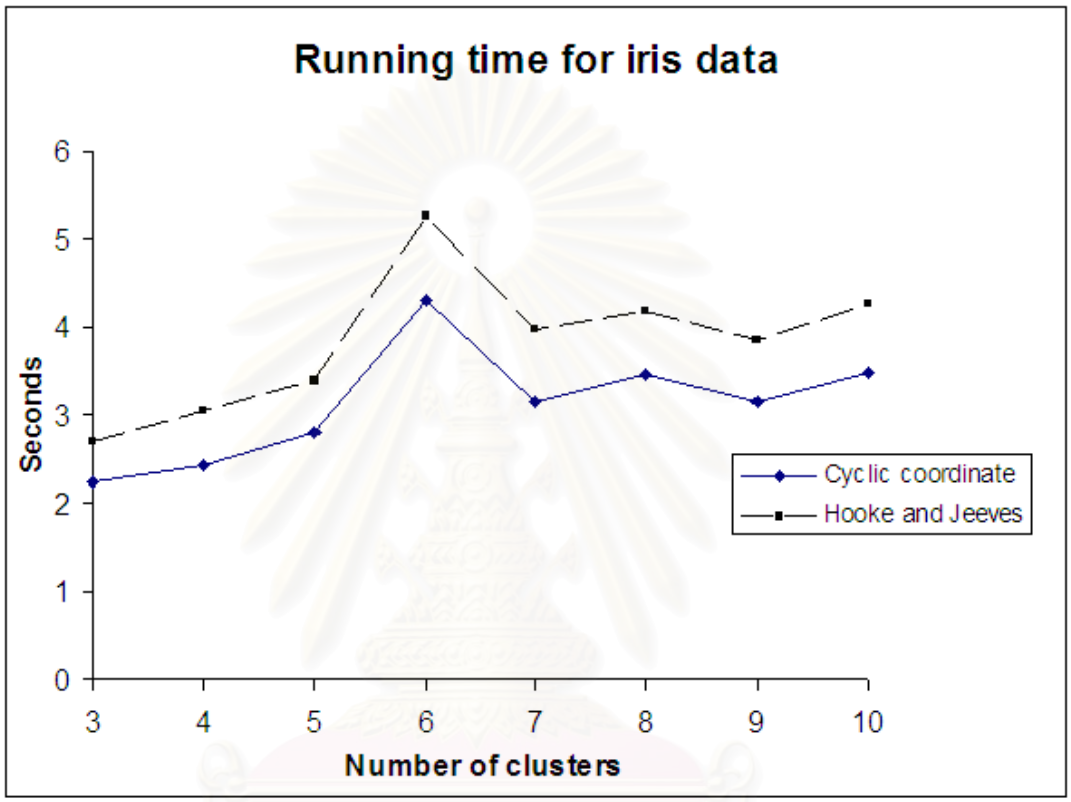

Fig.4.5 The graph of time used for cyclic coordinate method and Hooke and Jeeves method on Iris data set.

As seen in the figure, the running time used for Hooke and Jeeves method is larger than the running time for cyclic coordinate method about 20-25%. Both methods use only few iterations determining the optimal scaling vector. Therefore, Hooke and Jeeves method eventually uses more time due to additional pattern search.

The final analysis on iris dataset is the effect of the number of clusters for the supervised *K*-means algorithms. Normally, the bigger number of clusters should lead to less misclassification error due to the fact that the new cluster may group the minority data points that contain no other class. However this assumption may not

hold. The result of supervised *K*-means clustering with applying attribute-scaling vectors varied by the number of clusters for the iris dataset is shown in the figure 4.6

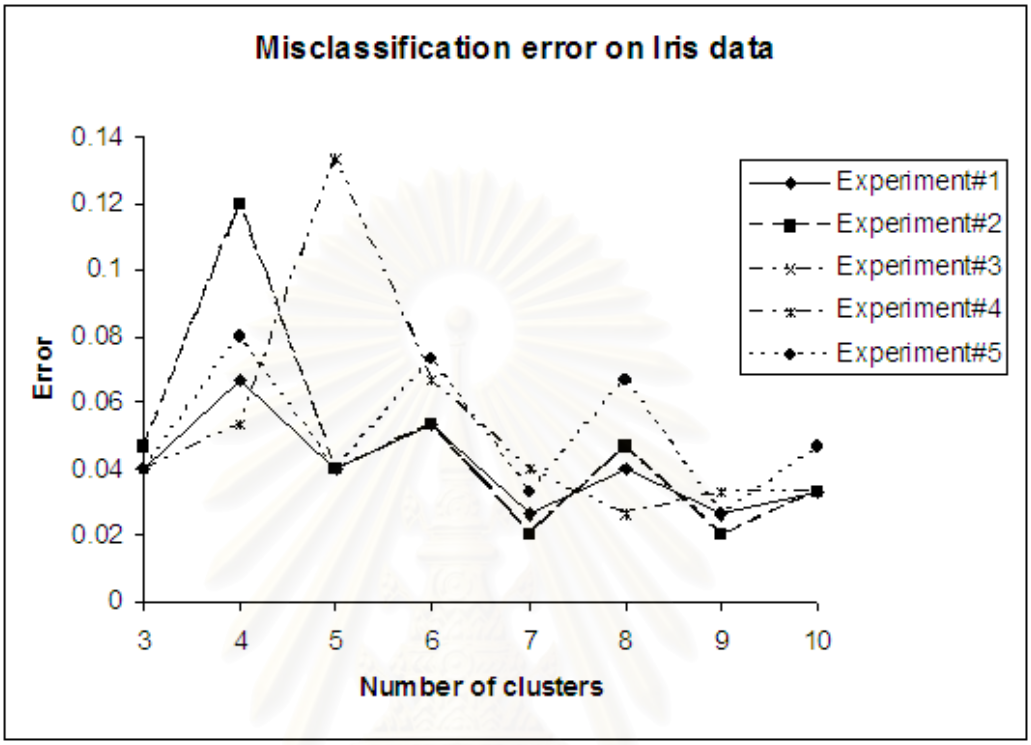

Fig.4.6 The misclassification error on iris dataset varied by the number of clusters

Figure 4.6 shows the fluctuation of the error when the numbers of clusters are small. As the number of clusters increases, the error decreases.

Iris dataset is a good example that can be classified by clustering. As previous result shows that our algorithms can effectively classify those three target class much more effectively. There is a research that introduced similar supervised K-means clustering with other approach on finding the attribute-scaling vector. Al-Harbi and Rayward-Smith [21] suggested Simulated Annealing to optimize the vector and perform the experiment on Pima-Indians diabetes dataset with the number of cluster $k = 2$. Their work had partitioned the dataset into 2 sets with the ratio 80:20, the former for training set and the latter for test set, which is the same as our experiment. Then, we compare the result from our algorithms to the result provided in that paper.

Clustering on Pima-Indians diabetes dataset with *K*-means algorithm without any modification gives the misclassification error of 0.3398 or around 34%. In the figure 4.7, we show the misclassification error from applying attribute-scaling vector on 5 sets of experiment in both algorithms compared with the error from *K*-means clustering without scaling vector on the same training sets.

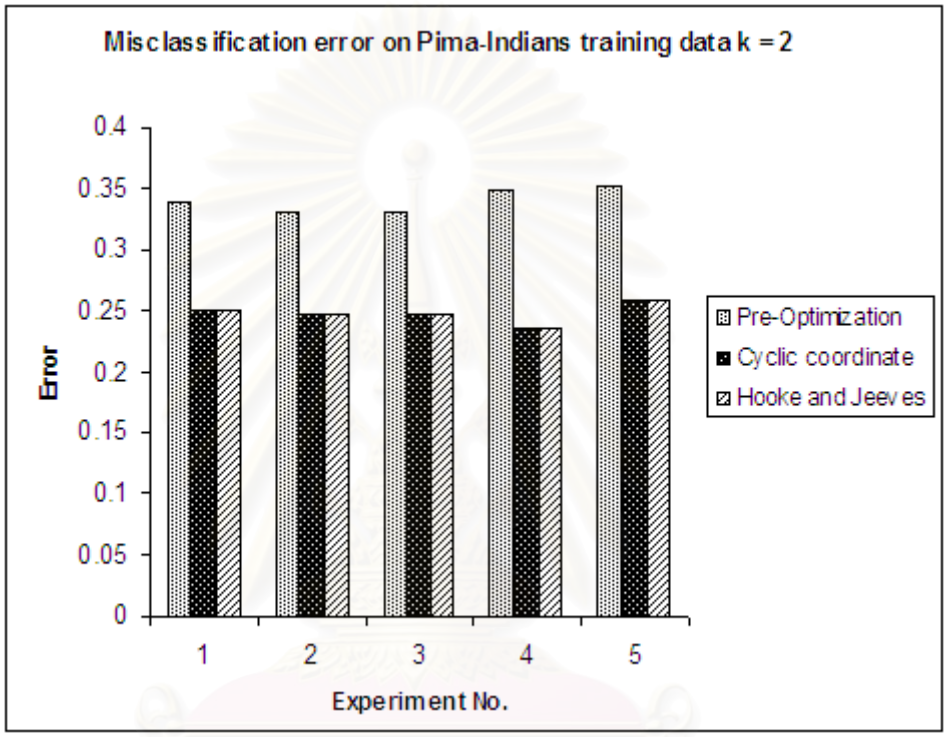

Fig.4.7 Misclassification error on Pima-Indians diabetes training set with $k = 2$

As shown in the figure, the reduction rate of error is not much as the rate from iris dataset, about 28-30% of the error. With the information of result in Al-Harbi and Rayward-Smith work, we compare our results to theirs as seen in the table 4.11.

	C4.5	Supervised K-means clustering				
		Simulated Annealing	Cyclic coordinate		Hooke and Jeeves	
Pima-Indians Diabetes	0.2631	0.2476	Average	0.24767	Average	0.24767
			Min	0.23491	Min	0.23491

Table 4.11 The error from the training set in each algorithm

For the table, C4.5 is a classification algorithm used to generate a decision tree developed by Ross Quinlan [36] and was considered as one of the effective classifiers. So Al-Harbi and Rayward-Smith use it to compare the result of their algorithm with the result from C4.5. In order to check the validity of these attribute-scaling vectors as a model for classification, we will evaluate with the test set (the other 20% of dataset) and see the result. The evaluation result is seen in the figure 4.8.

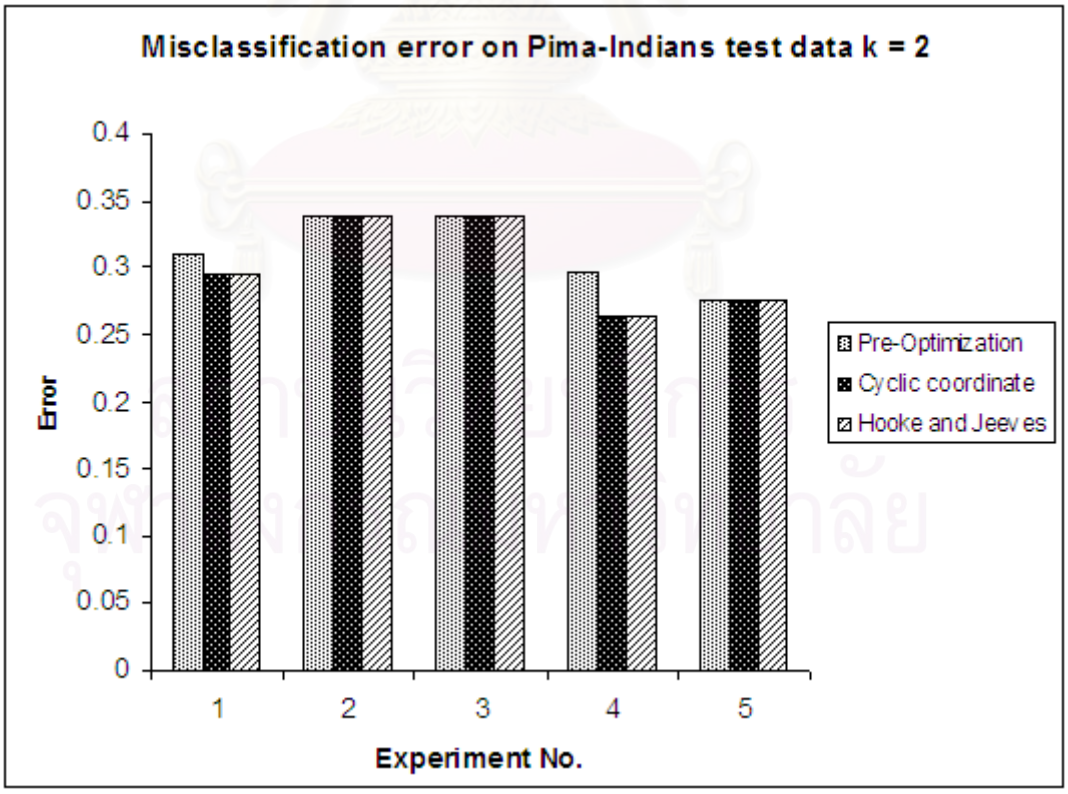

Fig.4.8 Misclassification error on Pima-Indians diabetes test set with $k = 2$

As seen in the figure 4.8, there are several experiments that our attribute-scaling vectors have the same misclassification error. This can be explained by uniform partition of dataset that fail to maintain the same target class distribution in the test set. We also compare the results to other algorithms in the table 4.12

	C4.5	Supervised K-means clustering				
		Simulated Annealing	Cyclic coordinate		Hooke and Jeeves	
Pima-Indians Diabetes	0.2850	0.2536	Average	0.30218	Average	0.30218
			Min	0.26452	Min	0.26452

Table 4.12 The error from the test set in each algorithm

Our work gives a slightly worse misclassification error than simulated annealing. Since the report about running time of simulated annealing is unavailable, we cannot compare and conclude about the running time between our works and theirs.

The time using for both algorithms in Pima-Indians diabetes dataset has the same trend as the time in iris dataset. The running time of Hooke and Jeeves method is larger than the running time of cyclic coordinate method about 10-15%. Again, the algorithms use only few rounds of iterations in finding the optimal attribute-scaling vector, so Hooke and Jeeves method cannot accelerate the multidimensional search and end up spend more time due to the additional pattern search. See the figure 4.9 for time comparison between two methods.

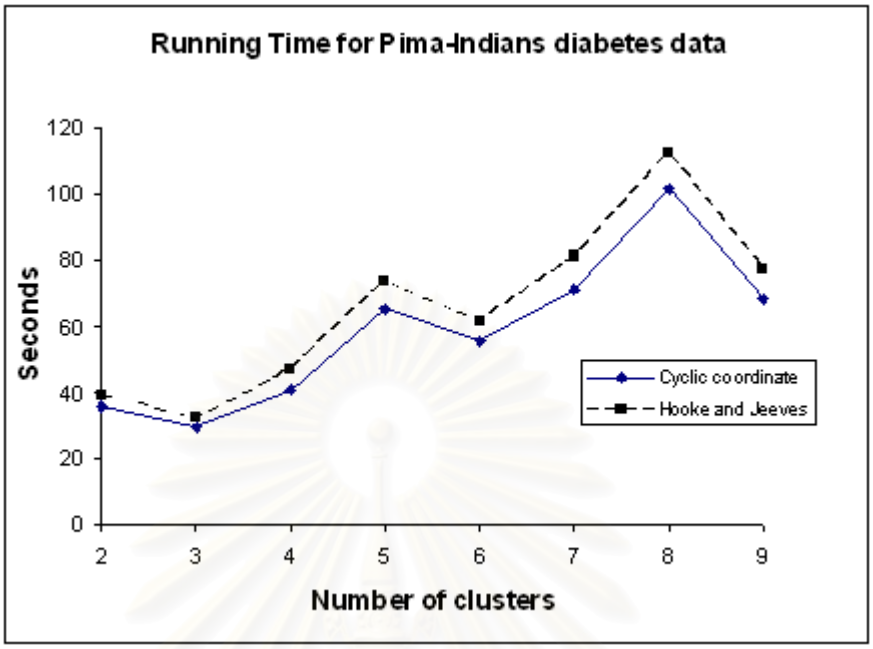

Fig.4.9 The graph of time used for cyclic coordinate method and Hooke and Jeeves method on Pima-Indians diabetes data set.

The rest of datasets do not show significant improvement on test set. Ecoli dataset is a dataset that gives a low misclassification error (around 18-23%), suggesting that this dataset might works well with supervised *K*-means clustering with applying attribute-scaling vector. And our algorithms also give the vector that provides lower error. See the figure 4.10 for error comparison between supervised *K*-means algorithm without scaling vector and with scaling vector.

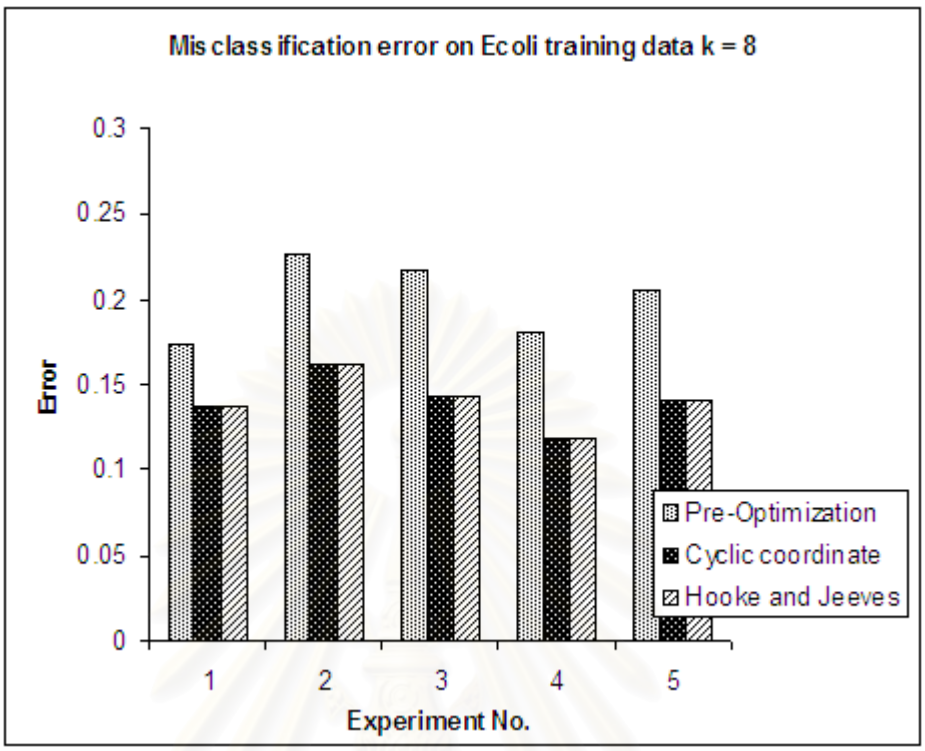

Fig.4.10 Misclassification error on Ecoli training set with $k = 8$

Our algorithms can provide the vectors that reduce the misclassification error from around 18-23% to 14-16% and look promising to adapt the supervised *K*-means clustering model. However, after we evaluate the vectors to their respectively test sets, we find these unsatisfying results as shown in the figure 4.11.

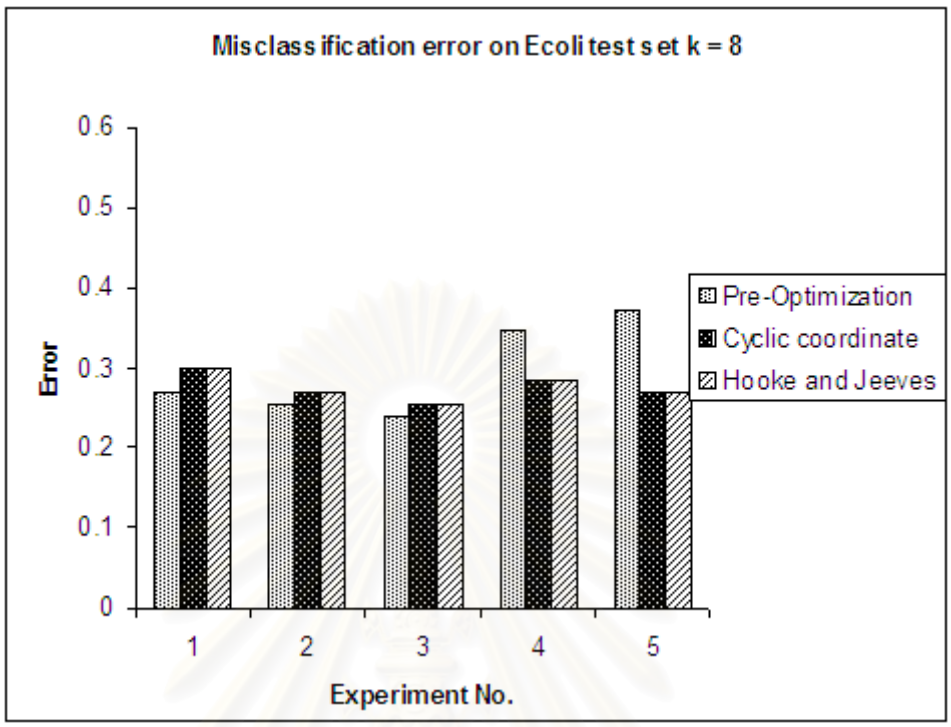

Fig.4.11 Misclassification error on Ecoli test set with $k = 8$

As seen in the figure 4.11, we see higher errors in the test sets on most of experiments. The reasons behind this failure might be the test set does not maintain the distribution of class and the dataset itself is imbalance which cause the test set fail to be a candidate evaluation dataset.

Other examples are the datasets of which clustering has a conflict with classification. The result from the experiment with glass dataset shows that supervised *K*-means clustering do not work well with this dataset. Even though our algorithms eventually reduce the number of error, the result is still hardly acceptable to be a successful method. See the figure 4.12 for the error before and after apply attribute-scaling vector we had trained in the training set of glass dataset with $k = 7$.

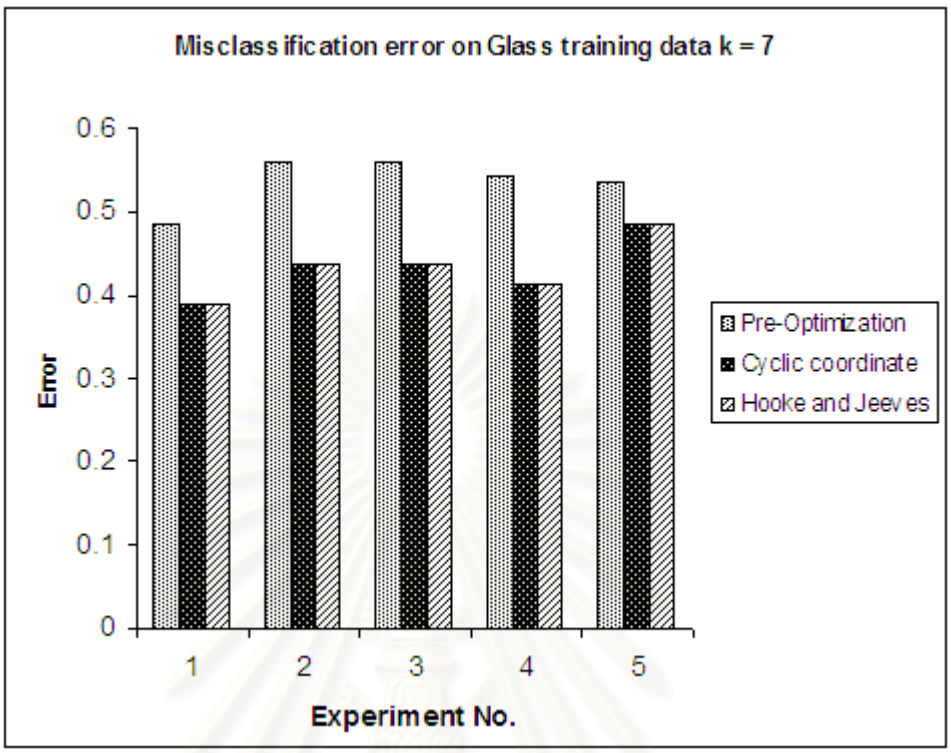

Fig.4.12 Misclassification error on Glass training set with $k = 7$

Our algorithms can reduce the misclassification error on the training set around 10%; however, the misclassification error after applying attribute-scaling vectors is still high.(around 40% of the dataset) This suggests that this dataset is not suitable for supervised *K*-means clustering. The results when we evaluate with the test set of their respective training set also show unsatisfying misclassification error, see the figure 4.13 for the results.

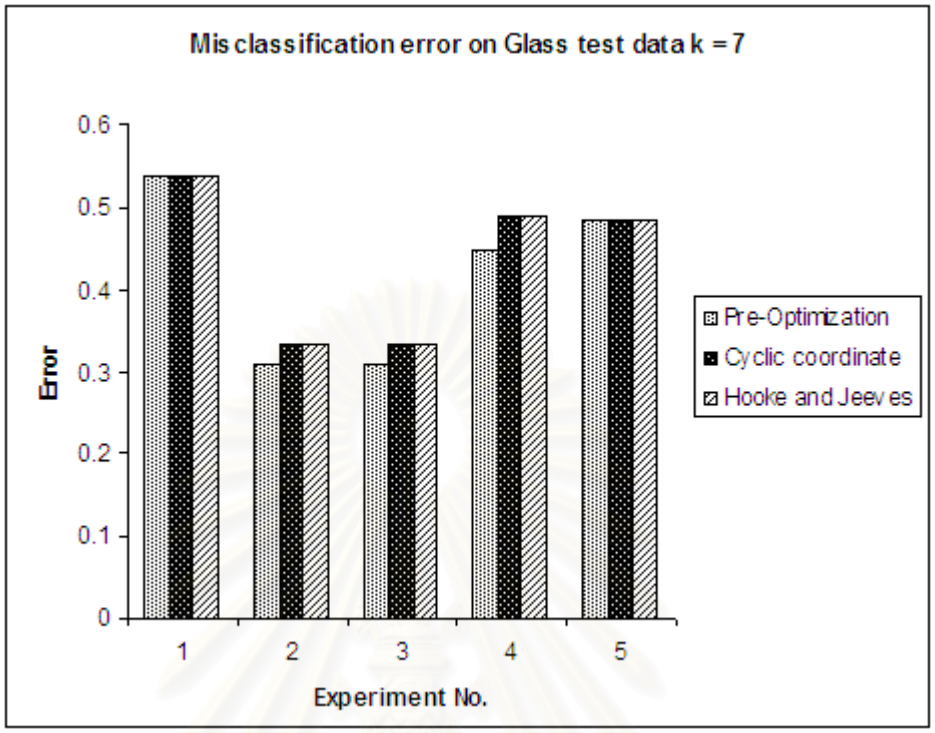

Fig.4.13 Misclassification error on Glass test set with $k = 7$

From the figure 4.13, three out of five experiments give the error from supervised *K*-means algorithm with scaling vector more than the error for supervised *K*-means algorithm without scaling vector. This may be interpreted that the classification model built by supervised *K*-means algorithm with attribute-scaling vector may not be suitable for glass dataset.

Another example of dataset that supervised *K*-means clustering does not work is abalone dataset which also gives a high misclassification error with supervised *K*-means algorithm. After we apply our algorithms finding an attribute-scaling vector, the vector reduces the error only a little compared with the previous error as shown in the figure 4.14.

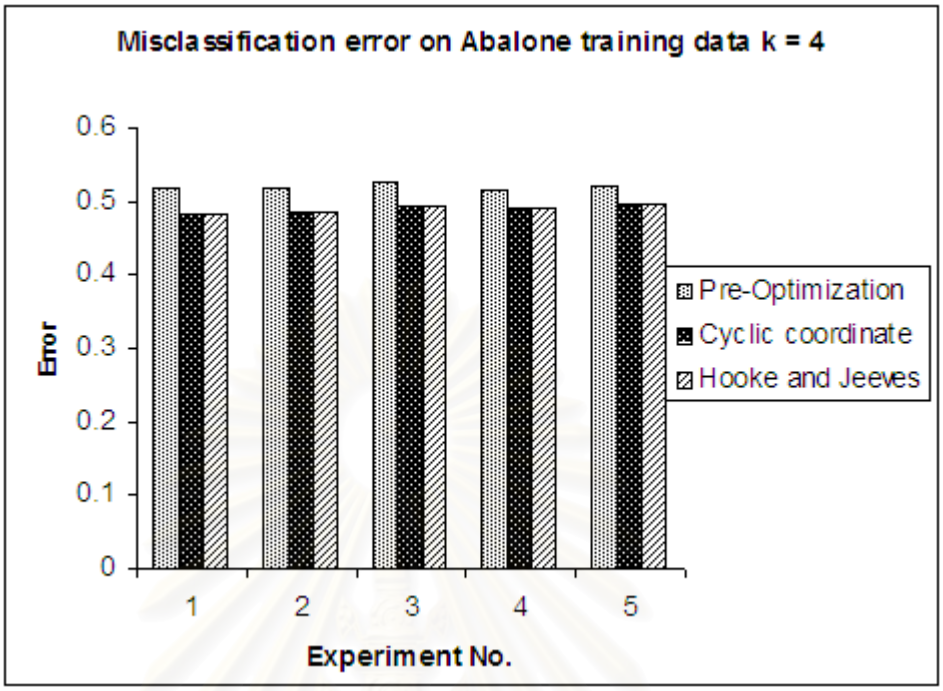

Fig.4.14 Misclassification error on Abalone training set with k = 4

From the figure 4.14, the reduction rate we retrieve after applying attribute-scaling vector is very low compared with supervised *K*-means clustering without scaling vector. Moreover, the error is still too high to be considered acceptable for this data set. To confirm this, we evaluate the model from scaling vector with the test set to see how well the model fits its data. The result is seen in the figure 4.15.

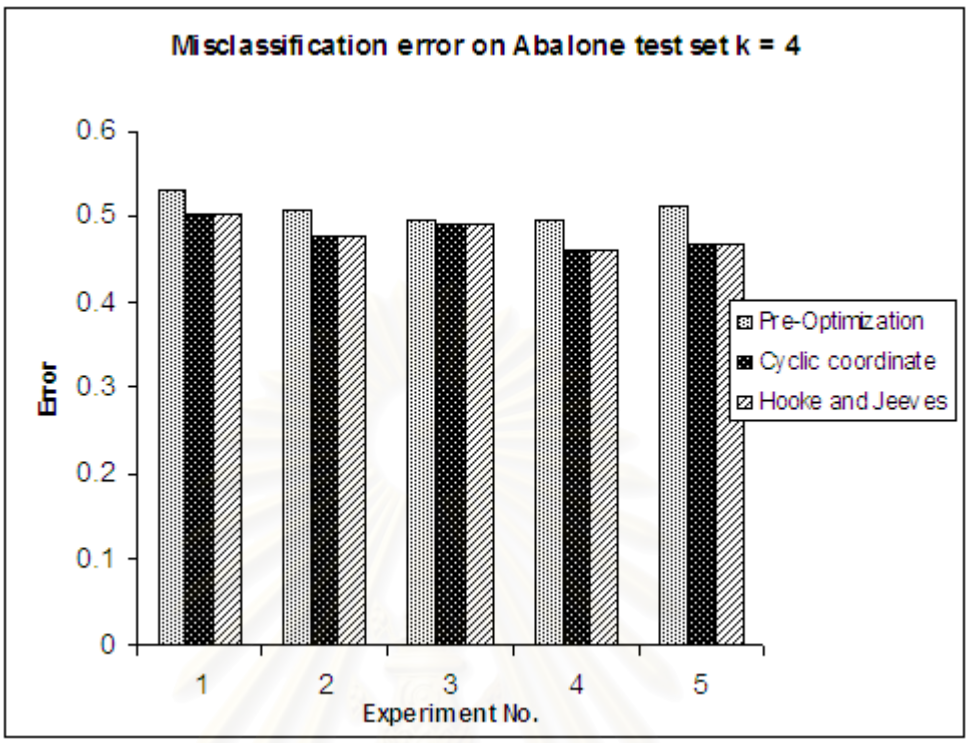

Fig.4.15 Misclassification error on Abalone test set with k = 4

From the figure 4.15, we find that our attribute-scaling vectors have lower misclassification error but these errors are too high (above 40%). This leads to the conclusion that abalone dataset is not suitable to apply supervised *K*-means clustering.

With the results from the experiments, the conclusion is discussed in the next chapter. The details of results from the experiments that were omitted are reported in Appendix.

CHAPTER V
CONCLUSION

5.1 Conclusion

Our experiments show that our algorithm can successfully reduce the misclassification error and give the effective supervised *K*-means clustering in two datasets, i.e. iris and Pima-Indians diabetes. The errors from algorithms with the cyclic coordinate and Hooke and Jeeves method in these two datasets are decreasing from the *K*-means clustering. On the other hand, the algorithm seems not to work well on the other datasets, glass, ecoli, and abalone. The success on the first two datasets may come from the characteristic of datasets themselves that their clustering are suitable for classification concept by having low distance function value among the data points in the same class and high value among the data points in the different class. So supervised *K*-means clustering with attribute-scaling vector can be applied and classified effectively. When we apply our algorithm to these datasets, the optimal vector we got will emphasis the attributes that can separate data points in the different class to different cluster and lead to the better clustering. For the other three datasets, the experiments show that they are not suitable to apply supervised *K*-means clustering and even though our algorithms can reduce the misclassification errors, they are still too high to be acceptable.

In order to determine future data set that can apply supervised *K*-means cluster with our attribute-scaling vector, we apply a measure called "Entropy" [37]. Entropy is a measure that determines the disorder of the group of data. In this case, the sets to be considered are the data clusters from *K*-means clustering without scaling vector. Since the misclassification error depends on the number of misclassified data point in each cluster, the value of cluster entropy will show the purity of data points in each cluster. The formula of the entropy is defined as :

$$H(i) = -\sum_{j \in T} p_i(Class_j) \cdot \log_2 p_i(Class_j)$$

where $H(i)$ is an entropy of cluster i. T is the set of target class. $p_i(\text{Class}_j)$ is the ratio of the data point with class j in cluster i.

If the entropy is high, it implies that each cluster will contain many misclassified points. This eventually gives the high error even though it will be reduced later by auto-scaling algorithm. Iris and Pima-Indians datasets give the value of cluster entropy less than 1 in every cluster (see table 5.1) while the entropy of cluster in glass and abalone data set are more than 1 in most clusters. Ecoli data set which is a data set that is good only in the training set give a couple clusters that their entropy is more than 1. From this measure, we conclude that the data set in which every cluster entropy is less than 1 can be applied in supervised *K*-means clustering effectively.

Cluster	1	2	3	4	5	6	7	8
Iris	0	0.3088	0.3912	0				
Pima-Indians	0.9987	0.8835						
Glass	0	1.4374	0.5159	1.4591	1.6405	0	1.5452	
Abalone	1.9909	1.1989	1.8283	1.3615				
Ecoli	0.1485	0.5189	0.1654	0.6030	1.9610	0.4855	0.8325	1.2025

Table 5.1 The entropy of each cluster after performing *K*-means clustering

Another conclusion that we have is that both cyclic coordinate method and Hooke and Jeeves method give the same solution or the same attribute-scaled vector. This means that pattern search direction may not help in this problem domain. One of the possible reasons is that there is a small number of iterations in most experiments before reaching the optimal solution; therefore, there is no iteration for Hooke and Jeeves method to skip. So, the number of line searches is not reduced. And with the addition of the pattern search in each iteration, the time used for Hooke and Jeeves method is more than the time used for cyclic coordinate.

Finally, changing the number of clusters does not provide any obvious trends for a better attribute-scaled vector. We can use the number of clusters on *K*-means clustering to be equal to the number of class since more number of clusters does not have a significant improvement on the misclassification error.

5.2 Future work

The similarity of this thesis and Al-Harbi and Rayward Smith is using the same Euclidean distance function with the similar multiplication of attribute-scaled vector. This supervised *K*-means clustering may work better if applying other distance function such as Manhattan distance function. Moreover, we can extend the problem by multiplying weight matrix ($n \times n$) to the distance metric [38]. Another extension will be to change the metric from Euclidean distance to Mahalanobis distance [39] which contains the correlation between variables. This will change the *n*-dimensional optimization problem to be n^2-dimensional problem.

Another choice of improvement is changing the optimization techniques to be able to find an optimal vector faster or determining the global optimal. There are many derivative-free optimization techniques that can be used for this direction.

After further improvements on this technique, we can apply this as an automatic scaling technique using in data pre-processing. Instead of spending time to manually scale the value of dataset or using the typical data normalization, we can let the machine to automatically scale the data into the suitable range according to the provided target class.

REFERENCE

[1] Hand, D. J., Smyth, P., and Mannila, H. Principles of data mining. Cambridge, MA : MIT Press, 2001.

[2] Chan, P.K., Fan, W., Prodromidis, A.L., and Stolfo, S.J. Distributed data mining in credit card fraud detection. Intelligent Systems and their Applications 14 , 6 (November 1999) : 545-556.

[3] Ng, M. K., and Huang, Z. Data-mining massive time series astronomical data: challenges, problems and solutions. Information and Software Technology 41, 9 (25 June 1999) : 545-556.

[4] Meinl, T., Borgelt, C., and Berthold, M.R. Mining Fragments with Fuzzy Chains in Molecular Databases. In Second International Workshop on Mining Graphs, Trees and Sequences, 49–60. Pisa, Italy : University of Pisa, 2004.

[5] Pyle, D.. Data preparation for data mining San Francisco, CA : Morgan Kaufmann, 1999.

[6] Kotsiantis, S. B., Kanellopoulos, D., and Pintelas, P. E. Data Preprocessing for Supervised Learning. International Journal of Computer Science 1, 2 (2006) : 112-117.

[7] Al-Shalabi, L., Shaaban, Z. Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix. Dependability of Computer Systems, 2006. DepCos-RELCOMEX '06. International Conference (25-27 May 2006) : 207-214

[8] Kriger, C., Tzoneva, R. Neural networks for prediction of wastewater treatment plant influent disturbances. AFRICON 2007 (26-28 September 2007) : 1-7.

[9] Dietterich T.G. Machine Learning. Annual Review of Computer Science 4 (Jun 1990) : 255-306.

[10] Fodor, I.K. A survey of dimension reduction techniques. LLNL Technical Report (June 2002).

[11] Zhang, Y.F., Mao, J.L., and Xiong, Z.Y. An efficient clustering algorithm. International Conference on Machine Learning and Cybernetics 1 (2003) : 261-265.

[12] Fukunaga, K. Introduction to statistical pattern recognition. 2nd ed.. Boston, MA : Academic Press, 1990.

[13] Jain, A.K., and Dubes, R.C. Algorithms for clustering data. Upper Saddle River, NJ : Prentice Hall, 1988.

[14] Squyres, J.M., Lumsdaine, A., Mccandless, B.C., and Stevenson, R.L. A cluster-based parallel image processing toolkit. In Proceeding of Society of Photo-optical Instrumentation Engineers (1995) : 228-239.

[15] Punj, G., and Stewart, D.W. Cluster Analysis in Marketing Research : Review and Suggestions for Application. Journal of Marketing Research (May 1983) : 134-148.

[16] Han, J., and Kamber, M. Data mining: concepts and techniques. San Francisco, CA : Morgan Kaufmann, 2000.

[17] Mitra, P., Pal, S.K., and Siddiqi, M.A. Non-convex clustering using expectation maximization algorithm with rough set initialization . Pattern Recognition Letters 24, 6(2003) : 863-873.

[18] Bradley, P. S., and Fayyad, U. M. Refining Initial Points for K-Means Clustering. In Proceedings of the Fifteenth International Conference on Machine Learning (July 1998) : 91-99.

[19] Wagstaff, K., Rogers, S. Constrained k-means clustering with background knowledge. Proceedings of the 18th International conference on Machine Learning (2001) : 577-584.

[20] Demiriz, A., Bennett, K.P., and Embrechts, M.J. Semi-supervised clustering using genetic algorithms. In Proceedings of ANNIE (1999) : 809-814.

[21] Al-Harbi, S.H., and Rayward-Smith, V.J. Adapting k-means for supervised clustering. Applied Intelligence 24 , 3 (June 2006) : 219 - 226.

[22] Eick, C.F.,and Zeidat, N. Using Supervised Clustering to Enhance Classifiers. Lecture Notes in Computer Science 3488 (2005) : 248-256.

[23] Bazaraa, M.S., Sherali, H.D., and Shetty, C.M. Nonlinear Programming. Theory and Algorithms. 2nd ed.. Singapore : Wiley, 1993.

[24] Kitano H. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In Proceeding of the 8th National Conference Artificial Intelligence (1990): 789-795.

[25] Kaufman, L., and Rousseeuw, P. Finding groups in data. New York, NY : Wiley, 1990.

[26] Ng, R.,and Han, J. Efficient and effective clustering methods for spatial data mining. In Proceedings of the 20th International Conference on Very Large Data Bases (1994) : 144-155.

[27] Guha, S., Rastogi, R., and Shim, K. CURE: An efficient clustering algorithm. in Proceeding of ACM SIGMOD International Conference Management of Data (1998) : 73-84.

[28] Zhang, Y.F., Mao, J.L., and Xiong, Z.Y. An efficient clustering algorithm. International Conference on Machine Learning and Cybernetics 1 (2003) : 261-265.

[29] Brimberg, J., Mladenovic, N., and Salhi, S. The multi-source Weber problem with constant opening cost. Journal of the Operational Research Society 55, 6 (June 2004) : 640-646.

[30] Garey, M.R., and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NPCompleteness. New York, NY : W. H. Freeman, 1979.

[31] Matousek, J. On approximate geometric k-clustering. Discrete and Computational Geometry 24 (2000) : 61-84.

[32] Lloyd, S.P. Least squares quantization in PCM. Information Theory 28 (1982) : 129-137.

[33] Basu, S., Banerjee, A., Mooney, R. Semi-supervised clustering. In Proceedings of the 19th international conference on machine learning (July 2002) : 19-26.

[34] Cohn, D., Caruana, R., McCallum, A. Semi-supervised clustering with user feedback. Technical Report TR2003-1892. Cornell University, 2003.

[35] Jourdan, L., Dhaenens, C.,and Talb, E.G. Clustering nominal and numerical data: a new distance concept for an hybrid genetic algorithm. In Proceeding of 4th European Conference on Evolutionary Computation in Combinatorial Optimization (April 2004) : 220-229.

[36] Quinlan, J. R. C4.5: Programs for Machine Learning. San Francisco, CA : Morgan Kaufmann, 1993.

[37] Cheng, C.H., Fu, A.W.,and Zhang, Y. Entropy-based subspace clustering for mining numerical data. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (1999) : 84-93.

[38] Xing, E.P., Ng, A.Y., Jordan, M.I., and Russell, S. Distance metric learning, with application to clustering with side-information. Advances in Neural Information Processing Systems 15 (2003) : 505-512.

[39] Mahalanobis, P.C. On the generalised distance in statistics. In Proceedings of the National Institute of Sciences of India 2, 1 (1936) : 49-55.

APPENDIX

APPENDIX
THE EXPERIMENT RESULTS

MATLAB 7.0 is used as a tool to implement thanks to the existing *K*-means algorithm on its statistic toolbox, while the experiments are performed through Pentium core2duo 2.13 GHz processer with 1GB RAM memory.

Databases for this thesis are gathering from UCI repository, which are iris data, pima-Indian diabetes data, glass data, abalone data and ecoli data. All of these datasets are numerical and have no missing value. Moreover, they all have low numbers of data points and attributes.

Data is partitioned with the ratio 80 : 20, using 80% of data set as the training set and the rest of data set is the test set. We also evaluate the whole data based on the trained model. The experiments are then repeated for 5 different partitions with uniform sampling.

Since we investigate the effect of the number of clusters on our method, more experiments with various numbers of clusters are performed. For the iris dataset, it is set from $k = 3$ up to 10, 7-14 for glass dataset, 2-9 for Pima-Indians diabetes data, 4-10 for abalone data and 8-15 for ecoli data.

Results on Cyclic Coordinate Method

Iris dataset #1 with cyclic coordinate method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.609	1.810	0.04098	0.10656	0.00000	0.04000	2.1996
4	0.000	0.617	1.000	1.000	0.04918	0.08197	0.14286	0.06667	2.1372
5	0.000	0.617	1.157	4.554	0.04098	0.08197	0.00000	0.04000	3.6036
6	0.000	0.000	1.000	6.381	0.04918	0.09836	0.00000	0.05333	2.9328
7	1.000	7.743	4.022	6.180	0.02459	0.04098	0.00000	0.02667	3.0732
8	0.000	0.907	1.307	2.175	0.03279	0.04098	0.00000	0.04000	4.0092
9	1.000	1.000	1.000	1.000	0.04098	0.04098	0.07143	0.02667	1.6536
10	1.732	1.000	1.833	1.732	0.02459	0.04098	0.00000	0.03333	3.9312

Iris dataset #2 with cyclic coordinate method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.03906	0.10156	0.04546	0.04667	2.2152
4	0.183	0.000	0.561	1.000	0.03906	0.13281	0.04546	0.12000	2.3556
5	1.000	2.496	2.496	15.015	0.03125	0.13281	0.04546	0.04000	3.0108
6	0.000	2.088	1.000	4.051	0.03906	0.08594	0.09091	0.05333	3.6348
7	0.000	1.149	0.876	1.000	0.02344	0.08594	0.00000	0.02000	3.1512
8	1.785	1.000	1.000	4.068	0.03125	0.07813	0.04546	0.04667	3.8688
9	1.000	1.483	1.483	1.483	0.02344	0.05469	0.00000	0.02000	4.2276
10	0.000	0.218	0.244	0.887	0.03125	0.06250	0.04546	0.03333	3.9780

Iris dataset #3 with cyclic coordinate method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.03906	0.10156	0.04546	0.04667	2.1684
4	0.183	0.000	0.561	1.000	0.03906	0.13281	0.04546	0.12000	2.2308
5	1.000	2.496	2.496	15.015	0.03125	0.13281	0.04546	0.04000	2.9172
6	0.000	2.088	1.000	4.051	0.03906	0.08594	0.09091	0.05333	3.6036
7	0.000	1.149	0.876	1.000	0.02344	0.08594	0.00000	0.02000	3.1200
8	1.785	1.000	1.000	4.068	0.03125	0.07813	0.04546	0.04667	3.8064
9	1.000	1.483	1.483	1.483	0.02344	0.05469	0.00000	0.02000	4.1340
10	0.000	0.218	0.244	0.887	0.03125	0.06250	0.04546	0.03333	3.9624

Iris dataset #4 with cyclic coordinate method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.01681	0.10924	0.16129	0.04000	2.2932
4	1.000	2.886	6.729	49.063	0.01681	0.12605	0.09677	0.05333	3.3696
5	2.160	0.000	1.000	1.773	0.09244	0.09244	0.16129	0.13333	2.1528
6	0.000	2.718	1.000	3.654	0.01681	0.09244	0.06452	0.06667	6.5676
7	1.652	1.000	2.987	5.301	0.01681	0.08403	0.06452	0.04000	2.8860
8	1.000	1.000	1.000	1.000	0.00840	0.00840	0.12903	0.02667	1.6692
9	0.000	1.008	1.175	1.000	0.00840	0.05882	0.09677	0.03333	3.8532
10	1.000	1.000	1.000	1.000	0.00840	0.00840	0.12903	0.03333	2.2152

Iris dataset #5 with cyclic coordinate method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.068	0.156	0.06897	0.11207	0.00000	0.04000	2.3400
4	0.315	0.000	0.750	0.677	0.04310	0.11207	0.05882	0.08000	2.0280
5	0.000	0.258	0.357	1.214	0.04310	0.10345	0.00000	0.04000	2.3712
6	0.000	5.797	1.000	3.777	0.04310	0.10345	0.11765	0.07333	4.7424
7	1.139	2.314	1.000	1.523	0.00862	0.06035	0.08824	0.03333	3.4944
8	1.101	1.000	2.559	1.000	0.03448	0.05172	0.05882	0.06667	3.9624
9	1.000	1.000	1.000	1.000	0.01724	0.01724	0.02941	0.02667	1.9032
10	1.543	2.191	1.000	1.543	0.00862	0.01724	0.08824	0.04667	3.3540

Glass Dataset #1 with cyclic coordinate method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	0.9078	1	6.5559	1	1	1	1	1
8	0	1	2.7384	7.2699	0.9467	6.1486	1	9.7942	0.0001
9	0	1.6321	1	4.4792	1	1	1	1	1
10	0	1	5.8414	9.2536	1	13.09	1	1.1254	7.6067
11	0	0.5974	1	1	1.8486	2.8703	1	1	1
12	0	1.6241	1.1914	2.4274	1	1	1	2.3642	1
13	12.145	1	1	1	1	1	1	1	0
14	0	3.5592	2.8562	7.6369	3.4861	15.354	3.9829	9.9458	2.0834

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.40571	0.48571	0.53846	0.44393	26.193
8	0.39429	0.45714	0.43590	0.40654	18.471
9	0.39429	0.48571	0.46154	0.46729	18.190
10	0.38857	0.48571	0.48718	0.44860	24.586
11	0.36571	0.38857	0.43590	0.47664	40.342
12	0.33714	0.41143	0.48718	0.45327	37.144
13	0.37143	0.37143	0.43590	0.42523	13.494
14	0.35429	0.39429	0.41026	0.39252	29.375

Glass Dataset #2 with cyclic coordinate method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0.0001	0	1	6.9189	2.0292	0.0001	1	2.7903	9.9164
8	0	1	1	5.8242	1	1	1	2.9903	9.1324
9	0.0001	1	3.4969	3.6818	0	0.0001	0.8739	1	4.6839
10	0	3.4535	3.6217	12.987	1	1	1	3.4036	9.3459
11	0	1	6.2353	3.5342	1	2.2081	1	14.744	3.637
12	0.0001	1.896	13.295	11.903	0.9878	8.8517	1	0	0.0001
13	0.0001	1	11.676	8.7998	0	1	1.9845	1	0.0001
14	0	1	1	6.4186	2.0607	10.71	3.2798	1.7846	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	35.568
8	0.41860	0.51163	0.38095	0.42991	25.257
9	0.41860	0.55233	0.28571	0.44393	38.688
10	0.41860	0.53488	0.30952	0.42991	25.803
11	0.40116	0.49419	0.28571	0.42991	29.515
12	0.40116	0.44767	0.28571	0.44393	28.938
13	0.39535	0.51163	0.23810	0.42523	34.991
14	0.41279	0.43605	0.21429	0.41589	23.416

Glass Dataset #3 with cyclic coordinate method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0.0001	0	1	6.9189	2.0292	0.0001	1	2.7903	9.9164
8	0	1	1	5.8242	1	1	1	2.9903	9.1324
9	0.0001	1	3.4969	3.6818	0	0.0001	0.8739	1	4.6839
10	0	3.4535	3.6217	12.987	1	1	1	3.4036	9.3459
11	0	1	6.2353	3.5342	1	2.2081	1	14.744	3.637
12	0.0001	1.896	13.295	11.903	0.9878	8.8517	1	0	0.0001
13	0.0001	1	11.676	8.7998	0	1	1.9845	1	0.0001
14	0	1	1	6.4186	2.0607	10.71	3.2798	1.7846	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	35.319
8	0.41860	0.51163	0.38095	0.42991	25.132
9	0.41860	0.55233	0.28571	0.44393	38.127
10	0.41860	0.53488	0.30952	0.42991	25.428
11	0.40116	0.49419	0.28571	0.42991	29.359
12	0.40116	0.44767	0.28571	0.44393	28.767
13	0.39535	0.51163	0.23810	0.42523	34.913
14	0.41279	0.43605	0.21429	0.41589	23.260

Glass Dataset #4 with cyclic coordinate method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	1	4.7185	7.211	1	11.99	1	0.0001	6.7581
8	0.0001	1.2872	1	5.818	1	0	1	6.5239	1
9	0	5.5235	1	10.382	1.6793	1.0746	6.8486	4.2051	0.0001
10	0	1	2.5015	8.0569	1	3.2332	1	3.9936	8.8526
11	0.0001	2.8489	15.224	13.939	1	4.5608	7.4788	0	18.479
12	0	1.1063	2.7288	1	1	1	1	1	1
13	0	0.0001	3.7934	1	1	1	1	0.0001	0.9982
14	0	1	5.6506	9.2255	1	0.9999	1	5.6719	3.8996

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	35.319
8	0.41860	0.51163	0.38095	0.42991	25.132
9	0.41860	0.55233	0.28571	0.44393	38.127
10	0.41860	0.53488	0.30952	0.42991	25.428
11	0.40116	0.49419	0.28571	0.42991	29.359
12	0.40116	0.44767	0.28571	0.44393	28.767
13	0.39535	0.51163	0.23810	0.42523	34.913
14	0.41279	0.43605	0.21429	0.41589	23.260

Glass Dataset #5 with cyclic coordinate method

# *k*	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	0.3637	1	4.2392	1	1	1	4.2323	7.4046
8	0.0001	2.6477	4.8489	10.464	2.2134	1	3.6086	4.561	0
9	0	1.0048	1	2.1828	1	6.8135	2.4007	0.0001	1
10	0	0.0001	4.2513	1	1	1	1	1	7.1836
11	1	1	1	1	1	1	1	1	1
12	0	1	1	5.0206	3.1872	1	1	0.97	1
13	0	1	3.0322	1	1	1	3.5503	1	1
14	0	1	1	4.9609	1.8001	7.4725	1	1	1

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.48619	0.53591	0.48485	0.46729	32.199
8	0.40331	0.54144	0.42424	0.43925	48.423
9	0.42541	0.44199	0.39394	0.44860	20.982
10	0.39227	0.42541	0.42424	0.51869	20.031
11	0.41436	0.41436	0.36364	0.42991	12.012
12	0.38122	0.40331	0.36364	0.47664	23.041
13	0.38122	0.42541	0.36364	0.43925	23.291
14	0.34807	0.41989	0.33333	0.40187	31.309

Pima-Indians diabetes Dataset #1 with cyclic coordinate method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0	11.312	0	17.708	0.895	7.4489	0.6766	7.5813
3	8.9392	8.7412	0	0.0001	1	0.0001	0.0001	5.6293
4	3.1414	9.7649	11.318	4.6477	1	1	0	5.7837
5	7.5274	14.875	6.0108	1	2.8234	3.7837	0	5.9473
6	7.0103	7.3605	9.932	1.5353	1	14.054	0	2.7613
7	19.544	15.309	4.0998	0.0001	1	0	0.0001	12.344
8	0	17.701	0.0001	0.0001	0.9586	5.2765	0.0001	10.01
9	0	14.892	1	1.7516	1.0813	10.608	0.0001	7.5915

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24960	0.34022	0.29496	0.25781	28.080
3	0.24483	0.34499	0.28058	0.25000	26.146
4	0.24960	0.33545	0.25899	0.25651	36.348
5	0.25437	0.33386	0.23741	0.25911	47.674
6	0.24960	0.31161	0.27338	0.25651	38.127
7	0.24324	0.30684	0.25899	0.25781	109.590
8	0.24642	0.29730	0.24460	0.25781	83.492
9	0.25119	0.30207	0.25180	0.25911	97.267

Pima-Indians diabetes Dataset #2 with cyclic coordinate method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0001	10.332	0.0001	15.273	1	0.9977	0.9918	0
3	14.049	8.3545	0.9989	6.095	1	7.5627	0	9.1034
4	13.16	8.6739	7.8553	2.2226	0.9404	13.879	0	1
5	0.9262	9.8098	5.7066	5.2204	1.0105	10.582	0	12.453
6	4.584	9.4339	7.4663	4.3731	1	11.52	0	10.826
7	22.175	12.144	1.9704	24.252	1.9704	15.161	1	5.5228
8	0	3.5673	0.4857	6.9448	1	5.1221	0.0001	7.996
9	1.8393	24.09	5.2438	1.8393	1.8393	1.8393	1	2.7279

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24765	0.33072	0.33846	0.26563	29.281
3	0.24451	0.34483	0.23846	0.25260	24.664
4	0.24295	0.33229	0.25385	0.25260	35.506
5	0.24451	0.32132	0.23846	0.24870	82.119
6	0.23824	0.30721	0.23846	0.25000	65.068
7	0.23668	0.31505	0.26923	0.25651	69.046
8	0.23511	0.30408	0.27692	0.25521	72.650
9	0.24765	0.29310	0.25385	0.25391	70.403

Pima-Indians diabetes Dataset #3 with cyclic coordinate method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0001	10.332	0.0001	15.273	1	0.9977	0.9918	0
3	14.049	8.3545	0.9989	6.095	1	7.5627	0	9.1034
4	13.16	8.6739	7.8553	2.2226	0.9404	13.879	0	1
5	0.9262	9.8098	5.7066	5.2204	1.0105	10.582	0	12.453
6	4.584	9.4339	7.4663	4.3731	1	11.52	0	10.826
7	22.175	12.144	1.9704	24.252	1.9704	15.161	1	5.5228
8	0	3.5673	0.4857	6.9448	1	5.1221	0.0001	7.996
9	1.8393	24.09	5.2438	1.8393	1.8393	1.8393	1	2.7279

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24765	0.33072	0.33846	0.26563	29.235
3	0.24451	0.34483	0.23846	0.25260	24.648
4	0.24295	0.33229	0.25385	0.25260	35.365
5	0.24451	0.32132	0.23846	0.24870	81.776
6	0.23824	0.30721	0.23846	0.25000	64.912
7	0.23668	0.31505	0.26923	0.25651	69.046
8	0.23511	0.30408	0.27692	0.25521	74.568
9	0.24765	0.29310	0.25385	0.25391	69.670

Pima-Indians diabetes Dataset #4 with cyclic coordinate method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0633	5.0134	0	0.0001	0.5076	15.287	0.0001	8.5433
3	0	7.7061	0.0001	1	0.89	0.9907	0.9644	11.424
4	3.1696	10.092	0	7.8915	1	7.5305	0.0001	10.972
5	9.0567	4.0916	4.1726	3.6589	1	9.4132	4.8055	8.764
6	1	4.3815	1	1	1	6.9382	0	0.0001
7	0	10.897	2.6559	2.8861	2.8761	15.671	0.0001	2.6083
8	5.033	7.3305	2.1853	8.4018	1	17.157	3.5265	14.431
9	7.7636	7.5938	1	9.7789	1	13.755	0	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.23491	0.34747	0.26452	0.24609	51.215
3	0.25122	0.34910	0.23871	0.25130	24.539
4	0.24633	0.34584	0.25161	0.24870	63.368
5	0.23817	0.30995	0.25161	0.25651	63.508
6	0.24633	0.31158	0.26452	0.25521	36.052
7	0.24633	0.31648	0.27742	0.25130	53.118
8	0.21860	0.30179	0.23871	0.23698	151.700
9	0.22838	0.30832	0.23226	0.26302	52.806

Pima-Indians diabetes Dataset #5 with cyclic coordinate method

# *k*	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0	8.9167	1.0859	14.722	0.9464	12.322	0.0001	9.1623
3	0	13.803	5.2428	11.889	1.3038	0.9258	0.988	9.8319
4	1.2546	10.275	8.2872	7.7335	1	14.36	1.1146	1.1805
5	6.7903	6.827	1.7963	1	1	12.329	0	0.3267
6	5.9019	12.09	4.3684	1	1	0.9996	0	13.9
7	6.4421	12.252	2.7717	1	1	0.9721	0	1
8	2.042	5.3111	7.7227	9.785	1	6.0737	0	10.851
9	0	15.068	7.2482	1.7555	1	1	0.0001	0.9999

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.25854	0.35122	0.27451	0.26693	41.605
3	0.25854	0.36098	0.20261	0.25130	47.081
4	0.24553	0.34309	0.19608	0.25260	34.632
5	0.25366	0.30244	0.20915	0.25651	52.136
6	0.25528	0.31382	0.18301	0.25000	74.990
7	0.25854	0.31382	0.18954	0.25781	54.132
8	0.25041	0.29919	0.22222	0.27604	125.720
9	0.24878	0.29756	0.19608	0.25781	51.777

Abalone Dataset #1 with cyclic coordinate method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	13.059	19.078	28.079	1.9176	1.3487	1	27.12
5	0	2.1009	19.12	0.6876	0.0001	1.9728	10.971
6	37.461	51.733	28.297	9.8398	8.0071	1	127.65
7	3.3045	17.959	2.0333	1	2.6721	3.452	39.756
8	3.7323	0	5.486	0.252	1	17.137	17.559
9	1.1021	1	10.333	1	8.2707	5.1352	16.089
10	4.8415	5.858	12.951	0	2.8401	0.3648	7.471

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.48286	0.51625	0.50441	0.48791	456.54
5	0.47813	0.52482	0.49937	0.48288	460.33
6	0.47961	0.51212	0.49559	0.48719	529.14
7	0.47872	0.50739	0.50315	0.48408	867.01
8	0.47193	0.51300	0.50315	0.49701	642.30
9	0.46188	0.50768	0.48802	0.47474	1259.40
10	0.45833	0.51152	0.49306	0.47307	821.67

Abalone Dataset #2 with cyclic coordinate method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	4.2758	0.0001	29.258	0	0.0001	0.9146	9.7525
5	3.489	12.009	2.8522	0.4732	0	3.5174	20.357
6	0.8589	16.394	1.4652	1	0	0.0001	19.023
7	1	13.72	10.257	1	0	1.9082	14.995
8	27.008	37.109	19.397	1	35.049	1.333	73.95
9	7.2405	1.8966	14.564	1	1.116	1.116	1.116
10	7.4082	1.0877	3.7893	1	1.7132	8.2176	12.356

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.48552	0.51655	0.47667	0.48863	833.50
5	0.48729	0.53221	0.47541	0.48480	618.87
6	0.49202	0.52009	0.47289	0.49390	602.44
7	0.49113	0.51300	0.47289	0.48839	1057.30
8	0.47872	0.51743	0.44388	0.48097	1280.70
9	0.48907	0.51537	0.48172	0.48839	530.47
10	0.47754	0.51596	0.45649	0.49437	1417.10

Abalone Dataset #3 with cyclic coordinate method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	5.925	8.6149	13.7	0.0001	0.0001	0	12.041
5	3.5006	10.883	0.0001	0.3618	0	0.9997	17.042
6	7.3533	7.2209	1.2721	1.7069	1	2.9012	20.343
7	0.9287	8.1773	0.0001	0	0.0001	3.6278	21.085
8	6.4845	5.7469	15.718	1.3598	14.819	1	31.951
9	11.351	11.973	7.517	0	9.0245	7.5946	19.003
10	7.8134	8.844	17.887	1.2273	4.2976	1	8.9629

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49255	0.52443	0.46285	0.48887	317.91
5	0.49225	0.53367	0.44823	0.48456	565.38
6	0.49315	0.52145	0.46285	0.48671	606.44
7	0.48927	0.51579	0.46529	0.48312	325.65
8	0.47765	0.51907	0.46650	0.47570	514.35
9	0.50328	0.51847	0.47016	0.47785	1053.50
10	0.47735	0.51996	0.44336	0.47283	1078.00

Abalone Dataset #4 with cyclic coordinate method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	19.818	1.7678	21.503	1	2.7071	1.7676	31.23
5	1	0	16.794	0.0001	1	3.5146	17.055
6	0.6196	2.5847	13.023	0.0007	0.0004	0	15.579
7	16.34	0	9.7004	1.7291	0.9998	2.8645	19.386
8	4.2118	18.318	12.223	1.2218	1	1.0492	25.018
9	0	1.304	0.0001	0.3745	9.7037	6.4624	18.546
10	28.768	3.4657	28.375	1	3.8052	3.8052	24.521

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49035	0.51469	0.49134	0.49102	373.37
5	0.48115	0.52894	0.48144	0.48504	455.35
6	0.48917	0.51410	0.48144	0.48504	601.10
7	0.49124	0.51113	0.49257	0.49318	672.26
8	0.48442	0.51083	0.48639	0.48719	1019.10
9	0.46661	0.51172	0.45545	0.47498	1522.30
10	0.47136	0.51291	0.48515	0.49078	826.01

Abalone Dataset #5 with cyclic coordinate method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	3.1605	13.783	15.346	0.8669	0	3.5166	16.075
5	5.8246	11.347	20.231	0	0.0001	4.7016	21.344
6	0	5.4619	17.912	0.4737	0.3824	2.8352	14.257
7	2.4614	5.075	58.221	1	15.038	12.267	26.783
8	8.1618	0	0.0001	1.4688	1.0065	0.9985	17.232
9	7.4963	7.3062	0	0.5228	8.7495	9.4012	16.904
10	7.1902	11.495	15.265	14.578	4.7501	0	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49676	0.51942	0.46983	0.48863	456.15
5	0.48558	0.53149	0.47240	0.48384	563.91
6	0.48852	0.51383	0.46727	0.48695	439.92
7	0.47528	0.51001	0.47240	0.50467	862.17
8	0.48764	0.51236	0.47368	0.48935	616.31
9	0.47028	0.51089	0.44544	0.47881	1079.70
10	0.50471	0.51207	0.50449	0.51017	706.95

Ecoli Dataset #1 with cyclic coordinate method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	0.6708	0.8238	2.8795	0	1	1	0.7894
9	1.3657	1	1	0	1	1.2766	0.6889
10	1.5182	1	1.3653	1.4271	1.3653	1.179	1.3653
11	3.0664	1.9167	1	5.2342	1.7949	2.1234	1.1684
12	1	0.9629	7.1769	0	1.7343	1	1.0848
13	1.0369	1.032	1.032	4.5661	1.032	1	1.0319
14	1.178	1.178	1	5.0193	1.178	1.3568	1.2173
15	3.2016	1	1	0	0.9587	1	0.5999

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.13718	0.17329	0.41026	0.18750	24.913
9	0.18051	0.19495	0.42308	0.16964	18.486
10	0.15162	0.18051	0.39744	0.16071	22.761
11	0.13718	0.17690	0.39744	0.16369	34.835
12	0.15162	0.19856	0.42308	0.17262	34.086
13	0.12635	0.15523	0.37179	0.17857	25.678
14	0.11191	0.13718	0.39744	0.19048	43.524
15	0.15162	0.17690	0.41026	0.17560	48.626

Ecoli Dataset #2 with cyclic coordinate method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1.468	1.0928	1	4.1087	1.0997	1.1934	1.3186
9	2.2337	1.0889	6.318	0	1.7079	1	1.738
10	0.9963	0.7353	0.5491	0.0001	1.2759	1.6497	0
11	1.3758	0.5256	1	0	1.3423	1.1518	0.6572
12	2.2636	1.159	1	3.8914	1.0664	1.2084	1.2644
13	1.1963	1.0976	3.7448	8.4239	1.7551	1.0976	1
14	1.6447	1.3242	1.8016	6.1208	1.7909	1	1.8013
15	322.09	209.83	123.82	1	239.61	201.02	239.61

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.16245	0.22744	0.39744	0.16667	26.302
9	0.16606	0.22022	0.39744	0.20833	35.100
10	0.14440	0.20939	0.43590	0.18155	54.975
11	0.15884	0.17690	0.42308	0.17262	32.245
12	0.13718	0.17690	0.39744	0.16071	33.259
13	0.13357	0.20217	0.38462	0.15476	54.429
14	0.10469	0.12996	0.38462	0.14583	24.757
15	0.11552	0.13357	0.38462	0.15179	33.665

Ecoli Dataset #3 with cyclic coordinate method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1	1.22	9.4084	0	0.9421	0.9995	1
9	1.5913	1	0	0.0001	1.0206	1	0.5753
10	2.2351	1	0.4827	0.0001	0	1.716	0.9998
11	0.6102	1.008	7.6186	0	1.0878	0.8723	0.8377
12	2.4797	1.8585	1	0	1	1	1.177
13	0.9363	1.1741	4.9175	0	1.6857	1	1
14	1	1.3079	1.0291	1.3079	1.3079	1.3079	1.3079
15	1.9447	1.8338	1	1.8338	1.8331	1.8338	1.8338

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.14440	0.21661	0.32468	0.22024	12.558
9	0.13718	0.18773	0.29870	0.16369	28.611
10	0.17329	0.20939	0.29870	0.20536	29.484
11	0.14801	0.16606	0.27273	0.17560	31.746
12	0.16968	0.18773	0.28571	0.18452	19.750
13	0.12635	0.15162	0.29870	0.17560	48.938
14	0.13718	0.15884	0.31169	0.18155	22.948
15	0.12996	0.14440	0.29870	0.12798	25.584

Ecoli Dataset #4 with cyclic coordinate method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1	2.2906	10.195	26.379	2.2906	2.2906	2.2906
9	1.3295	2.0757	9.7879	0	1.3708	1	1.3368
10	1	0.9484	6.5982	0	1.026	1	1.1579
11	1	1.0914	0.5948	0.00014	0.84777	1	0.9064
12	4.8698	8.2115	4.6992	1	8.2115	8.2115	8.2115
13	0.7257	0.2923	1	0	0.9389	0.8102	0.7578
14	2.4338	1	1.3765	4.8448	1.9428	1.6061	1.641
15	2.505	1.8566	3.9824	10.906	2.992	1.8566	1

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.11913	0.18051	0.34615	0.20536	15.04
9	0.13357	0.19495	0.26923	0.18452	35.209
10	0.11552	0.15162	0.35897	0.16964	29.344
11	0.13357	0.14079	0.32051	0.15774	15.975
12	0.10108	0.12635	0.26923	0.16071	18.034
13	0.12635	0.18051	0.26923	0.17857	44.398
14	0.11913	0.21300	0.23077	0.13988	46.878
15	0.11913	0.16606	0.25641	0.13690	34.133

Ecoli Dataset #5 with cyclic coordinate method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1.7933	1	1.4849	1.5419	1.1931	1.2059	1.5419
9	0.534	1	1	0	1	1	1
10	1.235	1	0.9995	0	1	1	1
11	2.0635	1.1114	1	3.3212	1	1	1
12	1.0668	1.0664	1.0664	4.0594	1	1.0664	1.0664
13	1.213	0.925	3.9604	0	1.2651	1	0.7563
14	1	1	1	2.6957	1	1.7917	1.0047
15	2.4585	1	1.0808	5.7633	1.2962	1.8634	1.0808

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.14079	0.20578	0.38462	0.19643	14.524
9	0.17690	0.18051	0.34615	0.16071	20.951
10	0.15523	0.15884	0.35897	0.15476	14.430
11	0.12274	0.15523	0.35897	0.17560	34.492
12	0.12274	0.14440	0.34615	0.13690	22.542
13	0.12635	0.13718	0.35897	0.16964	22.417
14	0.10830	0.12274	0.34615	0.15476	25.631
15	0.10830	0.18051	0.34615	0.13690	35.677

Results on Hooke and Jeeves Method

Iris dataset #1 with Hooke and Jeeves method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.609	1.810	0.04098	0.10656	0.00000	0.04000	2.6052
4	0.000	0.617	1.000	1.000	0.04918	0.08197	0.14286	0.06667	2.5272
5	0.000	0.617	1.157	4.554	0.04098	0.08197	0.00000	0.04000	4.4616
6	0.000	0.000	1.000	6.381	0.04918	0.09836	0.00000	0.05333	3.6504
7	1.000	7.743	4.022	6.180	0.02459	0.04098	0.00000	0.02667	3.8064
8	0.000	0.907	1.307	2.175	0.03279	0.04098	0.00000	0.04000	4.9296
9	1.000	1.000	1.000	1.000	0.04098	0.04098	0.07143	0.02667	2.0436
10	1.732	1.000	1.833	1.732	0.02459	0.04098	0.00000	0.03333	4.6644

Iris dataset #2 with Hooke and Jeeves method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.03906	0.10156	0.04546	0.04667	2.652
4	0.183	0.000	0.561	1.000	0.03906	0.13281	0.04546	0.12000	3.0108
5	1.000	2.496	2.496	15.02	0.03125	0.13281	0.04546	0.04000	3.6192
6	0.000	2.088	1.000	4.051	0.03906	0.08594	0.09091	0.05333	4.6332
7	0.000	1.149	0.876	1.000	0.02344	0.08594	0.00000	0.02000	4.134
8	1.785	1.000	1.000	4.068	0.03125	0.07813	0.04546	0.04667	4.6488
9	1.000	1.483	1.483	1.483	0.02344	0.05469	0.00000	0.02000	5.148
10	0.000	0.218	0.244	0.887	0.03125	0.06250	0.04546	0.03333	4.8204

Iris dataset #3 with Hooke and Jeeves method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.03906	0.10156	0.04546	0.04667	2.6208
4	0.183	0.000	0.561	1.000	0.03906	0.13281	0.04546	0.12000	3.0108
5	1.000	2.496	2.496	15.02	0.03125	0.13281	0.04546	0.04000	3.51
6	0.000	2.088	1.000	4.051	0.03906	0.08594	0.09091	0.05333	4.4772
7	0.000	1.149	0.876	1.000	0.02344	0.08594	0.00000	0.02000	4.0716
8	1.785	1.000	1.000	4.068	0.03125	0.07813	0.04546	0.04667	4.524
9	1.000	1.483	1.483	1.483	0.02344	0.05469	0.00000	0.02000	5.0388
10	0.000	0.218	0.244	0.887	0.03125	0.06250	0.04546	0.03333	4.7112

Iris dataset #4 with Hooke and Jeeves method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.000	0.000	0.01681	0.10924	0.16129	0.04000	2.7144
4	1.000	2.886	6.729	49.063	0.01681	0.12605	0.09677	0.05333	4.1652
5	2.160	0.000	1.000	1.773	0.09244	0.09244	0.16129	0.13333	2.5896
6	0.000	2.718	1.000	3.654	0.01681	0.09244	0.06452	0.06667	7.8
7	1.652	1.000	2.987	5.301	0.01681	0.08403	0.06452	0.04000	3.7596
8	1.000	1.000	1.000	1.000	0.00840	0.00840	0.12903	0.02667	2.1372
9	0.000	1.008	1.175	1.000	0.00840	0.05882	0.09677	0.03333	4.6488
10	1.000	1.000	1.000	1.000	0.00840	0.00840	0.12903	0.03333	2.8548

Iris dataset #5 with Hooke and Jeeves method

#k	Sepal length	Sepal width	Petal length	Petal width	Error				Running Time
					Training	Pre-Training	Test	Whole data	
3	0.000	0.000	0.068	0.156	0.06897	0.11207	0.00000	0.04000	2.8704
4	0.315	0.000	0.750	0.677	0.04310	0.11207	0.05882	0.08000	2.5272
5	0.000	0.258	0.357	1.214	0.04310	0.10345	0.00000	0.04000	2.7612
6	0.000	5.797	1.000	3.777	0.04310	0.10345	0.11765	0.07333	5.7564
7	1.139	2.314	1.000	1.523	0.00862	0.06035	0.08824	0.03333	4.1184
8	1.101	1.000	2.559	1.000	0.03448	0.05172	0.05882	0.06667	4.6644
9	1.000	1.000	1.000	1.000	0.01724	0.01724	0.02941	0.02667	2.3868
10	1.543	2.191	1.000	1.543	0.00862	0.01724	0.08824	0.04667	4.2432

Glass Dataset #1 with Hooke and Jeeves method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	0.9078	1	6.5559	1	1	1	1	1
8	0	1	2.7384	7.2699	0.9467	6.1486	1	9.7942	0.0001
9	0	1.6321	1	4.4792	1	1	1	1	1
10	0	1	5.8414	9.2536	1	13.09	1	1.1254	7.6067
11	0	0.5974	1	1	1.8486	2.8703	1	1	1
12	0	1.6241	1.1914	2.4274	1	1	1	2.3642	1
13	12.145	1	1	1	1	1	1	1	0
14	0	3.5592	2.8562	7.6369	3.4861	15.354	3.9829	9.9458	2.0834

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.40571	0.48571	0.53846	0.44393	17.566
8	0.39429	0.45714	0.43590	0.40654	30.249
9	0.39429	0.48571	0.46154	0.46729	17.550
10	0.38857	0.48571	0.48718	0.44860	80.075
11	0.36571	0.38857	0.43590	0.47664	29.812
12	0.33714	0.41143	0.48718	0.45327	35.958
13	0.37143	0.37143	0.43590	0.42523	28.111
14	0.35429	0.39429	0.41026	0.39252	42.292

Glass Dataset #2 with Hooke and Jeeves method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0.0001	0	1	6.9189	2.0292	0.0001	1	2.7903	9.9164
8	0	1	1	5.8242	1	1	1	2.9903	9.1324
9	0.0001	1	3.4969	3.6818	0	0.0001	0.8739	1	4.6839
10	0	3.4535	3.6217	12.987	1	1	1	3.4036	9.3459
11	0	1	6.2353	3.5342	1	2.2081	1	14.744	3.637
12	0.0001	1.896	13.295	11.903	0.9878	8.8517	1	0	0.0001
13	0.0001	1	11.676	8.7998	0	1	1.9845	1	0.0001
14	0	1	1	6.4186	2.0607	10.71	3.2798	1.7846	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	35.568
8	0.41860	0.51163	0.38095	0.42991	25.257
9	0.41860	0.55233	0.28571	0.44393	38.688
10	0.41860	0.53488	0.30952	0.42991	25.803
11	0.40116	0.49419	0.28571	0.42991	29.515
12	0.40116	0.44767	0.28571	0.44393	28.938
13	0.39535	0.51163	0.23810	0.42523	34.991
14	0.41279	0.43605	0.21429	0.41589	23.416

Glass Dataset #3 with Hooke and Jeeves method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0.0001	0	1	6.9189	2.0292	0.0001	1	2.7903	9.9164
8	0	1	1	5.8242	1	1	1	2.9903	9.1324
9	0.0001	1	3.4969	3.6818	0	0.0001	0.8739	1	4.6839
10	0	3.4535	3.6217	12.987	1	1	1	3.4036	9.3459
11	0	1	6.2353	3.5342	1	2.2081	1	14.744	3.637
12	0.0001	1.896	13.295	11.903	0.9878	8.8517	1	0	0.0001
13	0.0001	1	11.676	8.7998	0	1	1.9845	1	0.0001
14	0	1	1	6.4186	2.0607	10.71	3.2798	1.7846	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	41.028
8	0.41860	0.51163	0.38095	0.42991	28.033
9	0.41860	0.55233	0.28571	0.44393	43.150
10	0.41860	0.53488	0.30952	0.42991	27.987
11	0.40116	0.49419	0.28571	0.42991	32.292
12	0.40116	0.44767	0.28571	0.44393	32.074
13	0.39535	0.51163	0.23810	0.42523	38.392
14	0.41279	0.43605	0.21429	0.41589	25.584

Glass Dataset #4 with Hooke and Jeeves method

# k	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	1	4.7185	7.211	1	11.99	1	0.0001	6.7581
8	0.0001	1.2872	1	5.818	1	0	1	6.5239	1
9	0	5.5235	1	10.382	1.6793	1.0746	6.8486	4.2051	0.0001
10	0	1	2.5015	8.0569	1	3.2332	1	3.9936	8.8526
11	0.0001	2.8489	15.224	13.939	1	4.5608	7.4788	0	18.479
12	0	1.1063	2.7288	1	1	1	1	1	1
13	0	0.0001	3.7934	1	1	1	1	0.0001	0.9982
14	0	1	5.6506	9.2255	1	0.9999	1	5.6719	3.8996

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.43605	0.55814	0.33333	0.50935	16.817
8	0.41860	0.51163	0.38095	0.42991	39.250
9	0.41860	0.55233	0.28571	0.44393	33.103
10	0.41860	0.53488	0.30952	0.42991	32.323
11	0.40116	0.49419	0.28571	0.42991	35.443
12	0.40116	0.44767	0.28571	0.44393	38.626
13	0.39535	0.51163	0.23810	0.42523	19.968
14	0.41279	0.43605	0.21429	0.41589	39.375

Glass Dataset #5 with Hooke and Jeeves method

# *k*	Refractive index	Sodium	Mg	Al.	Silicon	K.	Calcium	Barium	Iron
7	0	0.3637	1	4.2392	1	1	1	4.2323	7.4046
8	0.0001	2.6477	4.8489	10.464	2.2134	1	3.6086	4.561	0
9	0	1.0048	1	2.1828	1	6.8135	2.4007	0.0001	1
10	0	0.0001	4.2513	1	1	1	1	1	7.1836
11	1	1	1	1	1	1	1	1	1
12	0	1	1	5.0206	3.1872	1	1	0.97	1
13	0	1	3.0322	1	1	1	3.5503	1	1
14	0	1	1	4.9609	1.8001	7.4725	1	1	1

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
7	0.48619	0.53591	0.48485	0.46729	36.223
8	0.40331	0.54144	0.42424	0.43925	52.541
9	0.42541	0.44199	0.39394	0.44860	22.667
10	0.39227	0.42541	0.42424	0.51869	22.667
11	0.41436	0.41436	0.36364	0.42991	12.854
12	0.38122	0.40331	0.36364	0.47664	25.288
13	0.38122	0.42541	0.36364	0.43925	26.037
14	0.34807	0.41989	0.33333	0.40187	33.774

Pima-Indians diabetes Dataset #1 with Hooke and Jeeves method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0	11.312	0	17.708	0.895	7.4489	0.6766	7.5813
3	8.9392	8.7412	0	0.0001	1	0.0001	0.0001	5.6293
4	3.1414	9.7649	11.318	4.6477	1	1	0	5.7837
5	7.5274	14.875	6.0108	1	2.8234	3.7837	0	5.9473
6	7.0103	7.3605	9.932	1.5353	1	14.054	0	2.7613
7	19.544	15.309	4.0998	0.0001	1	0	0.0001	12.344
8	0	17.701	0.0001	0.0001	0.9586	5.2765	0.0001	10.01
9	0	14.892	1	1.7516	1.0813	10.608	0.0001	7.5915

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24960	0.34022	0.29496	0.25781	30.763
3	0.24483	0.34499	0.28058	0.25000	29.001
4	0.24960	0.33545	0.25899	0.25651	40.295
5	0.25437	0.33386	0.23741	0.25911	55.458
6	0.24960	0.31161	0.27338	0.25651	42.479
7	0.24324	0.30684	0.25899	0.25781	122.070
8	0.24642	0.29730	0.24460	0.25781	96.112
9	0.25119	0.30207	0.25180	0.25911	110.930

Pima-Indians diabetes Dataset #2 with Hooke and Jeeves method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0001	10.332	0.0001	15.273	1	0.9977	0.9918	0
3	14.049	8.3545	0.9989	6.095	1	7.5627	0	9.1034
4	13.16	8.6739	7.8553	2.2226	0.9404	13.879	0	1
5	0.9262	9.8098	5.7066	5.2204	1.0105	10.582	0	12.453
6	4.584	9.4339	7.4663	4.3731	1	11.52	0	10.826
7	22.175	12.144	1.9704	24.252	1.9704	15.161	1	5.5228
8	0	3.5673	0.4857	6.9448	1	5.1221	0.0001	7.996
9	1.8393	24.09	5.2438	1.8393	1.8393	1.8393	1	2.7279

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24765	0.33072	0.33846	0.26563	31.918
3	0.24451	0.34483	0.23846	0.25260	27.129
4	0.24295	0.33229	0.25385	0.25260	42.619
5	0.24451	0.32132	0.23846	0.24870	90.262
6	0.23824	0.30721	0.23846	0.25000	71.620
7	0.23668	0.31505	0.26923	0.25651	80.091
8	0.23511	0.30408	0.27692	0.25521	81.011
9	0.24765	0.29310	0.25385	0.25391	77.142

Pima-Indians diabetes Dataset #3 with Hooke and Jeeves method

# *k*	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0001	10.332	0.0001	15.273	1	0.9977	0.9918	0
3	14.049	8.3545	0.9989	6.095	1	7.5627	0	9.1034
4	13.16	8.6739	7.8553	2.2226	0.9404	13.879	0	1
5	0.9262	9.8098	5.7066	5.2204	1.0105	10.582	0	12.453
6	4.584	9.4339	7.4663	4.3731	1	11.52	0	10.826
7	22.175	12.144	1.9704	24.252	1.9704	15.161	1	5.5228
8	0	3.5673	0.4857	6.9448	1	5.1221	0.0001	7.996
9	1.8393	24.09	5.2438	1.8393	1.8393	1.8393	1	2.7279

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.24765	0.33072	0.33846	0.26563	31.965
3	0.24451	0.34483	0.23846	0.25260	27.129
4	0.24295	0.33229	0.25385	0.25260	42.713
5	0.24451	0.32132	0.23846	0.24870	90.293
6	0.23824	0.30721	0.23846	0.25000	71.324
7	0.23668	0.31505	0.26923	0.25651	80.434
8	0.23511	0.30408	0.27692	0.25521	81.058
9	0.24765	0.29310	0.25385	0.25391	77.673

Pima-Indians diabetes Dataset #4 with Hooke and Jeeves method

# *k*	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0.0633	5.0134	0	0.0001	0.5076	15.287	0.0001	8.5433
3	0	7.7061	0.0001	1	0.89	0.9907	0.9644	11.424
4	3.1696	10.092	0	7.8915	1	7.5305	0.0001	10.972
5	9.0567	4.0916	4.1726	3.6589	1	9.4132	4.8055	8.764
6	1	4.3815	1	1	1	6.9382	0	0.0001
7	0	10.897	2.6559	2.8861	2.8761	15.671	0.0001	2.6083
8	5.033	7.3305	2.1853	8.4018	1	17.157	3.5265	14.431
9	7.7636	7.5938	1	9.7789	1	13.755	0	1

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.23491	0.34747	0.26452	0.24609	57.190
3	0.25122	0.34910	0.23871	0.25130	27.690
4	0.24633	0.34584	0.25161	0.24870	71.651
5	0.23817	0.30995	0.25161	0.25651	74.085
6	0.24633	0.31158	0.26452	0.25521	41.527
7	0.24633	0.31648	0.27742	0.25130	60.825
8	0.21860	0.30179	0.23871	0.23698	166.050
9	0.22838	0.30832	0.23226	0.26302	58.984

Pima-Indians diabetes Dataset #5 with Hooke and Jeeves method

# k	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age
2	0	8.9167	1.0859	14.722	0.9464	12.322	0.0001	9.1623
3	0	13.803	5.2428	11.889	1.3038	0.9258	0.988	9.8319
4	1.2546	10.275	8.2872	7.7335	1	14.36	1.1146	1.1805
5	6.7903	6.827	1.7963	1	1	12.329	0	0.3267
6	5.9019	12.09	4.3684	1	1	0.9996	0	13.9
7	6.4421	12.252	2.7717	1	1	0.9721	0	1
8	2.042	5.3111	7.7227	9.785	1	6.0737	0	10.851
9	0	15.068	7.2482	1.7555	1	1	0.0001	0.9999

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
2	0.25854	0.35122	0.27451	0.26693	44.460
3	0.25854	0.36098	0.20261	0.25130	51.012
4	0.24553	0.34309	0.19608	0.25260	38.267
5	0.25366	0.30244	0.20915	0.25651	56.301
6	0.25528	0.31382	0.18301	0.25000	79.826
7	0.25854	0.31382	0.18954	0.25781	62.759
8	0.25041	0.29919	0.22222	0.27604	136.440
9	0.24878	0.29756	0.19608	0.25781	59.811

Abalone Dataset #1 with Hooke and Jeeves method

# *k*	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	13.059	19.078	28.079	1.9176	1.3487	1	27.12
5	0	2.1009	19.12	0.6876	0.0001	1.9728	10.971
6	37.461	51.733	28.297	9.8398	8.0071	1	127.65
7	3.3045	17.959	2.0333	1	2.6721	3.452	39.756
8	3.7323	0	5.486	0.252	1	17.137	17.559
9	1.1021	1	10.333	1	8.2707	5.1352	16.089
10	4.8415	5.858	12.951	0	2.8401	0.3648	7.471

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.48286	0.51625	0.50441	0.48791	517.75
5	0.47813	0.52482	0.49937	0.48288	591.17
6	0.47961	0.51212	0.49559	0.48719	656.90
7	0.47872	0.50739	0.50315	0.48408	966.11
8	0.47193	0.51300	0.50315	0.49701	685.75
9	0.46188	0.50768	0.48802	0.47474	1377.80
10	0.45833	0.51152	0.49306	0.47307	945.62

Abalone Dataset #2 with Hooke and Jeeves method

# *k*	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	4.2758	0.0001	29.258	0	0.0001	0.9146	9.7525
5	3.489	12.009	2.8522	0.4732	0	3.5174	20.357
6	0.8589	16.394	1.4652	1	0	0.0001	19.023
7	1	13.72	10.257	1	0	1.9082	14.995
8	27.008	37.109	19.397	1	35.049	1.333	73.95
9	7.2405	1.8966	14.564	1	1.116	1.116	1.116
10	7.4082	1.0877	3.7893	1	1.7132	8.2176	12.356

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.48552	0.51655	0.47667	0.48863	517.75
5	0.48729	0.53221	0.47541	0.48480	591.17
6	0.49202	0.52009	0.47289	0.49390	656.90
7	0.49113	0.51300	0.47289	0.48839	966.11
8	0.47872	0.51743	0.44388	0.48097	685.75
9	0.48907	0.51537	0.48172	0.48839	1377.80
10	0.47754	0.51596	0.45649	0.49437	945.62

Abalone Dataset #3 with Hooke and Jeeves method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	5.925	8.6149	13.7	0.0001	0.0001	0	12.041
5	3.5006	10.883	0.0001	0.3618	0	0.9997	17.042
6	7.3533	7.2209	1.2721	1.7069	1	2.9012	20.343
7	0.9287	8.1773	0.0001	0	0.0001	3.6278	21.085
8	6.4845	5.7469	15.718	1.3598	14.819	1	31.951
9	11.351	11.973	7.517	0	9.0245	7.5946	19.003
10	7.8134	8.844	17.887	1.2273	4.2976	1	8.9629

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49255	0.52443	0.46285	0.48887	361.30
5	0.49225	0.53367	0.44823	0.48456	679.60
6	0.49315	0.52145	0.46285	0.48671	800.53
7	0.48927	0.51579	0.46529	0.48312	428.02
8	0.47765	0.51907	0.46650	0.47570	677.03
9	0.50328	0.51847	0.47016	0.47785	1170.70
10	0.47735	0.51996	0.44336	0.47283	1243.80

Abalone Dataset #4 with Hooke and Jeeves method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	19.818	1.7678	21.503	1	2.7071	1.7676	31.23
5	1	0	16.794	0.0001	1	3.5146	17.055
6	0.6196	2.5847	13.023	0.0007	0.0004	0	15.579
7	16.34	0	9.7004	1.7291	0.9998	2.8645	19.386
8	4.2118	18.318	12.223	1.2218	1	1.0492	25.018
9	0	1.304	0.0001	0.3745	9.7037	6.4624	18.546
10	28.768	3.4657	28.375	1	3.8052	3.8052	24.521

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49035	0.51469	0.49134	0.49102	420.86
5	0.48115	0.52894	0.48144	0.48504	519.95
6	0.48917	0.51410	0.48144	0.48504	758.24
7	0.49124	0.51113	0.49257	0.49318	751.05
8	0.48442	0.51083	0.48639	0.48719	1179.60
9	0.46661	0.51172	0.45545	0.47498	1820.70
10	0.47136	0.51291	0.48515	0.49078	948.41

Abalone Dataset #5 with Hooke and Jeeves method

# k	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4	3.1605	13.783	15.346	0.8669	0	3.5166	16.075
5	5.8246	11.347	20.231	0	0.0001	4.7016	21.344
6	0	5.4619	17.912	0.4737	0.3824	2.8352	14.257
7	2.4614	5.075	58.221	1	15.038	12.267	26.783
8	8.1618	0	0.0001	1.4688	1.0065	0.9985	17.232
9	7.4963	7.3062	0	0.5228	8.7495	9.4012	16.904
10	7.1902	11.495	15.265	14.578	4.7501	0	1

# k	Error				Running Time
	Training	Pre-Training	Test	Whole data	
4	0.49676	0.51942	0.46983	0.48863	515.43
5	0.48558	0.53149	0.47240	0.48384	655.03
6	0.48852	0.51383	0.46727	0.48695	505.32
7	0.47528	0.51001	0.47240	0.50467	975.35
8	0.48764	0.51236	0.47368	0.48935	723.81
9	0.47028	0.51089	0.44544	0.47881	1245.30
10	0.50471	0.51207	0.50449	0.51017	787.23

Ecoli Dataset #1 with Hooke and Jeeves method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	0.6708	0.8238	2.8795	0	1	1	0.7894
9	1.3657	1	1	0	1	1.2766	0.6889
10	1.5182	1	1.3653	1.4271	1.3653	1.179	1.3653
11	3.0664	1.9167	1	5.2342	1.7949	2.1234	1.1684
12	1	0.9629	7.1769	0	1.7343	1	1.0848
13	1.0369	1.032	1.032	4.5661	1.032	1	1.0319
14	1.178	1.178	1	5.0193	1.178	1.3568	1.2173
15	3.2016	1	1	0	0.9587	1	0.5999

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.13718	0.17329	0.41026	0.18750	28.080
9	0.18051	0.19495	0.42308	0.16964	20.904
10	0.15162	0.18051	0.39744	0.16071	28.314
11	0.13718	0.17690	0.39744	0.16369	40.186
12	0.15162	0.19856	0.42308	0.17262	38.595
13	0.12635	0.15523	0.37179	0.17857	33.790
14	0.11191	0.13718	0.39744	0.19048	51.699
15	0.15162	0.17690	0.41026	0.17560	55.474

Ecoli Dataset #2 with Hooke and Jeeves method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1.468	1.0928	1	4.1087	1.0997	1.1934	1.3186
9	2.2337	1.0889	6.318	0	1.7079	1	1.738
10	0.9963	0.7353	0.5491	0.0001	1.2759	1.6497	0
11	1.3758	0.5256	1	0	1.3423	1.1518	0.6572
12	2.2636	1.159	1	3.8914	1.0664	1.2084	1.2644
13	1.1963	1.0976	3.7448	8.4239	1.7551	1.0976	1
14	1.6447	1.3242	1.8016	6.1208	1.7909	1	1.8013
15	322.09	209.83	123.82	1	239.61	201.02	239.61

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.16245	0.22744	0.39744	0.16667	29.094
9	0.16606	0.22022	0.39744	0.20833	39.468
10	0.14440	0.20939	0.43590	0.18155	62.478
11	0.15884	0.17690	0.42308	0.17262	40.966
12	0.13718	0.17690	0.39744	0.16071	39.765
13	0.13357	0.20217	0.38462	0.15476	61.402
14	0.10469	0.12996	0.38462	0.14583	28.174
15	0.11552	0.13357	0.38462	0.15179	39.796

Ecoli Dataset #3 with Hooke and Jeeves method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1	1.22	9.4084	0	0.9421	0.9995	1
9	1.5913	1	0	0.0001	1.0206	1	0.5753
10	2.2351	1	0.4827	0.0001	0	1.716	0.9998
11	0.6102	1.008	7.6186	0	1.0878	0.8723	0.8377
12	2.4797	1.8585	1	0	1	1	1.177
13	0.9363	1.1741	4.9175	0	1.6857	1	1
14	1	1.3079	1.0291	1.3079	1.3079	1.3079	1.3079
15	1.9447	1.8338	1	1.8338	1.8331	1.8338	1.8338

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.14440	0.21661	0.32468	0.22024	17.035
9	0.13718	0.18773	0.29870	0.16369	41.933
10	0.17329	0.20939	0.29870	0.20536	40.498
11	0.14801	0.16606	0.27273	0.17560	41.496
12	0.16968	0.18773	0.28571	0.18452	25.615
13	0.12635	0.15162	0.29870	0.17560	58.485
14	0.13718	0.15884	0.31169	0.18155	26.723
15	0.12996	0.14440	0.29870	0.12798	28.283

Ecoli Dataset #4 with Hooke and Jeeves method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1	2.2906	10.195	26.379	2.2906	2.2906	2.2906
9	1.3295	2.0757	9.7879	0	1.3708	1	1.3368
10	1	0.9484	6.5982	0	1.026	1	1.1579
11	1	1.0914	0.5948	0.00014	0.84777	1	0.9064
12	4.8698	8.2115	4.6992	1	8.2115	8.2115	8.2115
13	0.7257	0.2923	1	0	0.9389	0.8102	0.7578
14	2.4338	1	1.3765	4.8448	1.9428	1.6061	1.641
15	2.505	1.8566	3.9824	10.906	2.992	1.8566	1

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.11913	0.18051	0.34615	0.20536	16.817
9	0.13357	0.19495	0.26923	0.18452	39.780
10	0.11552	0.15162	0.35897	0.16964	33.852
11	0.13357	0.14079	0.32051	0.15774	15.975
12	0.10108	0.12635	0.26923	0.16071	21.013
13	0.12635	0.18051	0.26923	0.17857	52.775
14	0.11913	0.21300	0.23077	0.13988	54.117
15	0.11913	0.16606	0.25641	0.13690	38.610

Ecoli Dataset #5 with Hooke and Jeeves method

# *k*	Mcg	Gvh	Lip	Chg	Acc	Alm1	Alm2
8	1.7933	1	1.4849	1.5419	1.1931	1.2059	1.5419
9	0.534	1	1	0	1	1	1
10	1.235	1	0.9995	0	1	1	1
11	2.0635	1.1114	1	3.3212	1	1	1
12	1.0668	1.0664	1.0664	4.0594	1	1.0664	1.0664
13	1.213	0.925	3.9604	0	1.2651	1	0.7563
14	1	1	1	2.6957	1	1.7917	1.0047
15	2.4585	1	1.0808	5.7633	1.2962	1.8634	1.0808

# *k*	Error				Running Time
	Training	Pre-Training	Test	Whole data	
8	0.14079	0.20578	0.38462	0.19643	16.942
9	0.17690	0.18051	0.34615	0.16071	23.712
10	0.15523	0.15884	0.35897	0.15476	17.254
11	0.12274	0.15523	0.35897	0.17560	37.908
12	0.12274	0.14440	0.34615	0.13690	27.565
13	0.12635	0.13718	0.35897	0.16964	25.475
14	0.10830	0.12274	0.34615	0.15476	28.969
15	0.10830	0.18051	0.34615	0.13690	44.491

VITAE

NAME : Mr. Wacharasak Siriseriwan

EDUCATION

1988 - 1994 : Bhadungsit pittaya School, Bangkok

1995 - 2000 : Samsen Wittayalai School, Bangkok

2001 – 2004 : Department of Mathemetic, Faculty of Science, Chulalongkorn University, GPAX 3.34 / 4.00

2005 – 2008 : Department of Mathemetic, majoring computational science, Faculty of Science, Chulalongkorn University. GPAX 3.61 / 4.00