ITERATIVE CLUSTERING USING HIERARCHICAL TREE FOR HIGH-DIMENSIONAL GENOTYPIC DATA

CHAINARONG AMORNBUNCHORNVEJ

A THESIS SUBMITTED IN FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN TELECOMMUNICATIONS ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2013

KMITL-2013-EN-M-010-019

หัวข้อวิทยานิพนธ์	การแบ่งกลุ่มข้อมูลแบบวนซ้ำโดยใช้ต้นไม้ลำดับชั้นสำหรับข้อมูลจีโนไทป์หลายมิติ
นักศึกษา	นายชัยณรงค์ อมรบัญชรเวช
รหัสประจำตัว	54611701
ปริญญา	วิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
พ.ศ.	2556
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ดร.ตุลยา ลิมปิติ

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ นำเสนอกรอบการวิเคราะห์สำหรับการแบ่งกลุ่มข้อมูลจีโนไทป์หลายมิติ ซึ่งมีรูปทรงของกลุ่มข้อมูลที่ไม่แน่นอนและมีขนาดใหญ่ โดยนำเอ าแผนภูมิต้นไม้ลำดับชั้นชนิดต่างๆ มาใช้ในการแก้ปัญหาการแบ่งกลุ่มข้อมูลในการศึกษาโครงสร้างประชากร ที่สามารถให้ผลลัพธ์เป็น จำนวนกลุ่มข้อมูล การจำแนกประชากรเข้ากลุ่มข้อมูล และ การบ่งบอกความสัมพันธ์ระหว่างกลุ่มข้อมูล ได้ในคราวเดียวกัน งานวิจัยส่วนแรกเป็นการปรับปรุงวิธีการวิเคราะห์พรินซิเพิลคอมโพเนนต์แบบวนซ้ำ เพื่อให้การแบ่งกลุ่มข้อมูลมีความแม่นยำขึ้น โดยการใช้การแบ่ง แผนภูมิต้นไม้ลำดับชั้นที่สร้างขึ้นด้วยเงื่อนไขความแปรปรวนที่น้อยที่สุดของวาร์ด จากนั้นได้ศึกษาผลกระทบของการลดขนาดข้อมูลโดยการเลือกคุณลักษณะเด่นต่อประสิทธิภาพการแบ่งกลุ่มข้อมูล สำหรับงานวิจัยส่วนที่สองเป็นการพัฒนาวิธีการแบ่งข้อมูลแบบวนซ้ำแบบใหม่โดยใช้เนย์เบอร์จอยนิงทรี ซึ่งเป็น แผนภูมิต้นไม้ลำดับชั้นสำหรับระบุความสัมพันธ์ทางพันธุกรรมระหว่างประชากร พร้อมทั้งได้นำเสนอมาตรวัดความบริสุทธิ์ของกลุ่มประชากรสองแบบเพื่อใช้เป็นเงื่อนไขหยุดการทำงานของอัลกอริธึมแบบวนซ้ำ คือ การวัดความบริสุทธิ์โดยการตรวจจับรูปแบบโทโพโลยีของเนย์เบอร์จอยนิงทรี และการวัดความบริสุทธิ์โดยการพิจารณาอัตราการเปลี่ยนแปลงของค่าดัชนีฟิกเชชัน อัลกอริธึมที่ทำการพัฒนาขึ้นใหม่ทั้งสองวิธีได้ถูกทดสอบประสิทธิภาพการทำงานด้วยโครงสร้างของกลุ่มประชากรข้อมูลจำลองและข้อมูลจริงจาก วัว แกะ และ มนุษย์ พบว่าผลลัพธ์การอนุมานจำนวนกลุ่มของประชากร และการจำแนกประชากรเข้ากลุ่มข้อมูลมีความถูกต้องแม่นยำมากขึ้น และมีความสอดคล้องกับความสัมพันธ์ทางชีวภาพที่แท้จริง

Thesis	Iterative Clustering using Hierarchical Tree for High-dimensional Genotypic Data
Student	Mr.Chainarong Amornbunchornvej
Student ID	54611701
Degree	Master of Engineering
Program	Telecommunications Engineering
Year	2013
Thesis Advisor	Dr. Tulaya Limpiti

ABSTRACT

This thesis proposes a framework for analyzing large, high-dimensional genotypic data which has irregular cluster pattern. Different types of hierarchical trees are exploited for solving a clustering problem in population structure analysis. The algorithms provide an estimated number of clusters, individual assignments, and relationship among individuals simultaneously as outputs. The first part of the thesis improves the individual assignment accuracy of an existing iterative principal component analysis method using Ward's minimum variance hierarchical tree as its clustering module. Moreover, the effect of dimensional reduction by way of an informative feature selection process has been investigated. The later part of this thesis develops a new iterative algorithm for data clustering using neighbor-joining tree, which is a hierarchical tree used to explain genetic relationships among subpopulations. Two cluster homogeneity measures have been proposed. One measure is based on the topological pattern of the neighbor-joining tree, where as the other measures the difference of the fixation index values at successive iterations. The performance of both algorithms in analyzing population structures are tested using simulated datasets and real datasets from bovine, sheep, and human populations. The result illustrates good clustering performance in terms of the estimated number of clusters and individual assignments, and is agreeable to the intrinsic genetic relationship.

ACKNOWLEDGEMENTS

The thesis has completely finished because of the contributions from many people. I would like to take this occasion to express my appreciation toward all of them.

First, I appreciate my advisor, Dr. Tulaya Limpiti, who always helps me solve problems, encourages me to overcome many tough obstacles, and motivates the process of this research. I would like to thank Dr.Anunchai Assawamakin, Dr. Apichart Intarapanich, and Dr.Sissades Tongsima for technical supports, encouragements and everything else they have done for me during this period.

I would like to acknowledge the National Center for Genetic Engineering and Biotechnology for technical and data supports. Special thanks go to my colleagues in the department of telecommunications engineering, King Mongkut's institute of technology Ladkrabang for their helps in all aspects.

I am grateful to the National Science and Technology Development Agency (NSTDA) for funding supports through research grants JSTP-06-54-07E. This work is also supported in part by NSTDA under grant SCH-NR2010-22-05.

Finally, this thesis is dedicated to everyone in my life, who always supports and encourages me.

Chainarong Amornbunchornvej

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Cont.)

Figure Page

LIST OF TABLES

Table Page

Chapter 1
Introduction

1.1 Motivation

At present, information technology has been applied to many research problems. In biology and medicine, there are many applications adopting computer technology in the fields called bioinformatics, which consolidates theory of computation, information technology, biology, and medicine in order to solve the problems such as finding the causes of genetic disease, identifying a person from forensic science, retina scanning in security systems, and so on.

One data analysis tool for bioinformatics that is widely adopted is data clustering. Data clustering has been used to discover life ancestry, find patterns of any organisms, speculate disease factors of organisms on gene pool, to name a few. However, using data clustering on genetic data is challenging, for clusters of genotypic data have complex structure and arbitrary shape. These reasons motivate us to investigate how to identify complex structure and obscure shape of genotypic clusters by developing new methods of graph-based data clustering. According to graph theory, there is an important advantage for using a graph on genotypic data clustering. The graph converts the clustering problem to graph partitioning problems [1-2], which does not concern about the shape of the clusters. That means using graph for clustering can cope with clusters of arbitrary shapes. Hence, the results of graph clustering should be better than the traditional model-based clustering method.

The most widely used graph-based clustering approach is the hierarchical clustering [3], which is more accurate than other methods [4]. Even if it requires large computational cost, the method is favorable because accuracy is the most important factor in genotypic data clustering discovery.

Therefore, we adopt a hierarchical clustering method for improving one of the high-dimensional genotypic data clustering framework called *ipPCA* [5-6], which have many advantages but suffers from individual assignment errors. In addition, we utilize a neighbor-joining tree [7-8], which is one of the hierarchical trees that is dedicated for representing relationships among individuals, but has not been used directly for clustering. The proposed algorithm is called *iNJclust*. It is desirable to develop a framework that can come up with an estimated number of clusters, individual assignments, and relationships among clusters simultaneously and efficiently. Because the framework is to be used with biological models, the results should correspond to real biological profiles of populations.

The unknown parameters related to the genotypic data clustering problems can be classified into four parts as follows.

1) Number of clusters

In data clustering, we want to know how many clusters there are within a given dataset. We may categorize the clustering algorithms into two types. For supervised clustering algorithms, a number of clusters must be known before hand. In contrast, unsupervised clustering algorithms are able to estimate a number of clusters from a model or some conditional criteria. In this research, we focus our study on the class of unsupervised clustering algorithms.

2) Individual assignments

After the number of clusters is known, the next step of data clustering is to assign an individual into an appropriate cluster. Usually the individual assignment depends on rigorous models or criteria which cause mistakes when the method encounters an arbitrary shape cluster that does not follow the correct model. In contrast, the graph clustering method is able to handle the arbitrary shape problem; it is also highly flexible for solving the complex structure of data.

3) Features of each cluster

An interesting aspect of data clustering is to identify the prominent features of each cluster. It is an important factor that can answer many problems. For

example, if some genes found can distinguish a disease cluster from a normal cluster, the cause of that disease will be discovered. Obviously, there are many benefits if we can find the cluster features.

4) Relationships among clusters

The evolution study relies on the knowledge of relationships among clusters. It may be used to infer ancestry of each individual, immigration patterns of animal, or history of human settlement. The relationships in data clustering is proportional to the distances among clusters

1.2 Objective

The objective of this research is to develop efficient data clustering methods which can handle large, complicated genotype datasets using graph theory. More specifically, this research is divided into two parts:

1) To address the issue of large individual assignment error found in the Iterative pruning principal component analysis (ipPCA) framework [5-6]. The hypothesis is that the fuzzy c-mean algorithm uses as a clustering module within the ipPCA framework suffers from an arbitrary shape cluster. Thus, using a non model-based approach such as graph-based clustering should result in an algorithm with better individual assignments.

2) To develop a novel graph-based genotypic data clustering framework which, besides giving an estimate of cluster numbers and provide proper individual assignments, can also infer relationships among clusters.

1.3 Scope of the Research Work

The theme of this thesis is applying hierarchical tree clustering techniques [3] to cluster populations within genotypic datasets. In the first part, we adopt Ward's hierarchical clustering [9], with the ipPCA framework to improve its clustering results. We also investigate the use of a feature selection technique originally proposed in [10] to reduce data dimension and how it may affect the clustering performance. In the second part, we develop a new iterative clustering framework called iNJclust that uses a well-known phylogenetic tree called neighbor-joining tree to infer relationships among individuals. Moreover, we also develop two new measures for homogeneity determination of populations based on topology pattern and F-statistic, and provide its mathematical proof.

The algorithms for both parts of the thesis are developed in C++ programming language. Their inputs are the single nucleotide polymorphisms (SNPs) sequences. The common outputs of the algorithms are an estimated number of clusters within the dataset and individual assignments. For the iNJclust framework, relationships among populations are also available in the Newick format, which can be visualized using Dendrogram.

We use two simulated datasets and four real genotypic datasets from bovine, sheep, and human datasets to test the efficacy of our algorithms.

1.4 Outline of the Thesis

The thesis is divided into six chapters. The first chapter introduces the motivation, objective, and the scope of this thesis. In chapter 2, we provide some necessary background and theory regarding genotypic data clustering and related topics in order to pave the way for the next chapters.

Chapter 3 introduces the ipPCA framework, its advantages and disadvantages, system architectures of the improved algorithm using hierarchical clustering, and how the dimensional reduction is achieved using feature selection.

Chapter 4 is allocated for the details of the iNJclust framework. It contains related research literature, overall system process, two new measures for cluster homogeneity used as the stopping criterion of the iterative process, and the mathematical proof for a homogeneity measure based on a behavior of the fixation indices.

In chapter 5, we demonstrate the clustering performance of the two algorithms using both simulated and real datasets. We present our findings of the effects of using feature selection to reduce data size and also investigate the effects of evolution times and number of clusters on the value of the proposed homogeneity measure. The chapter ends with some insightful discussions.

The last chapter is the conclusion of this thesis, its contribution to the research community, and future works.

Chapter 2
Background

2.1 Genotypic data clustering

2.1.1 Genotypic data

DNA or Deoxyribonucleic acid is a nucleic acid with double helix structure made up from a sequence of four nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). Different portions of these nucleotides, called genes, represent genetic information of the organism. Different genes are responsible for different instructions and genetic traits for the body and are packed in a chromosome. The same species always has a same number of chromosomes. Usually, parents' chromosomes which contain ancestry information will be inherited to a descendant in the next generation. The concept of DNA had been introduced by Johann Friedrich Miecher in 1869 but its structure was not found until 1953. James D. Watson and Francis Crick constructed DNA structure model which gave them the Nobel Prize award on Physiology or Medicine in 1962.

Single-nucleotide polymorphism or SNP (pronounced *snip*) is a variety of nucleotides in a DNA sequence which occurs from a single nucleotide — A, T, C or G in a position, or a *locus*, being different between individuals in the same species or subpopulations. For example, suppose we have two DNA sequences AACTAGC and AATTAGC, we said there are two *alleles* of the SNP in this locus. Usually, most SNPs have only two alleles. The variation which occurs more in nature is called the *major allele*, and the variation with less occurrence is the *minor allele*. Figure 2.1 depicts a DNA fragment. The SNPs can be seen in Fig. 2.2.

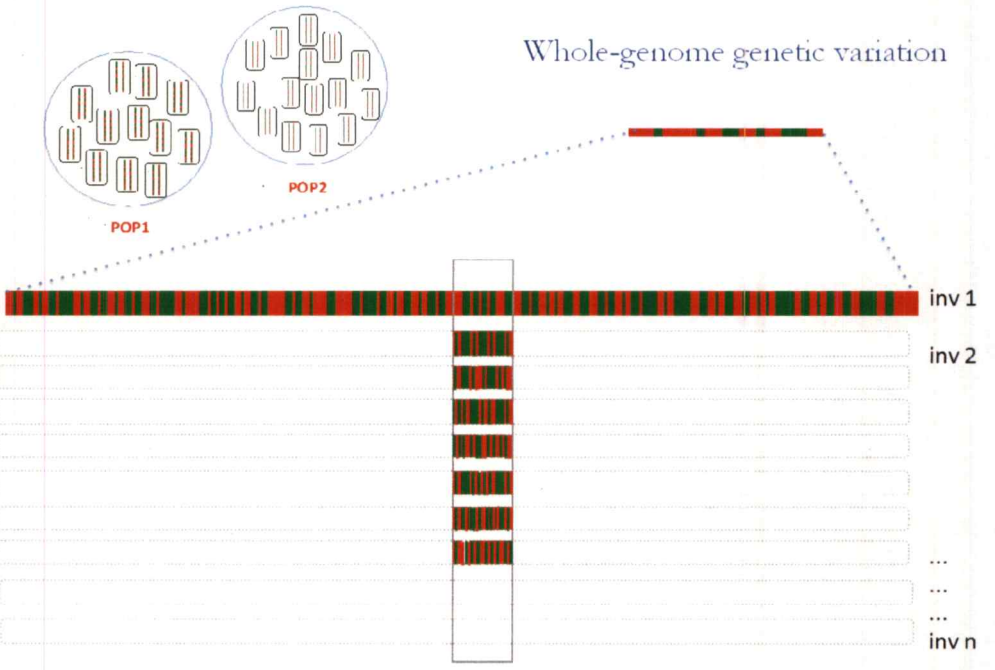


Figure 2.1 Fragment of DNA sequence in subpopulations. Red color represents the SNPs; green color represents other components of the DNA.

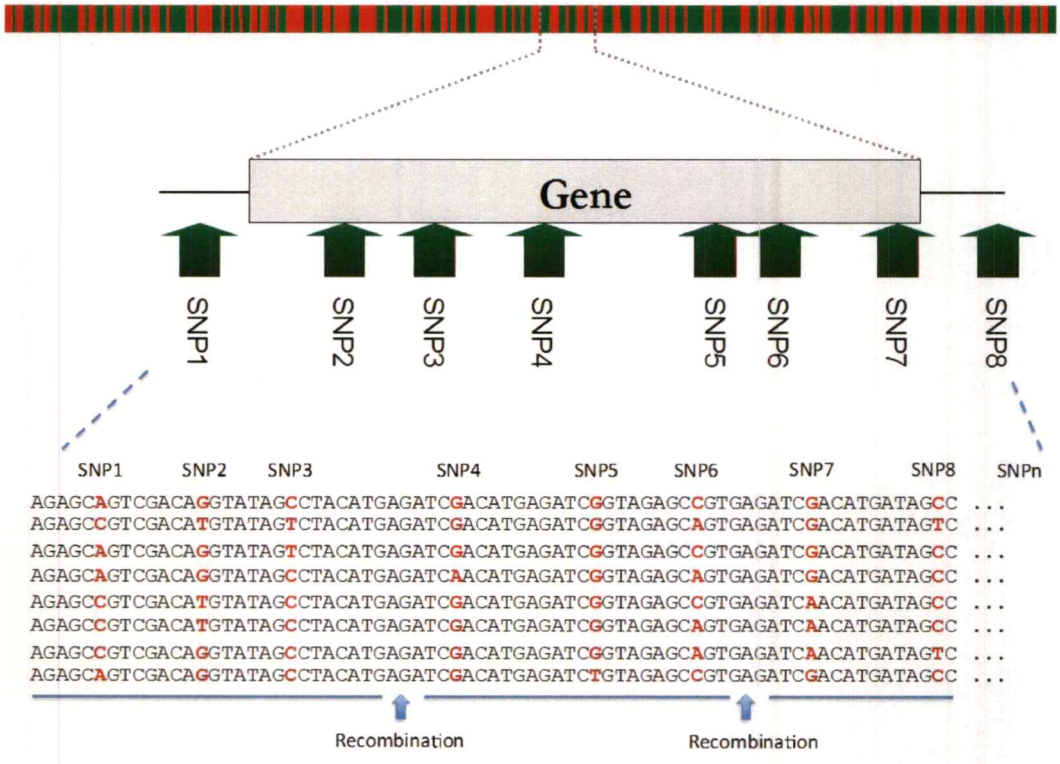


Figure 2.2 SNPs on many positions of DNA sequences, shown in red.

According to natural selection theory, SNPs is an adaptation of genetic [11]. Recombination rate and mutation rate can be used to determine a density of SNPs in a population. This variation of DNA sequence in human genetic corresponds to disease, chemical reaction, vaccine reaction, and so on, on a human body. Hence, SNPS contain vital information for human treatment [12].

The SNPs can be categorized into three types: The method for determining types of SNPs can be defined by counting frequencies of nucleotides that occur in each column independently. For example, there are 'A' and 'T' nucleotides in column i. If we found that 'A' more occur frequently than 'T', then we designate 'A' as a dominant type and 'T' as a recessive type. Therefore, we can assign 'AA' in column i as a *Wild* SNP, which is denoted numerically as a zero. 'AT' or 'TA' are called Normal Type and are denoted by one. And 'TT' is a *mutant type* denoted by **two**. Similarly, in the other column of SNPs, if 'G' occurs more frequently than 'C', 'G' can be determined as a dominant type and 'C' as a recessive type. We define a SNPs matrix as $X_{M\times L}$, which consists of M individuals and L SNPs. It is accepted that for different populations the SNP sequences are different. For the same population, the allele frequency is supposed to be the same. In this research we cluster data using the SNPs of populations in dataset. Before clustering, we convert the SNPs sequence into a numerical vector (Fig. 2.3). The missing data portion will be assigned as minus one.

2.1.2 Classes of clustering algorithm

We can categorize clustering algorithms into two classes as follows:

1. Supervised clustering, sometimes called parametric method, is a class of clustering algorithms in which the number of cluster, n, must be known. Examples of algorithms in this class are K-means algorithm [13], Fuzzy c-means algorithm [14], EM clustering [15], self-organized mapping [16].

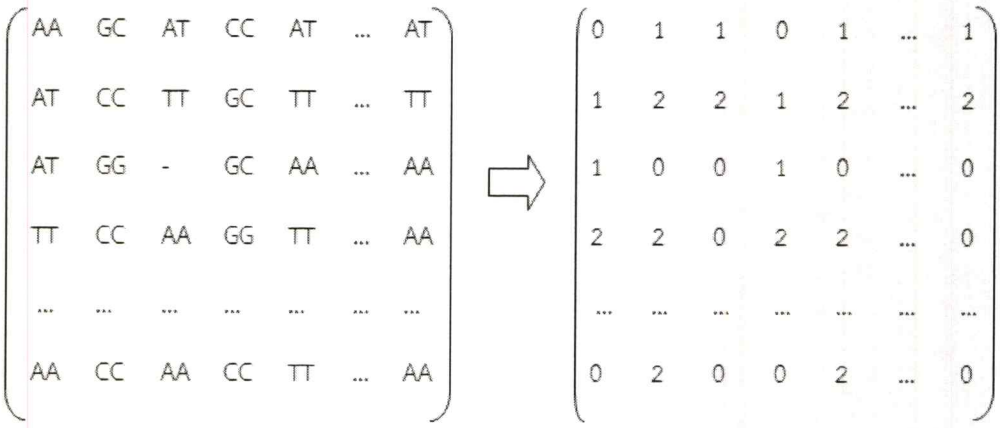


Figure 2.3 Numerical encoding of SNPs sequences.

2. Unsupervised clustering or non-parametric method is a clustering approach that is able to estimate a cluster number using some criteria. It is more useful than the supervised clustering algorithms because in real circumstances we do not know the cluster number in a dataset. The algorithms in this type are for example AWclust [17] and ipPCA [5].

Alternatively, we can classify clustering algorithms using a manner of algorithm results, as shown on Fig. 2.4 [4].

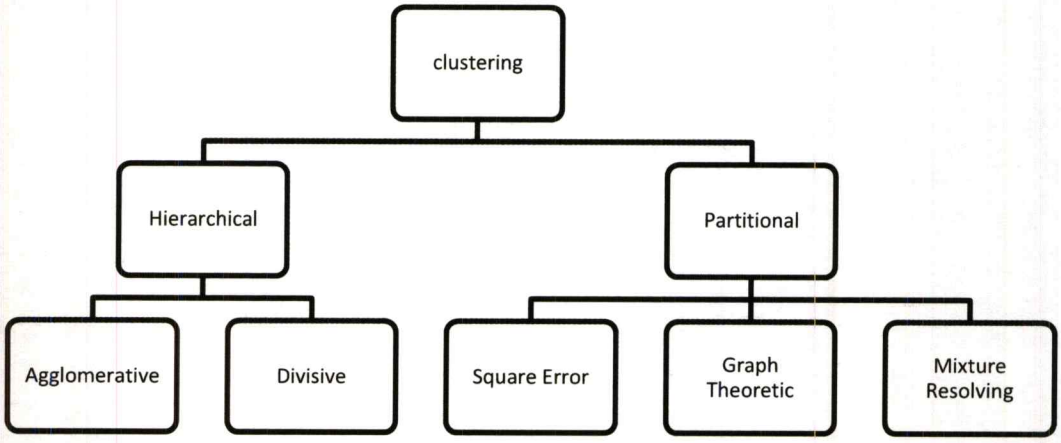


Figure 2.4 Types of clustering approaches (modified from [4]).

(i) Hierarchical clustering is a clustering method which produces a series of clusters. Individual assignment results are reported in the form of the final hierarchical structure. The benefit of this approach is that a user can choose a number of clusters by cutting a hierarchical tree at the desired level of clusters. Hierarchical clustering is a graph clustering method. It inherits an advantage of graph clustering, so it is capable of clustering on arbitrary shape effectively. Since the method is able to determine multilevel of clusters, we can exploit more information for clustering than other graph-theoretic [18] approaches. However, this method consumes a lot of computational cost to process since it is an NP-Hard problem [18]. The algorithms in this class are single link, complete link, and so on.

(ii) Partitional clustering is a clustering approach which produces only one result. Any results of individual assignment are reported on a set of clusters. The advantage of this approach is low computational cost (lower than hierarchical clustering) in a large dataset. The method attempts to divide a dataset into partitions by optimizing some criterion, e.g. square error, likelihood function, and so on. The algorithms in the category are K-means, Fuzzy c-means, expectation-maximization, and so on.

2.1.3 Data clustering process

Process for genotypic data clustering can be summarized in Fig. 2.5. Each individual is input to the system as a numerical sequence. The first step of data clustering is data pre-processing. It attempts to prepare raw data into appropriate form before data clustering process, e.g. data normalization, feature selection, noise filtering, etc. After data pre-processing have been done, the data clustering module will be activated. In traditional clustering, only the number of clusters and individuals assignment will be reported.

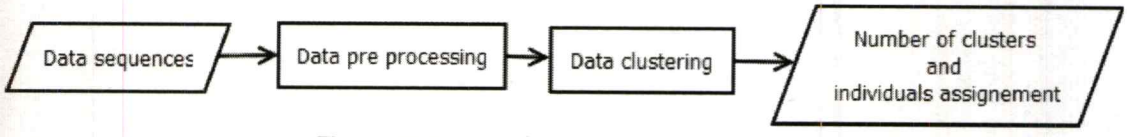


Figure 2.5 Data clustering process

2.2 Principal component analysis (PCA)

After we encoded SNP sequences to numerical vectors, the next step is to convert the vectors to a new domain to increases the clustering performance. The most widely used method is principal component analysis (PCA), which is proposed by Karl Pearson in 1901 [19]. PCA projects the data points into a subspace according to their similarities. The numerical vectors projected onto a principal component vectors that are more tend to be close. Figure 2.6 demonstrates this technique, where only the first two principal components are shown. PCA is found to be efficient for increasing the clustering performance.

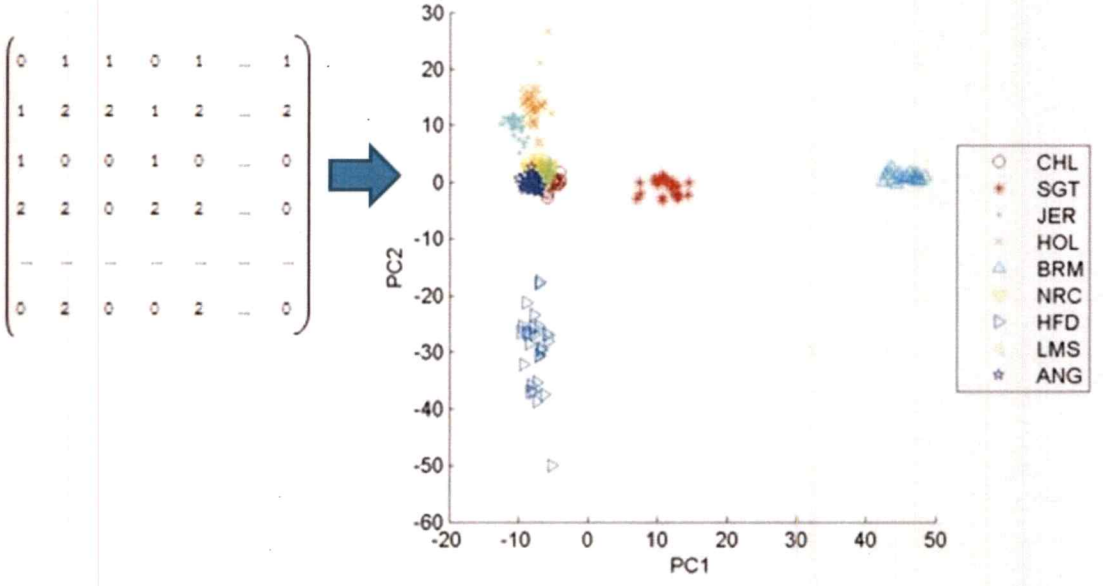


Figure 2.6 Projection of data points onto the first two principal components.

Suppose data matrix X has M individuals by L SNP dimensions. Singular value decomposition (SVD) of X is employed for calculating the principal components,

$$X_{M\times L} = USV^T \qquad (2.1).$$

Where U is a covariance matrix of the rows of X, S is a diagonal matrix containing the eigenvalues of X, and V is a covariance matrix of X's columns. Sometime, L can be tremendously larger than M, preventing us to compute (2.1) directly. In this case we compute the principal components using

$$\left(XX^T\right)_{M\times L} = US^2U^T \qquad (2.2).$$

Suppose $r = rank(XX^T)$ we can delete the zero elements to reduce the dimensions of S to S' which is $r\times r$. We can reduce U^T's dimension from $M\times M$ to $r\times M$. Then the result is U'^T.

$$V'^T_{r\times L} = S'^{-1}U'^T X \qquad (2.3)$$

$$Y_{M\times r} = XV' \qquad (2.4)$$

Finally, we use Y for data clustering in the next step. In fact, Y contains M individuals with r features that clustering method can group data by determining distances among individuals from their features.

2.3 Similarity measurements on genotypic data

2.3.1 Euclidean distance

This method is widely used for measuring distance between a pair of vectors [4].

$$d(\vec{u},\vec{v}) = \sqrt{\sum_{i=1}^{L}(u_i - v_i)^2} \qquad (2.5)$$

Where $\vec{u}$ and $\vec{v}$ are L-dimensional vectors.

2.3.2 Allele-sharing distance

The method is an effort to use the Manhattan distance for measuring distance between SNP sequences by counting the number of allele dissimilarity between the two SNP sequences.

$$d(\vec{u},\vec{v}) = \frac{1}{L}\sum_{i=1}^{L}|u_i - v_i| \tag{2.6}$$

In bioinformatics, a dissimilarity matrix calculated from the allele sharing distances is called an "Allele sharing distance matrix" or an ASD matrix [20].

2.3.3 F-statistic

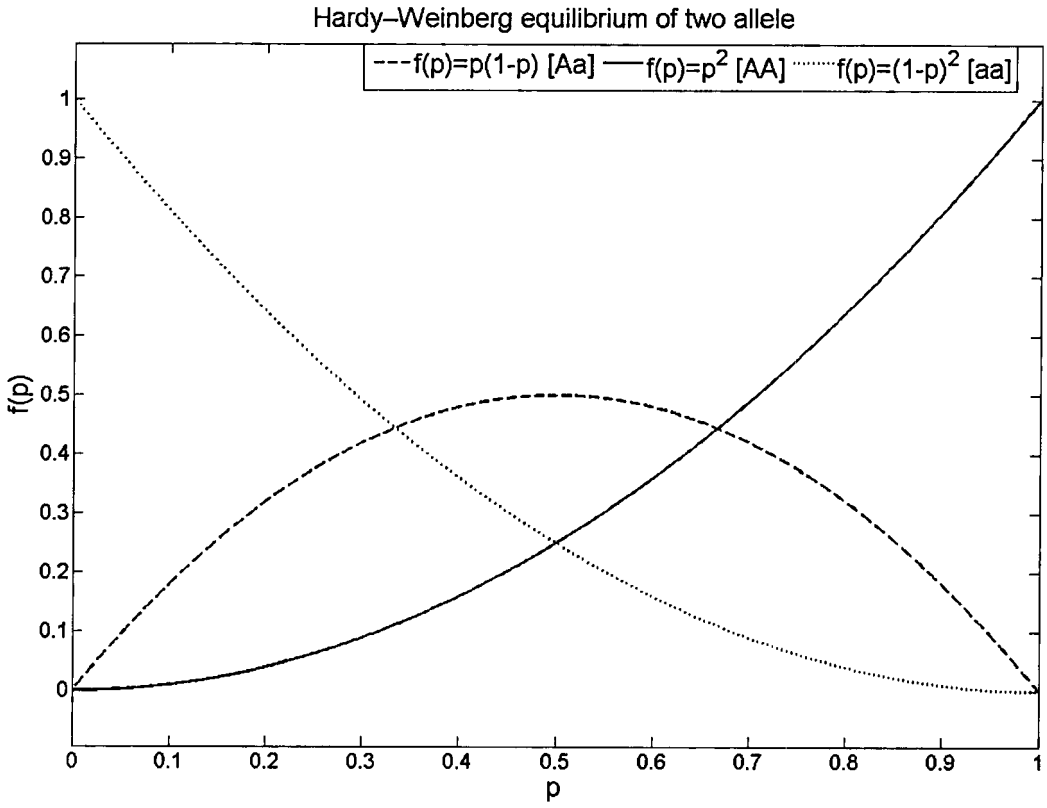


Figure 2.7 Hardy–Weinberg equilibrium of two allele which relates allele frequency (horizontal axis) to genotype frequency (vertical axis).

According to population genetics, F-statistic [21] is adopted for statistic computation regarding heterozygosity of clusters. This approach relies on the Hardy–Weinberg principle [22]. The Hardy–Weinberg principle on Fig. 2.7 illustrates the relationship between allele frequency and genotype frequency. The solid line

represents homogeneous wild type allele, the dashed line is normal type, and the dotted line is mutant type. According to Fig. 2.7, if we found a lot of wild types, then we found fewer mutant type, and vice versa. The factor used to measure a genetic distance related to the ratio of subpopulations and total individuals is called a "Fixation index" or F_{ST}, [23]. To calculate the value of F_{ST}, we have to determine the related factors as follows. Let p be an allele frequency of dominant gene or the probability of finding the dominant gene; q is an allele frequency of the recessive gene or the probability of finding the recessive gene, and m is a number of individuals in a population. The p and q can be calculated using the following equations:

$$p = \frac{2 \times Count(WildTypeSNPs) + Count(NormalTypeSNPs)}{m} \quad (2.7),$$

$$q = 1 - p \quad (2.8).$$

The next step is to calculate local observed heterozygosity (probability of normal type SNPs from observation).

$$H_{obs} = \frac{Count(NormalTypeSNPs)}{m} \quad (2.9)$$

Then, we calculate the local expected heterozygosity, or gene diversity, of each subpopulation.

$$H_{\exp} = 1 - (p^2 + q^2) \quad (2.10)$$

Let $\overline{p}$ be an average dominant gene allele frequency of the entire subpopulations (global probability of finding dominant gene on the whole dataset) and $\overline{q}$ is an average recessive gene allele frequency of the entire subpopulations (global probability of finding recessive gene on the whole dataset).

$$\overline{p} = \frac{\sum_{i=1}^{n}(p_i m_i)}{\sum_{i=1}^{n} m_i} \tag{2.11}$$

$$\overline{q} = 1 - \overline{p} \tag{2.12}$$

where n is a number of clusters, m_i is a number of members in the i^{th} cluster, and p_i is a dominant allele frequency of the i^{th} cluster. Actually, for a SNPs matrix $X_{M\times L}$ consists of M individuals and L SNPs, we have $\sum_{i=1}^{n} m_i = M$. The next step is to calculate the global heterozygosities indices (over Individuals, Subpopulations and Total population).

$$H_I = \frac{\sum_{i=1}^{n}(H_{obs})_i \times m_i}{\sum_{i=1}^{n} m_i} \tag{2.13}$$

$$H_S = \frac{\sum_{i=1}^{n}(H_{\exp})_i \times m_i}{\sum_{i=1}^{n} m_i} \tag{2.14}$$

$$H_T = 2 \times \overline{pq} \tag{2.15}$$

H_I is based on observed heterozygosities in M individuals in n subpopulations, H_S is based on expected heterozygosities in subpopulations, and H_T is based on expected heterozygosities for the total population.

Finally,

$$F_{IS} = \frac{H_S - H_I}{H_S} \tag{2.16}$$

$$F_{ST} = \frac{H_T - H_S}{H_T} \tag{2.17}$$

$$F_{IT} = \frac{H_T - H_I}{H_T} \tag{2.18}$$

F_{IS} is a correlation between each individual and subpopulations, F_{ST} is a correlation between subpopulations and all individuals, and F_{IT} relates each individual to all other individuals. For $X_{M\times L}$ SNPs matrix which consists of M individuals and L columns of SNPs, we will have F_{ST} for each SNPs column. It means that we have L F_{ST} values for $X_{M\times L}$. In this thesis, we use an average value of F_{ST} for determining homogeneity of cluster in chapter 4.

2.4 Graph theory for data clustering

2.4.1 Basic concept of graph and tree

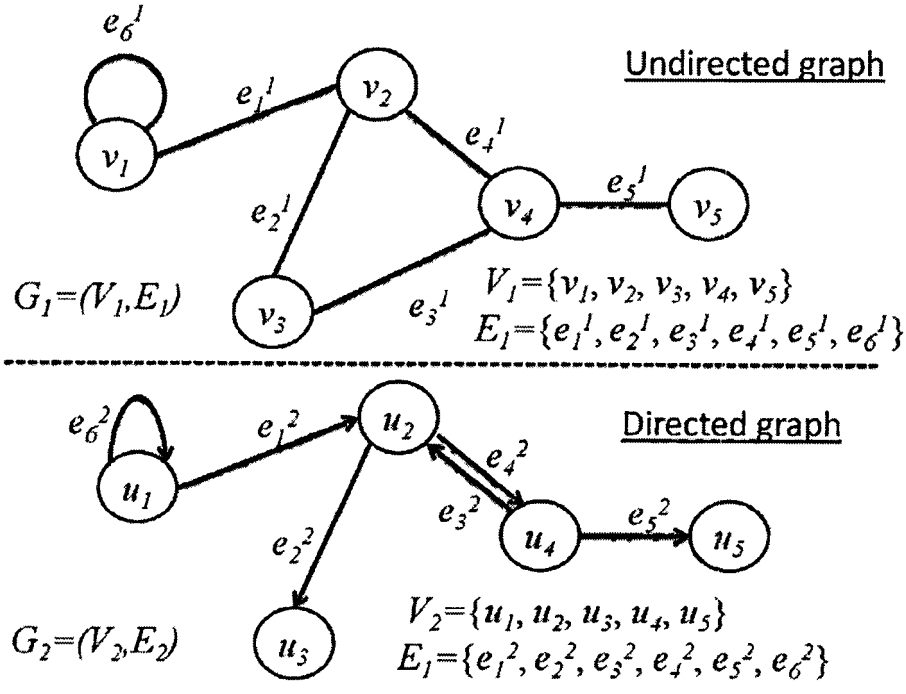


Figure 2.8 Example of undirected graph and directed graph.

- A **Graph** G is a pair of sets $G(V,E)$ where $V=\{v_1,...,v_n\}$ is a set of vertices or **nodes** and $E=\{e_1,...,e_m\}$ is a set of **edges**, representing relationships, distances or paths between pairs of nodes. Two nodes that are connected to the same edge are called **endpoints** of that edge. In graph theory, there are two kinds of graph classified by direction on edges: **directed** and **undirected** graph. In directed graph, each edge of graph has a single direction and can traverse only one way. On the other hand, there is no direction in undirected graph. Hence, we can traverse between two nodes that are connected together in all directions. There are some graphs that have weights on their edges and others with no weight. The weight of a graph can be real number R.

For example, in Fig. 2.8, G_1 is an undirected graph, so all of its nodes, such as v_1 and v_2 can traverse two directions on only one edge: e_1^1, while G_2 is a directed graph, so nodes, such as v_1 and v_2 can traverse only one direction from v_1 to v_2 using edge e_1^2. A **loop of graph** is an edge which is connected at the same node, such as e_6^1 on graph G_1. When two nodes are linked together by the same edge, it is called an **adjacent node**. Sometimes, there are many edges that have the same endpoints, called **multiple edges**.

A **simple graph** is a graph which has no loop and no multiple edges; otherwise it is called a **complex graph**. A **path** is a simple graph which uses a set of nodes to traverse from a begin node to an end node. If we can traverse from v_1 to v_5, then we have a path v_1,v_5-path. A **cycle** is a path which begins and ends on the same node, such as the v_2 cycle in Fig 2.8. We can traverse from v_2 to v_3 and then from v_3 to v_4 and come back to v_2. A **subgraph** $G_3(V_3,E_3)$ is a subset of graph G_1 if and only if a set of nodes $V_3 \subseteq V_1$ and a set of edges $E_3 \subseteq E_1$. A **walk** is a list of nodes and edges used to traverse from a begin node to an end node. For example, the walk from u_1 to u_5 on a directed graph G_2 in Fig. 2.8 is $u_1,e_1^2,u_2,e_4^2,u_4,e_5^2,u_5$. A **trial** is a walk without repeated edge on the list. If a trial or a walk has the same first and last nodes on the list, then it is called a **circuit**. A **degree** of node is a number of edges that are connected to the node. A graph is **connected** if

every nodes on G has at least one edge connected to it. Otherwise it is **disconnected**.

Adjacent matrix of graph G_2

	u_1	u_2	u_3	u_4	u_5
u_1	1	1	0	0	0
u_2	0	0	1	1	0
u_3	0	0	0	0	0
u_4	0	1	0	0	1
u_5	0	0	0	0	0

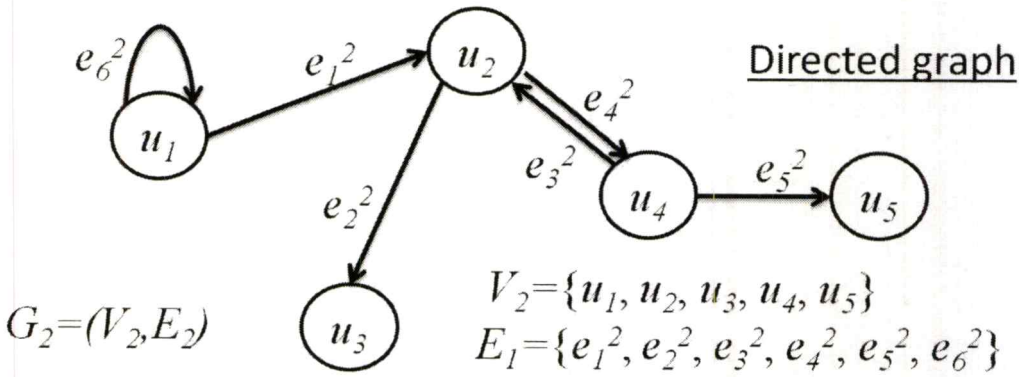


Figure 2.9 Adjacency matrix of graph G_2.

In addition to graph diagram representation, we can represent a graph using a matrix called an **adjacency matrix** (Fig. 2.9). An element in row i and column j of the matrix is a status of connection between node i and node j; it will be one if these nodes are connected together. Otherwise it will be zero. The adjacency matrix is always a square matrix. In a weighted graph, an adjacency matrix element can be a weight of edge that connects the pair of nodes on the graph.

- A **tree** is an undirected simple graph without any cycle in it. In Fig. 2.10, T_1 is not a tree because it has one cycle, while T_2 and T_3 are trees. In a tree, there are many kinds of nodes that can be defined as follows. A **root** is a node which is designated as a begin node of traverse. A **leave** node is a terminal node, which has only one degree. If a node is neither a root nor a leave node, then it is called an **internal** node. Classes of tree can be determined

as **a rooted** tree if there is a root node assigned on it, otherwise it is an **unrooted** tree

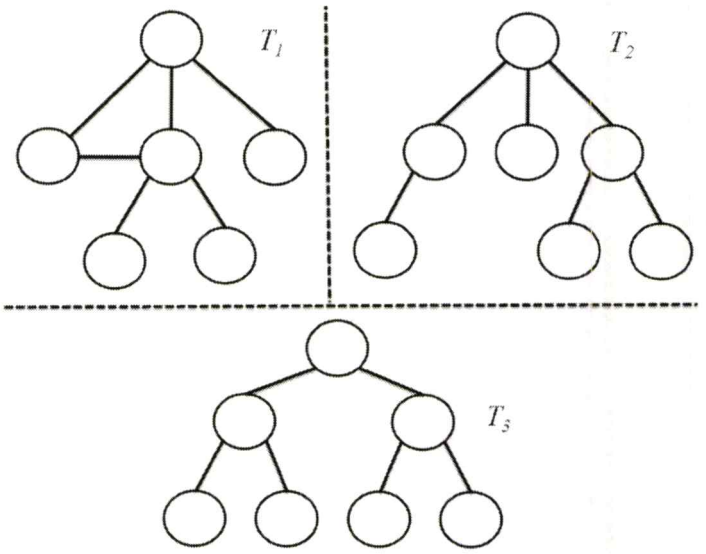


Figure 2.10 Trees and a non-tree graph.

A **high level** on a tree is defined by how far the node is from its root. The root itself has a high level zero. The nodes that connect to the root directly have high level one. Naturally, the nodes that connect to high level one have high level two. Hence, we can infer that any nodes connected to a node with high level N has high level $N+1$. If there are two nodes connected together, the node that has higher level is called a **child** node and the node with lower level is called **parent** node. A root is a node with no parent and a leave is a node with no child. An example of a tree with these definitions is depicted in Fig. 2.11. All nodes have only one parent except a root.

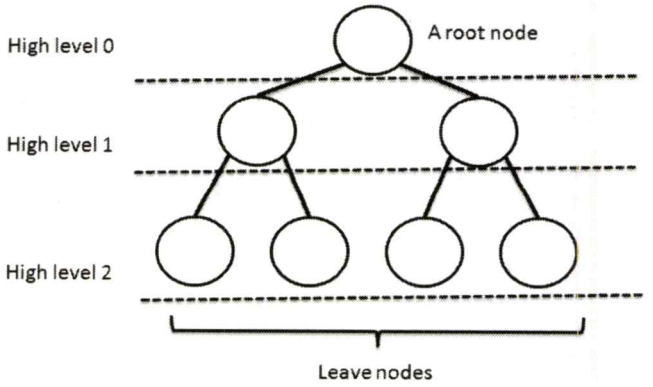


Figure 2.11 Anatomy of a rooted tree.

Furthermore, we can rooted trees from the number of its child nodes. If all parent nodes have less than two child nodes, the tree is called a binary tree.

2.4.2 Graph-based data clustering on genotypic data

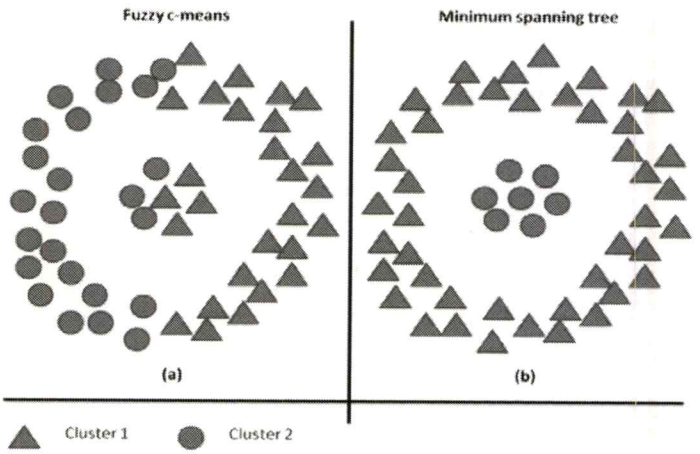


Figure 2.12 Results of data clustering using fuzzy c-means (left) and minimum spanning tree (right).

According to genotypic data clustering problem, we encounter arbitrary shape, complex structure, and various patterns of clusters. These problems cause a model-based method which assumes particular cluster shape to suffer from model mismatch. For example, a fuzzy c-mean algorithm which assumes circular cluster shape cannot separate data points from two clusters that form inner and outer rings (see Fig. 2.12). From this situation, the graph theory, which is able to tackle arbitrary shape of clusters efficiently, is adopted for clustering genetic data. The graph theory converts clustering problem to graph partitioning problem, which does not depend on a shape of clusters. It concentrates only on graph topology [1-2]. Because of this, graph is more fixable than traditional model-based method in data clustering. Results of model-based vs. graph-based clustering methods are shown in Fig. 2.12. In this section, we introduce some graph-based methods used in genotypic data clustering.

2.4.3 Minimum spanning tree

This method is introduced in [2] for a clustering of gene expression. All individuals are converted to nodes on a tree and their distances are constructed using minimum spanning tree (MST) algorithm. The minimum spanning tree is introduced by Otakar **Borůvka** in 1926. It is a tree which has a minimum weight from all possible spanning trees. At present, commonly used methods utilizing MST are Prim's algorithm [24] and Kruskal's algorithm [25] which are greedy algorithms.

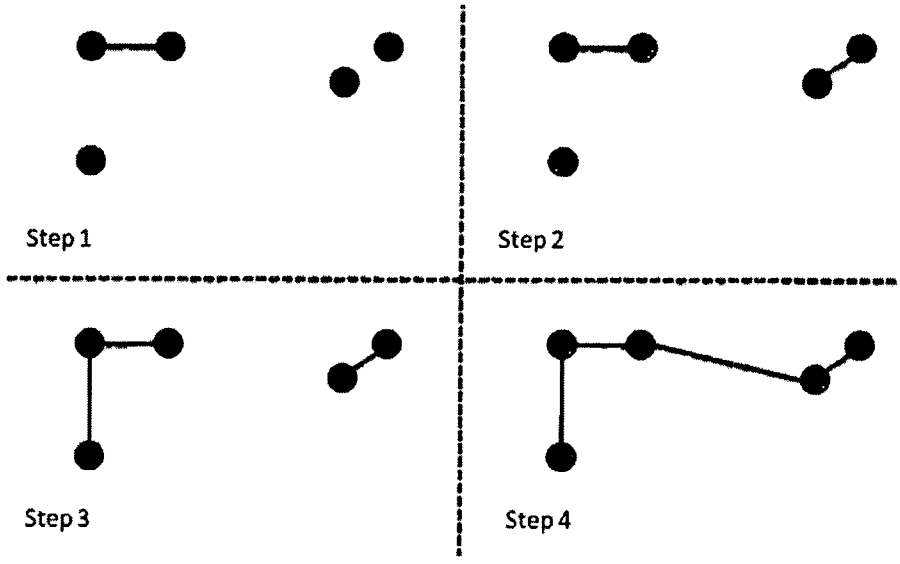


Figure 2.13 Steps of Kruskal's tree construction.

The first step of Kruskal's tree construction begins by finding a closest pair of nodes and combining them together. In the second step, other paths among merged nodes are eliminated and the most minimum path is kept. The process continues until all nodes are merged into the tree. The steps of Kruskal's algorithm are depicted on Fig. 2.13. An example of an MST is illustrated in Fig. 2.14.

After the minimum spanning tree is constructed, the next step of data clustering is to find the cutting point. Traditional method uses an edge with the longest weight as the cutting point. This method also has a higher precision of

clustering than other non-graph methods [1]. Although it works well in genetic clustering [2], computation cost of this method is so high that we should be aware.

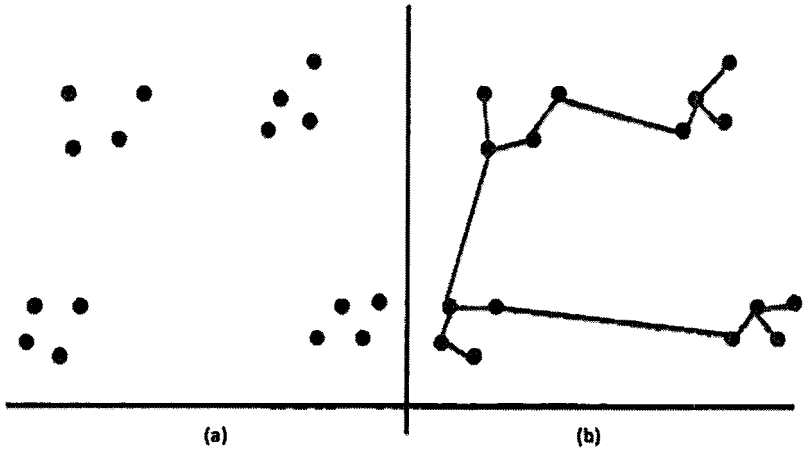


Figure 2.14 Example of an MST. (a) Node of all individuals in the initial state. (b) The MST After construction (modified from [1]).

2.4.4 Hierarchical clustering

A widely used method of genotypic data clustering is hierarchical clustering [3]. It contains a series of clusters in a hierarchical tree structure (see Fig. 2.15 for example). Actually, a hierarchical tree can be determined as a likely MST which has multi-level of clusters. In fact, a hierarchical tree is not a real MST because it always create a new node to represent merged nodes, while MST does not. A well-known tool for visualizing a hierarchical tree is the Dendrogram [18]. A group of leave nodes in the sub-tree represents a sub-cluster. The method of tree construction can be categorized into two types: bottom-up method and top-down method [18].

In **bottom-up** method, the initial state starts with all of the nodes as clusters. The second step, the nearest nodes will be merged together to form a new node by determining the cost value (distance), which is viewed as a parent node of the two merged nodes. The distance of the new parent node to all other nodes will be calculated. The process continues by repeating the second step until all of nodes are merged into the tree. The steps of tree construction is depicted in Fig. 2.16.

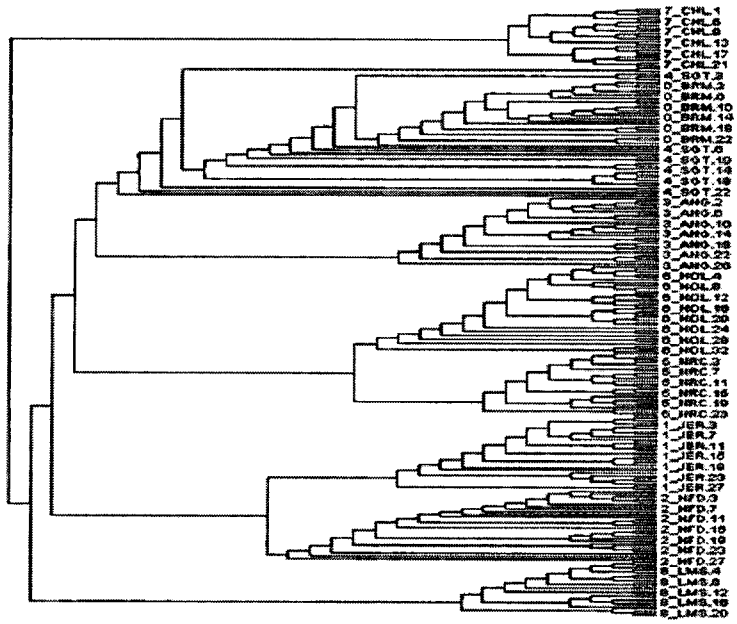


Figure 2.15 Example of a hierarchical tree from a bovine dataset.

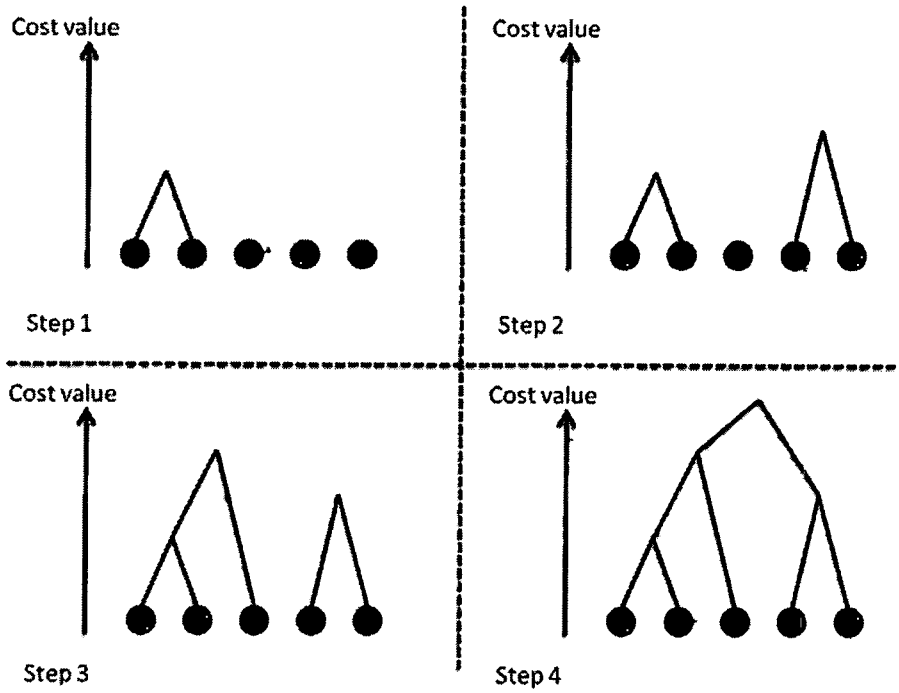


Figure 2.16 Steps of hierarchical tree construction using bottom-up method.

Common bottom-up method are single-link method [4], which chooses a distance between clusters by the minimum distance between each member of clusters, and complete-link method, which chooses a distance between clusters by the maximum distance between each member of clusters. Sometimes the distance among clusters can be calculated using the mean, mode, or other statistical distances. In contrast, the **top-down** method begins with one node and the next step is to divide the nodes into two sub-nodes. The process continues dividing until a stopping criterion is satisfied or only single individuals are left within a node.

In hierarchical clustering we also attempt to find the cutting point. The traditional method of finding the cutting point is to cut from the level of hierarchy of the tree or to use the longest edge for cutting. We used hierarchical tree for improving the clustering algorithm in Chapter 3.

2.4.5 Neighbor joining tree

Phylogenetic tree is used in almost every biological fields for representing relationships among individuals, species, genes, histories of populations, evolutionary links, and so on [26]. The most widely used phylogenetic tree is the Neighbor joining tree [7] or NJ tree, which approximates relationship among genetic data from its distance matrix [26]. It requires only a dissimilarity matrix for construction, so it has a fast computational speed [26]. In fact, the NJ tree is simply a subset of hierarchical tree constructed with a bottom-up method. Instead of choosing the closest nodes for merging first, NJ tree uses a minimum evolution criterion or the Q-criterion [8] as a cost function to choose the pair of nodes which has the minimum $Q(i,j)$ value according to the equation

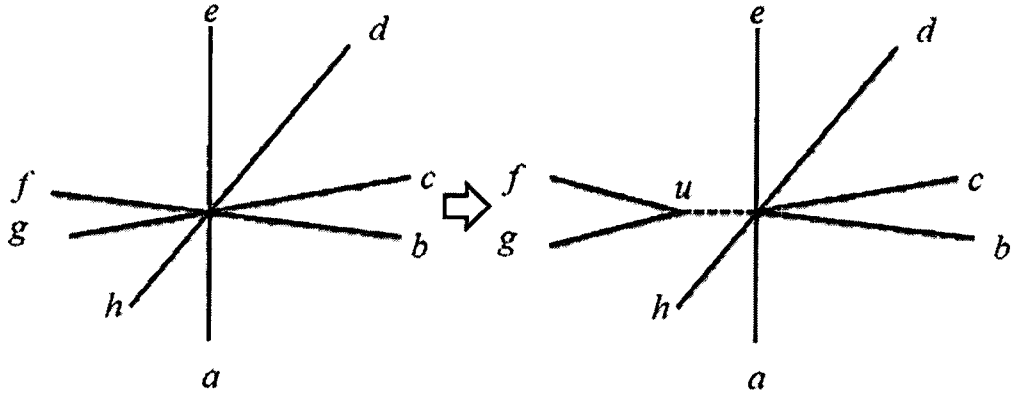
$$Q(i,j) = (r-2)d(i,j) - \sum_{k=1}^{r} d(i,k) - \sum_{k=1}^{r} d(j,k) \qquad (2.19)$$

where $d(\bullet,\bullet)$ is a dissimilarity value between node i and node j. In this research, we use the ASD matrix as our dissimilarity matrix. The variable r is a number of individuals which have yet been merged onto the tree. The process chooses the pair of nodes which produces the minimum value of $Q(i,j)$ for merging. After choosing the pair of nodes, the next step is to calculate a distance between the selected nodes and their new parent node using

$$d(f,u)=\frac{1}{2}d(f,g)+\frac{1}{2(r-2)}\left[\sum_{k=1}^{r}d(f,k)-\sum_{k=1}^{r}d(g,k)\right] \tag{2.20}$$

Here f and g are selected nodes and u is a new parent node of f and g. $d(i,j)$ is the Fitch-Margoliash distance between node i and j. Figure 2.17 illustrates this process. Then, distance from node u to all other non-selected nodes k is calculated by the equation

$$d(u,k)=\frac{1}{2}[d(f,k)-d(f,u)]+\frac{1}{2}[d(g,k)-d(g,u)] \tag{2.21}$$



Step 1: initial tree

Step 2: the tree after 1st iteration of neighbor joining

Figure 2.17 The NJ tree construction process.

The process repeats to choose a new pair of nodes until no non-selected nodes is left. The minimum evolution criterion hypothesis is to construct the minimum weight tree, which believes to be an approximation of a true tree. Furthermore, the NJ tree is a phylogenetic tree with hierarchical characteristic, making it easier to visualize and represent a highly complex spatial structure than traditional clustering-based methods [27]. This inspires us to use the NJ tree to develop the iNJclust algorithm in Chapter 4.

2.5 Clustering accuracy measurement using F-measure

One way to investigate the cluster accuracy of a clustering algorithm is to compare the clustering results to known-genetic labels of each individual within the dataset. If the clustering results are more similar to known-genetic labels, it means the clustering method has more accuracy. We adopt an F-measure [28] for comparing the results of clustering algorithm to known-genetic labels.

Given dataset φ, we denote $\varphi_1 = \{\varphi_1^1, \varphi_2^1, \varphi_3^1, ..., \varphi_\alpha^1\}$ as a set of known labels with α clusters and $\varphi_2 = \{\varphi_1^2, \varphi_2^2, \varphi_3^2, ..., \varphi_\beta^2\}$ as the clustering results of an algorithm of interest with β clusters. Depending on the algorithm, β may not equal α. The similarity between a set of clusters φ_1 and φ_2 can be calculated by

$$\kappa(\varphi_1, \varphi_2) = \sum_{i=1}^{\alpha} \frac{|\varphi_i^1|}{|\varphi|} \max_j \left(\frac{2\frac{|\varphi_i^1 \cap \varphi_j^2|^2}{|\varphi_i^1||\varphi_j^2|}}{\frac{|\varphi_i^1 \cap \varphi_j^2|}{|\varphi_i^1|} + \frac{|\varphi_i^1 \cap \varphi_j^2|}{|\varphi_j^2|}} \right) \tag{2.22}$$

Where $|\bullet|$ represents number of members in a set and $|\varphi|$ is a total number of individuals within the dataset. The F-measure value ranges from zero to one. The larger value of F-measure means more similarity between φ_1 and φ_2. If an F-measure

value of one clustering algorithm is larger than others, then that algorithm has more clustering accuracy.

Chapter 3
Improved iterative pruning principal component analysis

3.1 Introduction

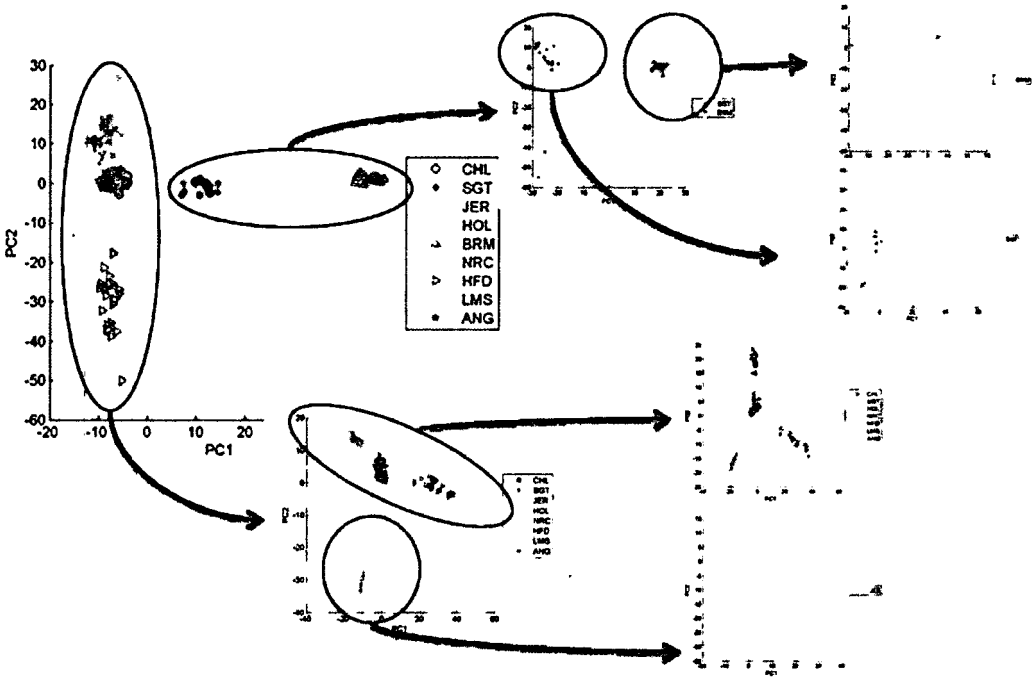


Figure 3.1 Iterations of PCA on subsets of data samples.

Various unsupervised clustering methods have been applied to genetic data to infer hidden characteristics such as population structures, ancestry information, or number of relationships between clusters of individuals. One unsupervised clustering framework, which has originally been proposed for detecting population structure in large, complex genotypic datasets, is the iterative pruning principal component analysis or ipPCA [5]. It performs PCA iteratively on subsets of data samples and clusters them using the fuzzy c-mean algorithm (Fig. 3.1). In contrast to other classical techniques where the number of clusters is needed *a priori*, the framework gives an estimated number of clusters and individual assignments as outputs.

Although ipPCA is robust when dealing with closely-related clusters that may appear conglomerate and gives reasonable individual assignments, the estimated number of clusters is not accurate. This is due to the use of the Tracy-Widom statistic [29] to identify the stopping point in the iterative process. Subsequently, the second algorithm in the ipPCA framework called EigenDev-ipPCA has been proposed [6], which uses a new stopping criterion called EigenDev. It has been shown to produce better estimates of the number of clusters. Nevertheless, the errors in individual assignments remain. We suspect that these errors originate mainly from the choice of the clustering algorithm. This motivates us to improve the quality of individual assignments in the ipPCA framework by modifying its clustering technique.

In this chapter we introduce the hierarchical clustering concept from graph theory into the ipPCA framework. In addition, we have adopted an unsupervised feature selection method from [10] for reducing the data dimension in order to improve the computational efficiency of our algorithm. To assess the performance of the proposed algorithm, we apply it to bovine and sheep genotype datasets. The evaluation of results in terms of individual assignment errors and cluster quality is presented in Chapter 5.

3.2 Method

Figure 3.2 depicts the improved ipPCA framework, where the proposed modifications are highlighted in gray boxes. In the original ipPCA [5], [6], the data sequences are projected onto the PCA subspace, then clustered using fuzzy c-mean. The process iterates until the stopping criterion is reached. For our improvements, the data vectors first enter the feature selection step, where the data dimension is reduced. Then the data is projected onto the principal component axes before clustering. The proposed clustering method is the hierarchical clustering [3], hence we termed the modified algorithm **HiClust-ipPCA** to emphasize the new incorporated technique. The details of the modified framework including our proposed changes are given below.

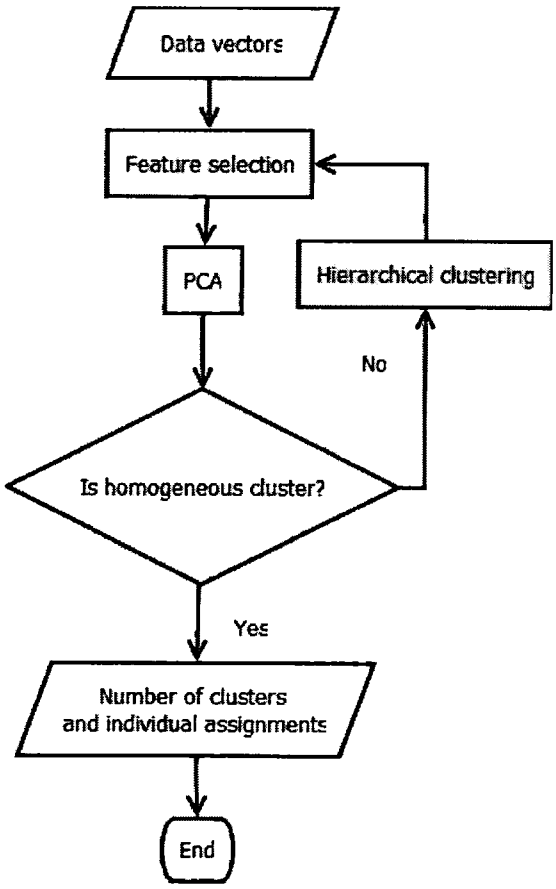


Figure 3.2 Diagram of the proposed HiClust-ipPCA algorithm.

3.2.1 PCA-based feature selection

Given an N-dimensional data vector from M individuals genotyped at L SNPs, we concatenate all data vectors to form an $M \times L$ data matrix X as the input. Typically, $M << L$. We adopt [10] as a method for reducing data dimension, keeping only dimensions with the most amount of information. Our hypothesis is that by keeping the SNPs that contains essential information, the clustering results should remain the same (Fig. 3.3).

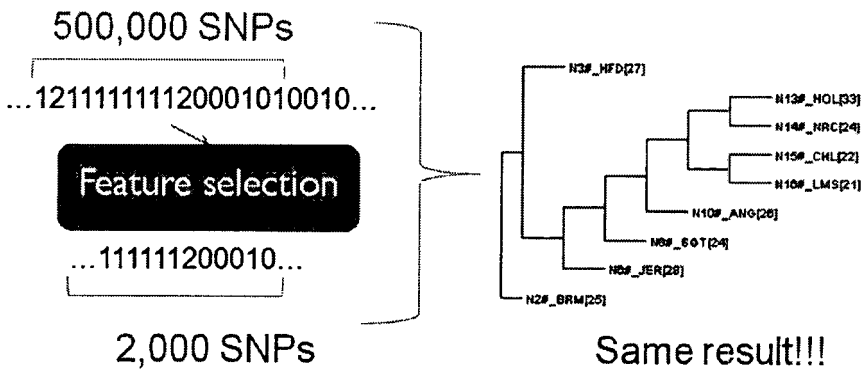


Figure 3.3 Hypothesis for reducing SNPs using feature selection.

This feature selection technique is based on the principal component analysis, thus it is readily compatible with the ipPCA framework. The informative rank of the data matrix X can be computed from the singular value decomposition,

$$X = USV^T . \tag{3.1}$$

The importance score $P(j), j = 1, \ldots, N$ of the j^{th} data dimension is obtained by summing the j^{th} component of the right singular vectors from all individuals,

$$P(j) = \sum_{i=0}^{m} V(j,i). \qquad (\tag{3.2}$$

The larger the value of $P(j)$, the more significant that dimension is. We rank $P(j)$ and choose only the first ϕ informative dimensions. Therefore, the data matrix X is reduced to $\tilde{X}_{M\times\phi}$, where $\phi \leq L$. In the next step, $\tilde{X}$ is projected onto its principal components. The homogeneity of these projected data points are determined using an EigenDev measure from [6],

$$\text{EigenDev} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\log(\hat{\sigma}_i^2) - \log(\sigma_i^2))} \tag{3.3}$$

$$\log(\hat{\sigma}_i^2) = \log(\sigma_1^2) + (i-1)\left(\frac{\log(\sigma_M^2) - \log(\sigma_1^2)}{m-1}\right) \tag{3.4}$$

and $\sigma_i^2 \quad i = 1,...,M$ are the M eigenvalues of $\tilde{X}\tilde{X}^T$ ranked in descending order. If the EigenDev value exceeds a user defined threshold, the data points are considered coming from more than one cluster and we proceed to the hierarchical clustering step. Otherwise, the cluster is considered homogeneous and the algorithm terminates.

3.2.2 Graph-theoretic hierarchical clustering

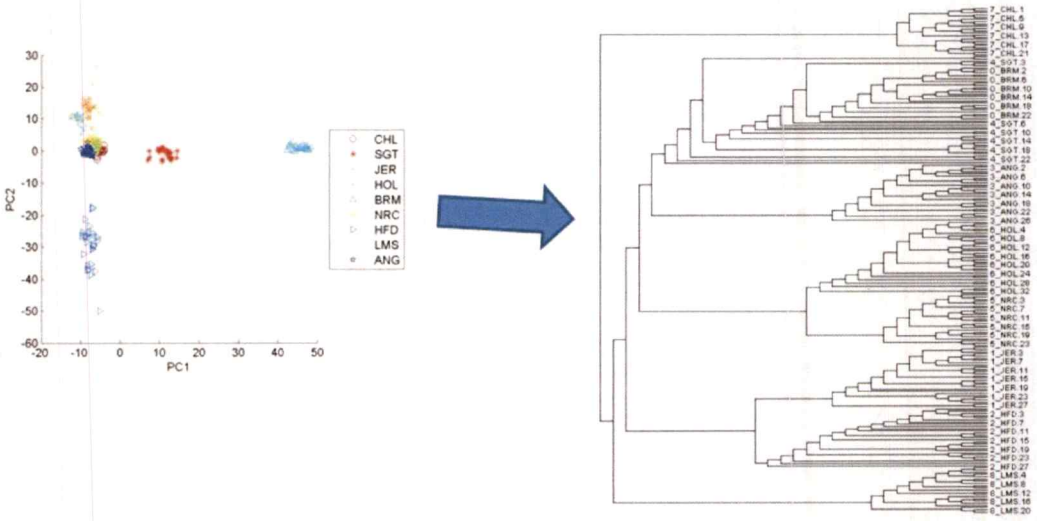


Figure 3.4 Conversion of data from principal component space to hierarchical tree.

We choose to use hierarchical tree [3] (Figure 3.4) for clustering genetic data. The typical method in hierarchical clustering is cutting the tree edges from top down to separate the nodes into a desired number of clusters. Example of tree cutting is shown in Fig. 3.5. Five nodes A, B, C, D, and E are used to form a hierarchical tree. The branch labeled $\mathrm{W_{de}}$ is cut, splitting the tree into two clusters $C_1 = \{A, B, C\}$ and $C_2 = \{D, E\}$.

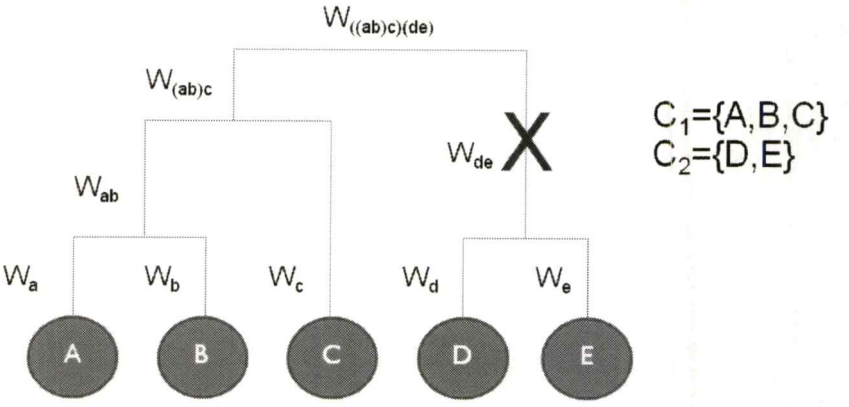


Figure 3.5 An example of clustering by cutting on hierarchical tree.

The top-down tree cutting does not make biological sense because the order in which the nodes are combined does not always translate to the actual genetic distance between clusters. The number of clusters n also needs to be known. We eliminate the need for n by folding the hierarchical tree into the iterative process, using Ward's Minimum-variance [9] as a measure for genetic distance between clusters. The distance of Ward's minimum variance method is the ANOVA sum of squares between pair of clusters from $Y_{M\times r}$ in Eq. 2.4,

$$W_{ab} = \frac{\left\|\bar{y}_a - \bar{y}_b\right\|^2}{1/m_a + 1/m_b} \tag{3.5}$$

where $\bar{y}_a$ and $\bar{y}_b$ represent means of cluster a and b and m_a and m_b represent the numbers of individuals within the clusters, respectively. After we construct the hierarchical tree, we cluster by cutting the edge between two nodes with largest distance W_{ab}, splitting the tree into 2 sub trees. To avoid the effect from outliers, the nodes which have less than 10% of the total number of individuals are excluded from consideration.

3.3 The HiClust-ipPCA Algorithm implementation

The HiClust-ipPCA algorithm is implemented in C++ language using Numerical Recipes [30] and Lapack++ [31] libraries on a Intel(R) Core(TM) i5-2500K CPU 3.30 GHz PC with 16 GB RAM. There are versions of the algorithm for both Windows and UNIX

operating systems. The system architecture design of the HiClust-ipPCA algorithm in Figure 3.6 can be divided into the following parts.

- *Main* is a center connection to gain user inputs from command line or graphic user interface. It uses *ClOManagement* object so that the system can manage inputs, outputs, and other processes together.
- *ClOManagement* is a class for system management. It connects to every parts of the system: input part, output part, and process parts, while the system is processing. A configuration file is parsed in order to obtain parameters of framework by this class. Afterwards, the other parts of the system will receive the parameters.
- *CMatrixIO* is a class which fetches input file for converting it to matrix variable, and vice versa.
- *CipPCA* is a central class of the ipPCA framework. Its purposes are to run iteration of process and to call related functions that are essential for processing the tasks.
- *CClustering* is a class which runs PCA transformation parts, analyzes homogeneity of clusters using EigenDev, splits data into new clusters, and pares ordered tree into Newick format to be displayed by other software, such as Dendroscope.
- *CMSTClust* is a class which constructs the hierarchical tree. It calls functions regarding tree building and discriminates subclusters within the tree.
- *CPCACorrelated* is a class which is used for feature selection using PCA information from an input.
- *CComputingLIB* is a class library used for all calculation within the framework, such as matrix operation, similarity measurement, and so on. It communicates with C++ standard library so that framework can use necessary classes, such as string, vector, and basic mathematic operations. Moreover, this library also connects to LAPACK++, which calculates matrix operations.

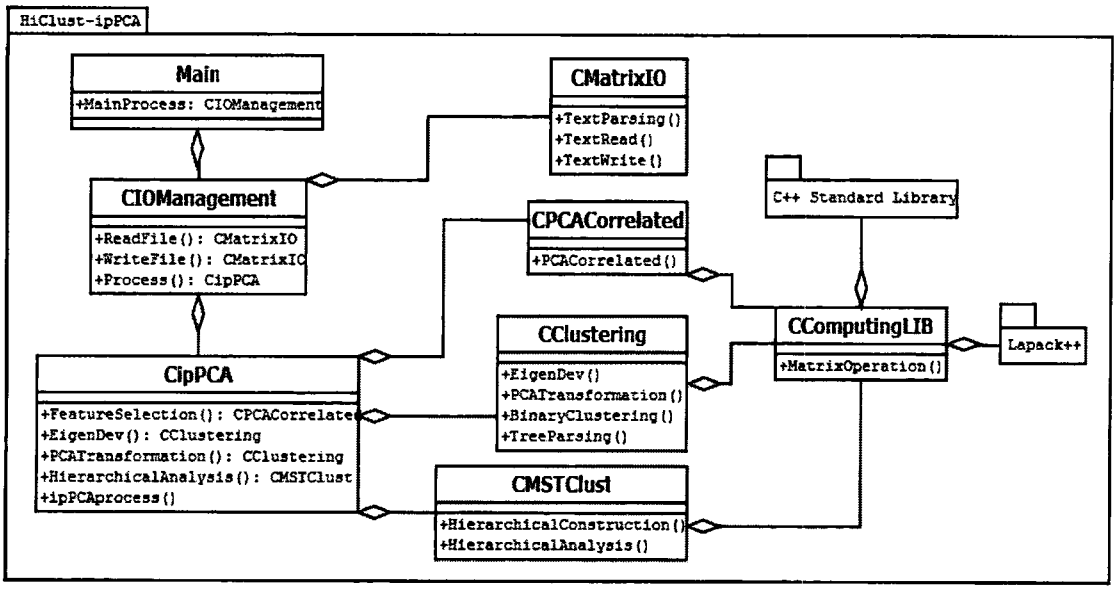


Figure 3.6 System architecture design of The HiClust-ipPCA algorithm.

Chapter 4

Iterative neighbor-joining tree clustering

4.1 Introduction

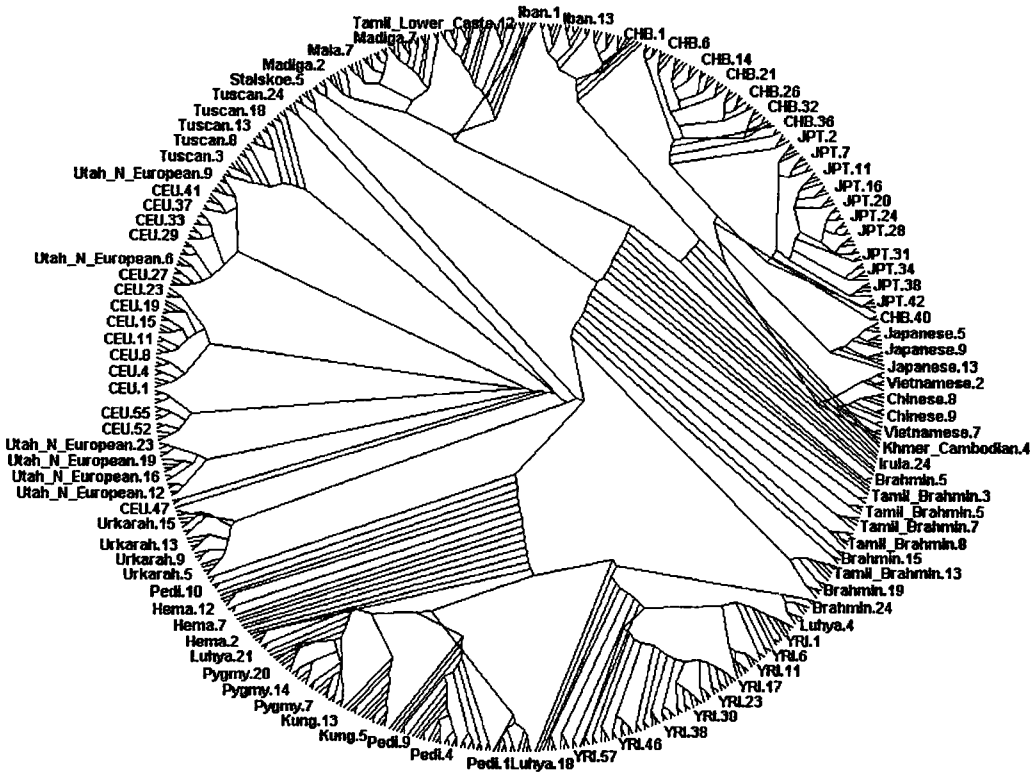


Figure 4.1 A phylogenetic tree of human.

In genetic data clustering problems, we often encounter clusters that cannot be differentiated correctly and efficiently using model-based methods due to their arbitrary shapes and minimal characteristic variations. An alternative approach is to convert the clustering problem into a graph partitioning problem. Data points are viewed as leaf nodes on a graph, the pattern of clusters as graph topology. Graph partitioning is performed by selectively cutting the graph edges until a desired number of clusters is achieved. Thus, it is independent of any assumptions on cluster shapes and is found to work reasonably well as a clustering method [1, 2].

A key problem with many graph clustering methods in genetics, e.g., hierarchical tree clustering [3] is that the edge lengths on the graph do not translate

to actual genetic relationships among individuals. These relationships can be inferred in phylogenetic trees (see Fig. 4.1). One well-known tree is Neighbor-joining (NJ) tree [8], which is constructed from a distance matrix of a data. The edge lengths on an NJ tree are proportional to genetic distances between the tree nodes. Surprisingly, the NJ tree has been utilized only in interpreting genetic relationships, but not for the clustering problems.

The distinct advantages of graph partitioning and NJ tree inspire us to develop a new framework for genotypic data clustering by combining the power of the two– using graph partitioning on an NJ tree. This enables us to cluster complex patterns according to the similarities of their genetic codes. We design an iterative scheme similar to [5, 6] and HiClust-ipPCA in Chapter 3. Besides increasing the algorithm's ability to differentiate closely related clusters, the iterative process allows automatic correction of the NJ tree topologies. The resulting clustering algorithm is termed **iterative Neighbor-Joining tree clustering** or **iNJclust**. The investigation of the performance of the iNJclust algorithm using two simulated datasets and three large, highly complex genotypic datasets from human, sheep, and bovine are reported in Chapter 5.

4.2 Method

The system flowchart of the proposed algorithm is depicted in Fig. 4.2. Similar to the set up in Chapter 3, the SNP sequences of M individuals forms the input matrix X of the iNJclust algorithm. Each row vector $x_i, i = 1,...,M$ is genotyped from L loci of individual i, at which there are two alleles of either A (major allele) or a (minor allele) for three possible genotypes (AA, Aa, aa). As mentioned in the previous chapter, the SNP sequence is encoded by counting the number of minor allele a. Therefore, x_i is an L-dimensional vector of numerical values 0,1, or 2.

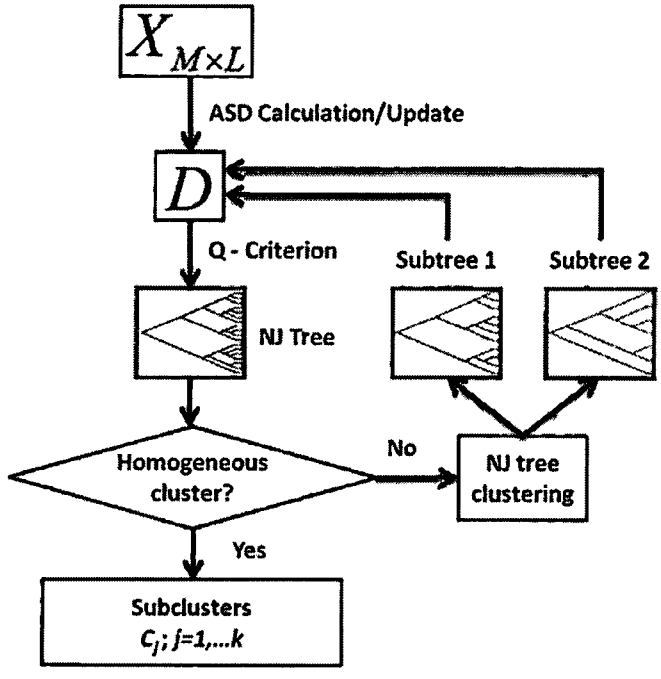


Figure 4.2 The flowchart of the iNJclust algorithm.

After receiving the input X, the iNJclust algorithm computes the ASD matrix D, whose elements are

$$d(i,j)=\frac{1}{L}\left\|x_i-x_j\right\|_1 \quad i,j=1,...,M \tag{4.1}$$

Therefore, $d(i,j)$ is proportional to the L1-norm of the pairwise difference between the SNP sequences of individual i and those of individual j. The smaller the value of $d(i,j)$, the closer they are genetically.

The ASD matrix D is subsequently used to construct the NJ tree. Treating each individual as a leaf node, a pair of nodes who are nearest to each other as measured by the minimum evolution criteria are merged into a parent node. The tree construction process continues until all nodes are merged onto the NJ tree. The algorithm then determines if the cluster is homogeneous, i.e., all individuals on the tree come from the same subpopulation. If the cluster is said to be heterogeneous, the algorithm performs clustering on the NJ tree by bisecting the tree into two subtrees. The NJ tree is splitted at the longest branch (edge) between two nodes

within the tree. Similar to the condition in the HiClust-ipPCA algorithm, we only consider the branches that have more than 10% of individuals left on the branches to avoid a mis-cluster due to noise or outliers. In the next stage, both subtrees cycle back to the ASD update step where they are processed independently.

Note that the ASD matrix is only calculated once for the original input X. The ASD matrix of individual within each subtree is readily available by selecting elements of the original ASD matrix D corresponding to the appropriate subset of individuals. After the new ASD matrix is obtained, the NJ tree of each subset of individual is constructed. We discover that the new NJ tree reconstructed at each iteration may have different topology from the old subtree, since the genetic relationships illustrated in the new tree only account for those individual within the subcluster, thus enhancing the resolution of the algorithm to differentiate closely-related subpopulations. The cluster homogeneity criterion is than checked against the new tree. Two criteria are proposed for checking cluster homogeneity, i.e., the cluster contains only a single population. We will explain both criteria in details in the following subsection. Once the subtree is deemed homogenous, it is output from the process as one of the resulting subpopulation C_i. Because the NJ tree clustering only bisects the tree in each iteration, the number of final iterations k provides an estimated number of subpopulations. We choose to present the clustering result of the iNJclust algorithm in the form of a binary tree, whose terminal nodes represent the final subpopulations and the edge lengths preserve the genetic relationship among inferred subpopulations.

4.3 Homogeneity measurements

4.3.1 Topology method

To decide whether the algorithm should perform the NJ-tree clustering in each iteration, we propose to analyze the homogeneity of the sub-tree from its topological pattern. If the homogeneity criterion is met, the iNJclust algorithm terminates. This criterion stems from the NJ tree construction mechanism. During the early stage of tree building, nodes from the same subpopulation would be merged first before nodes from different subpopulations are combined. Consequently, it is observed that the bifurcating sub-tree of a homogeneous cluster always contain one

descendant with the so-called **UT1** topology pattern. The **UT1** pattern is an unbalanced bifurcating tree with a leaf node as one descendant, as depicted in the dashed-line box of Figure 4.3

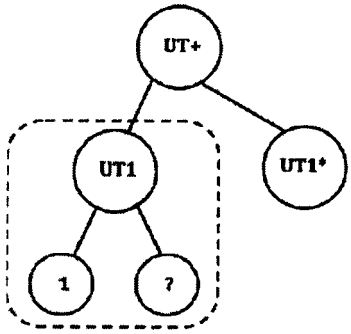


Figure 4.3 The homogeneous topology.

Let $\mathrm{UT1}_k^+$ be a tree topology at the k^{th} iNJclust iteration before NJ tree clustering. The iNJclust algorithm is to terminate when the homogeneous topology,

$$\mathrm{UT1}_k^+ = \mathrm{UT1}_{(k+1)} \cup \mathrm{UT1}_{(k+1)}^* \tag{4.2}$$

is detected. $\mathrm{UT1}_{k+1}^*$ can be of type $\mathrm{UT1}$ or else. Intuitively, this stopping criterion indicates that a heterogeneous cluster (containing more than two subpopulations) should not be too close to the leaf nodes.

4.3.2 Fixation index difference method

Although the topological stopping criterion in section 4.3.1 works well in practice, it is an ad-hoc method based purely on data observations with no concrete mathematical background. Therefore, we propose a second homogeneity determination criterion called ΔF, which follows directly from the behavior of the fixation index value (F_{ST}) after data clustering. We prove that the fixation index monotonically increases at each iNJclust iteration until a homogeneous cluster is formed.

Proposition 1 (F_{ST} behavior after data clustering):

Suppose the k^{th} iteration of the iNJclust algorithm has its F_{ST} value F_k. The new F_{ST} value F_{k+1} at the next iteration is AT LEAST F_k.

Proof:

At the k^{th} iteration, the original cluster of M individuals is divided into $k \geq 2$ subclusters $C_1, C_2, ..., C_k$ containing $m_1, m_2, ..., m_k$ individuals, respectively. Let the corresponding major allele frequencies within each subcluster be $p_1, p_2, ..., p_k$ (Eq. (2.7)). Hence, the average major allele frequencies of the entire dataset is $\overline{p} = \frac{1}{M}\sum_{i=1}^{k} p_i m_i$ (Eq. (2.11)), and the quantity $H_T = 2\overline{p}(1-\overline{p})$ (Eq. (2.15)) represents the expected normal-type allele frequency of the data.

Using Eq. (2.17),

$$F_k = \frac{H_T - H_S^k}{H_T} = 1 - \frac{H_S^k}{H_T}, \tag{4.3}$$

Hence, to proof that $F_{k+1} \geq F_k$ it is equivalent to showing that $H_S^{k+1} \leq H_S^k$. In the thesis, we adopt Jensen's inequality (Appendix A) in order to prove $H_S^{k+1} \leq H_S^k$. The next step is to rearrange equations of H_S^k and H_S^{k+1} to fit the form of Eq. (A2).

According to Eq. (2.14), we write,

$$H_S^k = \frac{1}{M}\sum_{j=1}^{k} H_{\exp}(p_j) m_j = C + \frac{1}{M} H_{\exp}(p_k) m_k \tag{4.4}$$

where

$$C \equiv \frac{1}{M}\sum_{j=1}^{k-1} H_{\exp}(p_j) m_j \tag{4.5}$$

and in accordance with Eq. (2.10),

$$H_{\exp}(p_j) = 2p_j(1-p_j) \tag{4.6}$$

If the cluster considered at iteration k with the value H_S^k , is still heterogeneous, the NJ tree clustering bisects the cluster which has m_k individuals and major allele frequencies p_k into two subclusters, which have m_{k+1}^1 and m_{k+1}^2 individuals and have major allele frequencies p_{k+1}^1 and p_{k+1}^2 respectively for determining H_S^{k+1}. Consequently,.

$$H_S^{k+1} = C + \frac{1}{M}\left[H_{\exp}(p_{k+1}^1)m_{k+1}^1 + H_{\exp}(p_{k+1}^2)m_{k+1}^2 \right], \tag{4.7}$$

where

$$m_k = m_{k+1}^1 + m_{k+1}^2 \tag{4.8}$$

Trivially,

$$p_k = \frac{p_{k+1}^1 m_{k+1}^1 + p_{k+1}^2 m_{k+1}^2}{m_k}. \tag{4.9}$$

For ease of reading, we drop the subscript denoting the iteration number and shorthand p_{k+1}^1 as p_1 , p_{k+1}^2 as p_2 , m_{k+1}^1 and m_{k+1}^2 as m^1 and m^2 , respectively. Thus,

$$H_S^{k+1} = C + \frac{2}{M}\left[p^1(1-p^1)m^1 + p^2(1-p^2)m^2 \right] \tag{4.10}$$

$$H_S^{k+1} = C - \frac{2(m^1+m^2)}{M}\left[-p^1(1-p^1)\frac{m^1}{m^1+m^2} - p^2(1-p^2)\frac{m^2}{m^1+m^2} \right] \tag{4.11}$$

$$H_S^{k+1} = C - \frac{2(m^1+m^2)}{M}\left[f(p^1)\lambda_1 + f(p^2)\lambda_2 \right] \tag{4.12}$$

using $f(x) = -x(1-x)$, $\lambda_1 = \frac{m^1}{m^1+m^2}$, and $\lambda_2 = \frac{m^2}{m^1+m^2}$.

Since $\lambda_1, \lambda_2 > 0$, $\lambda_1 + \lambda_2 = 1$, and $f(y)$ is a continuous concave up function, it follows from Jensen's inequality (Eq. (A2) in Appendix A) that

$$C - \frac{2(m^1+m^2)}{M}\left[f(p^1)\lambda_1 + f(p^2)\lambda_2 \right] \le C - \frac{2(m^1+m^2)}{M}\left[f(p^1\lambda_1 + p^2\lambda_2) \right] \tag{4.13}$$

$$H_S^{k+1} \le C - \frac{2(m^1+m^2)}{M}\left[f(p^1\lambda_1 + p^2\lambda_2)\right] \tag{4.14}$$

$$H_S^{k+1} \le C - \frac{2(m^1+m^2)}{M}\left[f(p_k)\right] \tag{4.15}$$

$$H_S^{k+1} \le C + \frac{1}{M}H_{\exp}(p_k)m_k \tag{4.16}$$

$$H_S^{k+1} \le H_S^k. \tag{4.17}$$

When the cluster before splitting is already homogenous, the average major allele frequencies $p^1 \approx p^2 \approx p_k$. Thus,

$$H_S^{k+1} = H_S^k \qquad \square$$

Furthermore, in more general case, the proposition holds when a cluster is divided into $L \ge 2$ subclusters, since we can write a more general form of Eq. (4.9).

$$p_k = \frac{\sum_{l=1}^{L} p_l m_l}{\sum_{l=1}^{L} m_l} \tag{4.17}$$

$$\lambda_k = \frac{m_l}{\sum_{l=1}^{L} m_l} \tag{4.18}$$

and Jensen's inequality still applies.

From this proposition, the F_{ST} value of the data after each iNJclust iteration increases monotonically and converges after all homogenous subpopulations have been clustered. We propose using the difference of the F_{ST} value,

$$\Delta F = F_{k+1} - F_k \tag{4.19}$$

to detect homogeneous clusters. We announce that the cluster is homogeneous and terminate the iNJclust iteration for that subcluster if ΔF is sufficiently small. The smaller the ΔF threshold, the higher the resolution of the iNJclust algorithm to differentiate between clusters.

4.4 The iNJclust Algorithm implementation

The iNJclust algorithm is also implemented in C++ language using Numerical Recipes [30] and Lapack++ [31] libraries on a Intel(R) Core(TM) i5-2500K CPU 3.30 GHz PC with 16 GB RAM. There are versions of the algorithm for both Windows and UNIX operating systems. The system architecture design of the iNJclust algorithm is depicted in Figure 4.4.

- *Main* is a center connection to receive user inputs from command line or graphic user interface. It uses CManagement object so that the system can manage inputs, outputs, and other processes together.
- *CManagement* is a class for system management. It connects to every parts of system: input part, output part, and process parts, while the system is processing. A configuration file is parsed in order to obtain parameters of framework by this class. Then, the other parts of the system will receive the parameters afterwards.
- *iNJclust* is a central class of iNJclust framework. Its purposes are to run iteration of process and call related functions that are essential for processing the tasks.
- *iNJtree* is a class which constructs NJ tree, traverses the tree for finding the cutting point, and splits the tree afterwards.

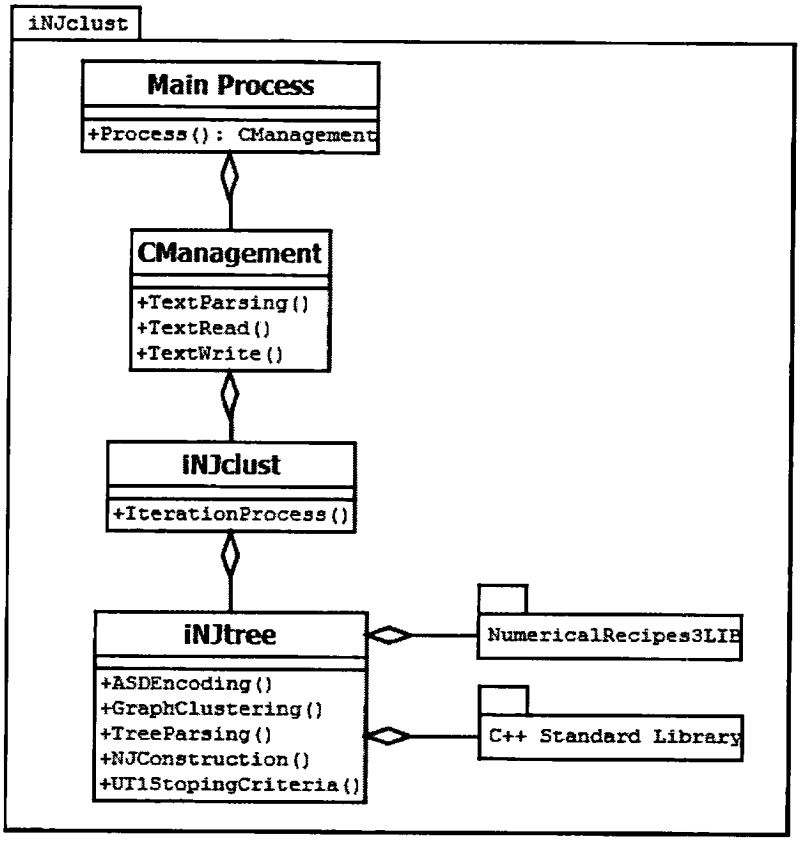


Figure 4.4 The system architecture design of the iNJclust algorithm.

Chapter 5
Results

5.1 Introduction

We test the performance of both the HiClust-ipPCA algorithm and the iNJclust algorithm. The HiClust-ipPCA algorithm is compared against the second version of the ipPCA framework, the EigenDev-ipPCA algorithm. We compare the iNJclust algorithm to an algorithm called AWclust [17], which is conceptually close to our proposed method. AWclust is the only existing method which uses the ASD matrix as input and performs clustering on a tree [32]. However, it uses a hierarchical tree [3] instead of an NJ tree, and does not include any iterative process. The main limitation of the AWclust algorithm is that it can automatically detect subpopulations only up to 16 clusters. More than that, the number of clusters is required *a priori*. Furthermore, to illustrate the importance of tree rebuilding on the clustering performance we also investigate a simplified version of our algorithm (termed NJclust) where we iterate on the same stopping criterion. However, the NJclust algorithm builds the NJ tree only in the first iteration and uses the same tree on all subsequent iterations.

The similarities and differences of all methods used in the clustering performance testing in this chapter are presented in Table 5.1. The results of all algorithms are in Newick format. In order to visualize the tree we adopt software Dendroscope [33] for displaying the results.

Table 5.1 Comparison of methods used for testing clustering performances

	EigenDev-ipPCA	HiClust-ipPCA	AWclust	iNJclust	NJclust
Input	Genetic sequences	Genetic sequences	ASD matrix	ASD matrix	ASD matrix
Stopping criteria	EigenDev	EigenDev	Gap statistic	UT1, ΔF	UT1, ΔF
Clustering algorithm	Fuzzy c-mean clustering	Hierarchical clustering with feature selection	Hierarchical clustering	NJ tree clustering with tree rebuilding	NJ tree clustering (without tree rebuilding)
Output	Individual assignment, Number of clusters	Individual assignment, Number of clusters	Individual assignment, Number of clusters	Individual assignment, Number of clusters, Cluster relationships	Individual assignment, Number of clusters

5.2 The HiClust-ipPCA algorithm performances

5.2.1 Effect of hierarchical clustering

We first evaluate the effect of the hierarchical clustering method on the performance of the algorithm by comparing clustering results from the HiClust-ipPCA algorithm with that of the EigenDev-ipPCA algorithm using two genotype datasets from bovine and sheep. The first dataset comes from 47 breeds of bovine, having 1089 individuals and 44,706 SNPs [34]. The second dataset is a collection of 28 sheep breeds with 392 individuals and 1,406 SNPs [35].

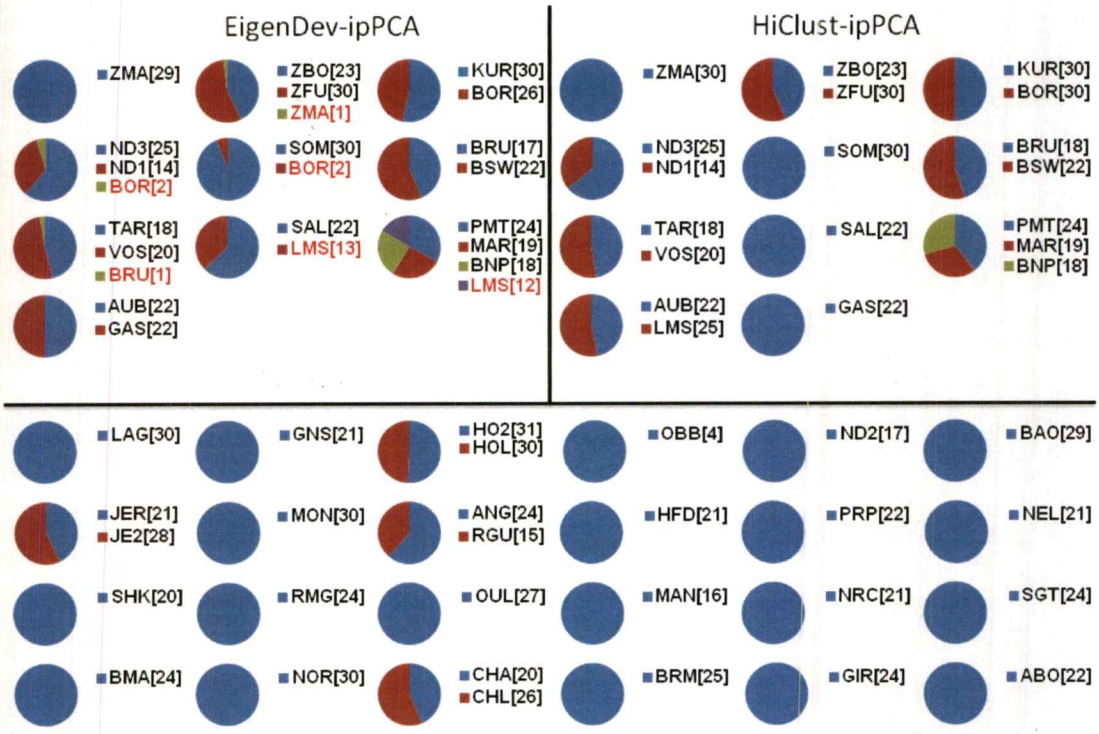


Figure 5.1 Clustering results of the 47-breed bovine dataset. Top-left panel: EigenDev-ipPCA; Top-right panel: HiClust-ipPCA. Bottom panel: Similar results between the two algorithms.

The clustering results for the bovine dataset are shown in Fig. 5.1. The EigenDev threshold of 0.37 is used in both algorithms. Each circle in the figure represents a cluster assigned by the algorithms. The colors on the circle represent subpopulations within the cluster. The numbers of individuals are denoted in the square brackets after cluster labels. The estimated cluster numbers are 34 for the EigenDev-ipPCA algorithm and 35 for the HiClust-ipPCA algorithm. Although the estimated numbers of cluster are lower than 47, it is hardly unexpected since some breeds are extremely close genetically. The results are also comparable to the original results reported in [34]. The cluster assignments in the bottom panel are similar between the HiClust-ipPCA and the EigenDev-ipPCA algorithms. Different cluster assignments are depicted in the top panel of Fig. 5.1. It is observed that the clusters from the EigenDev-ipPCA algorithm have more outliers. The HiClust-ipPCA algorithm produces no outlier, but there are clusters with mixed labels from breeds that are genetically similar. For example, the outliers from breeds BOR, BRU, and ZMA (highlighted in red) are eliminated when we use the HiClust-ipPCA algorithm.

The LMS breed which is initially separated into two clusters by the EigenDev-ipPCA algorithm is correctly combined into one cluster by the HiClust-ipPCA algorithm.

Figure 5.2 depicts the clustering results for the 28-breed sheep dataset with the EigenDev threshold of 0.30. The estimated cluster number from the EigenDev-ipPCA algorithm is 21; The HiClust-ipPCA algorithm detects 27 clusters out of 28 clusters. It is evident that clusters from the EigenDev-ipPCA algorithm (left panel of Fig.5.2) contains many more outliers (highlighted in red). Half of the clusters are also mixtures of at least three breeds. In contrast, the HiClust-ipPCA clusters are mostly pure with exception of only three mixed clusters. This improvement is also evident visually from the pie charts, where outliers presented in the EigenDev-ipPCA clusters such as COM, USDA DOS, SOA, and TIB breeds are assigned to the appropriate clusters by the HiClust-ipPCA algorithm.

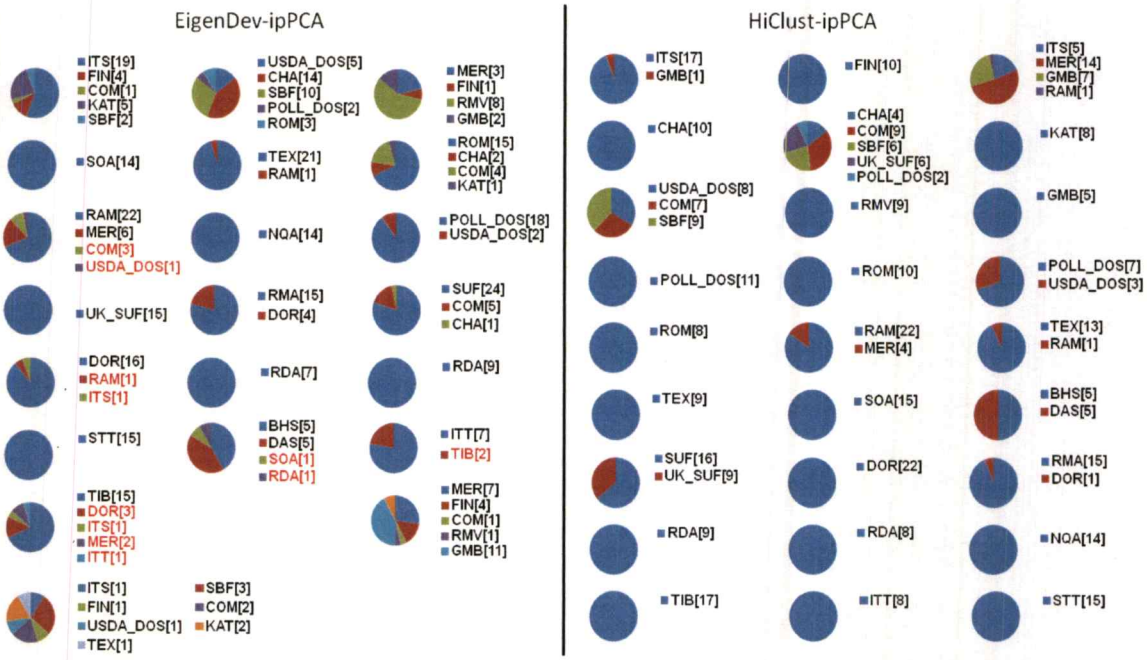


Figure 5.2 Clustering results of the 28-breed sheep dataset. Left: EigenDev-ipPCA. Right: HiClust-ipPCA. The EigenDev threshold is 0.30.

To quantify the clustering performance of the HiClust-ipPCA algorithm, we also compared the similarity of individual assignments with known genetic labels using F-measure [28] in Eq. (2.22). The known labels that come with the data are provided by the experts genotyping the animals. Hence, it is assumed that the labels represent

the true subpopulations. The F-measure values of the EigenDev-ipPCA algorithm and the HiClust-ipPCA algorithm are shown in Table 5.2. The F-measure values of the HiClust-ipPCA algorithm are greater than the values from the EigenDev-ipPCA algorithm on both datasets, which confirming the better clustering performance observed in Fig. 5.1 and 5.2.

Table 5.2 F-measure values of the EigenDev-ipPCA and the HiClust-ipPCA algorithms on bovine 47 breeds and sheep 28 breeds datasets.

	EigenDev-ipPCA	HiClust-ipPCA
Bovine 47 breeds	0.889291	0.914081
Sheep 28 breeds	0.716138	0.778155

5.2.2 Effect of feature selection

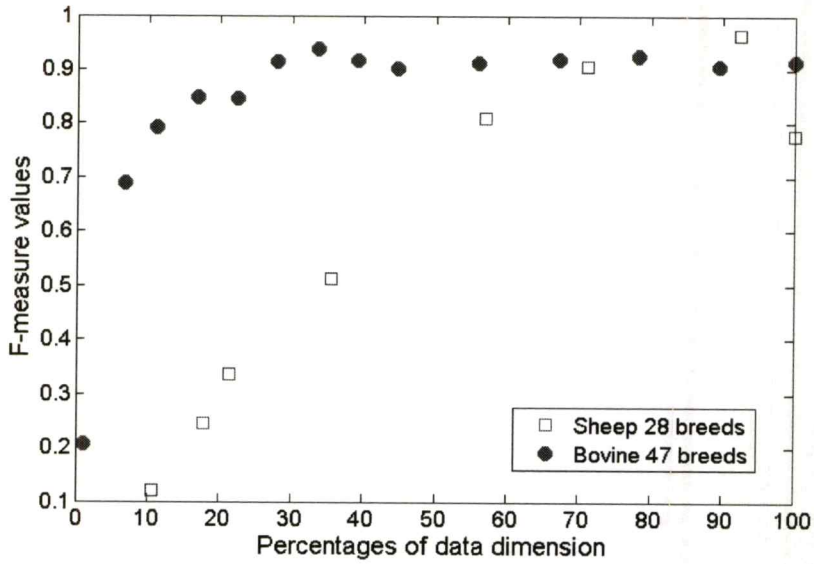


Figure 5.3 F-measures of the HiClust-ipPCA clustering results vs. percentages of data dimensions.

We also investigate the effect of performing the PCA-correlated feature selection procedure from Section 3.2.1 on clustering quality. The HiClust-ipPCA algorithm is applied to both bovine and sheep datasets with varying numbers of data dimensions (number of SNPs). The F-measures of the clustering results in comparison with the given genetic labels are shown in Figure 5.3. As expected, the F-measure

value increases when the data dimension increases, because there is more information in the data matrix. For bovine dataset, the clustering quality stabilizes when the number of SNPs reaches approximately 35% (15,000 SNPs). In sheep dataset, the clustering quality keeps increasing with data dimensions. This result corresponds to the original data dimensions. The bovine dataset has 44,706 SNPs, so it is likely that some SNPs are either correlated or noise. On the other hand, the sheep dataset only has 1,406 SNPs, so all SNPs are informative.

5.3 The iNJclust algorithm performances

5.3.1 The iNJclust algorithm with topological stopping criterion

Three large, complex real SNPs datasets are used to evaluate the performance of the iNJclust algorithm with *UT1* stopping criterin. The first dataset contains 243,855 SNPs from 27 human populations, having 554 individuals [36]. The second comes from 47 breeds of bovine, having 1089 individuals and 44,706 SNPs in all [34]. The last dataset has 392 individuals and 1,406 SNPs from 28 breeds of sheep [35].

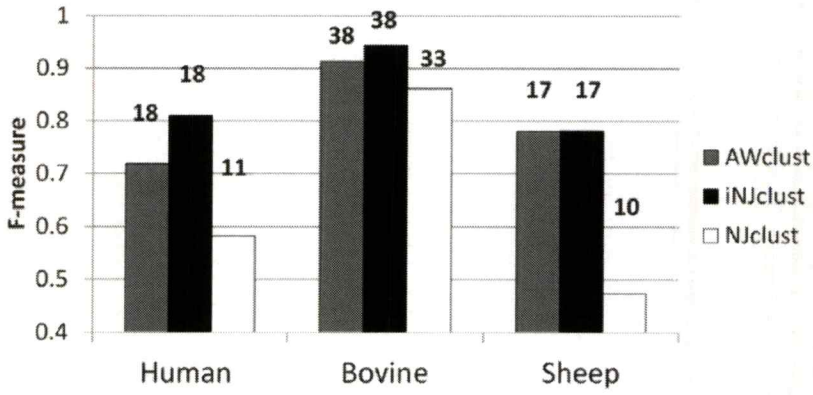


Figure 5.4 F-measure values of the AWclust, iNJclust, and NJclust algorithms on bovine 47 breeds, sheep 28 breeds, and human 27 population datasets.

The F-measures from the AWclust, NJclust, and iNJclust algorithms are shown in Fig. 5.4 along with the estimated numbers of clusters at the top of the bars. The superiority of the iNJclust algorithm over the NJclust and AWclust algorithms can be seen in the F- measure values. For comparison purpose, the number of clusters for the AWclust algorithm is set to equal the estimated cluster numbers from the

iNJclust algorithm in all datasets. Recall that if we do not set the cluster numbers for the AWclust algorithm, it will cluster the data to at most 16 clusters, which are much less than the actual cluster numbers.

For human dataset, the labels are self-reported, so it is questionable if the label corresponds to the real population that person belongs. For example, a family may relocate to a different region of the country. Hence, the F-measure value for the human dataset in Fig. 5.4 appears to be lower than the bovine and sheep dataset. To better analyze the clustering results, we examine the cluster pie charts, as depicted in Fig. 5.5.

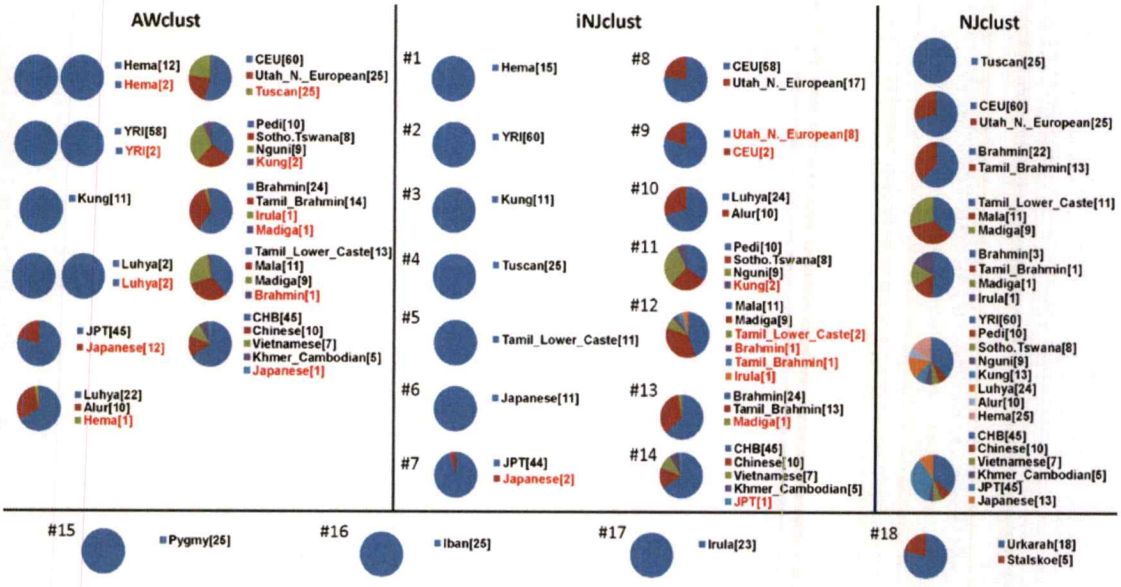


Figure 5.5 Clustering results of the human dataset. Bottom panel: Similar clusters among the three algorithms.

It is observed that the iNJclust algorithm assigns more pure clusters, although all three algorithms provide some clusters with mixed populations. This is unavoidable because some population groups, e.g., Indians (Mala, Madiga, Tamil, Brahmin), east Asians (Chinese, Vietnamese, Khmer), are very similar genetically. However, the AWclust clustering result contains many more outliers (highlighted in red), which are corrected in the iNJclust clusters. The cluster assignment from the NJclust algorithm is the worst. This is because the NJ tree of the original dataset may have an incorrect topology, as illustrated in Fig. 5.6. Without the tree reconstruction

step, the use of an incorrect NJ tree by the NJclust results in a bad clustering performance. We illustrate two genetic labels which have initially been separated in the topology of the NJ tree at the first iteration (main Fig. 5.6). The iNJclust algorithm is able to correct the tree topologies of these labels at the subsequent iterations (Fig. 5.6 insets) and eventually assign them to the right clusters.

Another advantage of the iNJclust algorithm is its ability to infer genetic similarities between clusters from the order at which each cluster is bifurcated in the iterative clustering process. This can be seen in Fig. 5.7 where the cluster numberings correspond to the assigned clusters in Fig. 5.5. African individuals are first separated from the rest because they are the most distinct subpopulation. Then East Asians are separated out from Europeans and Indians. At terminal nodes, genetically similar subpopulations are finally differentiated.

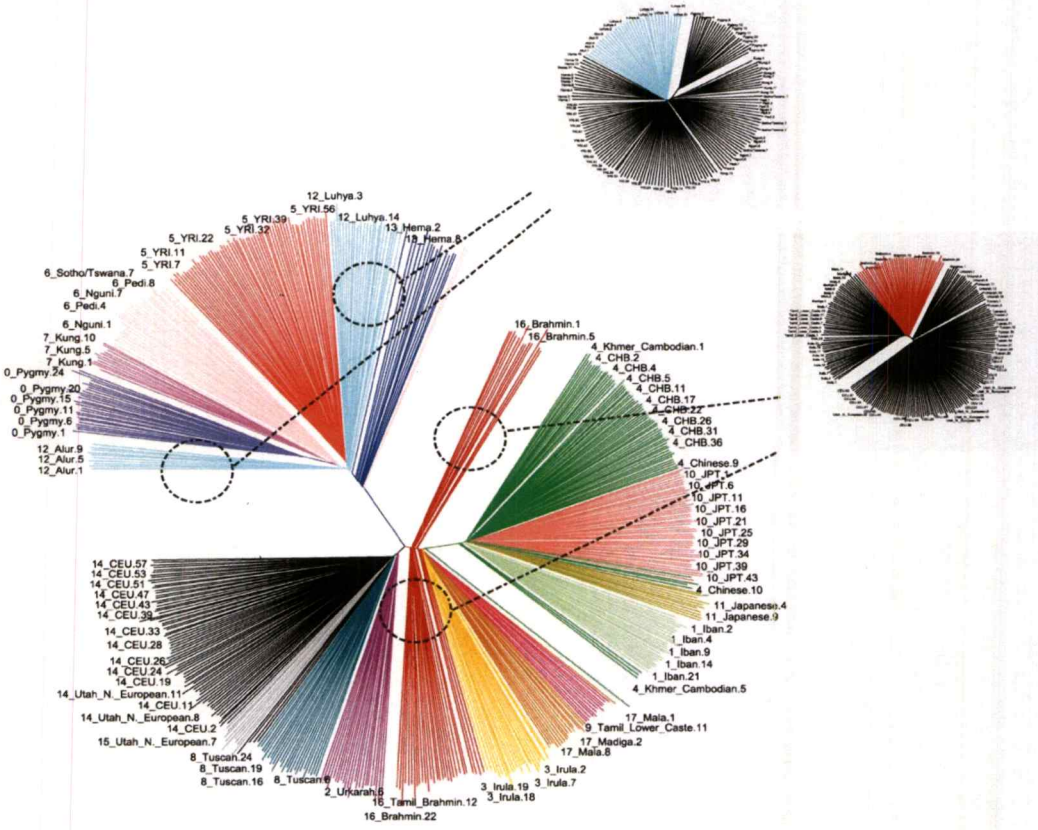


Figure 5.6 NJ trees with cluster labels of the Human 27 populations dataset.

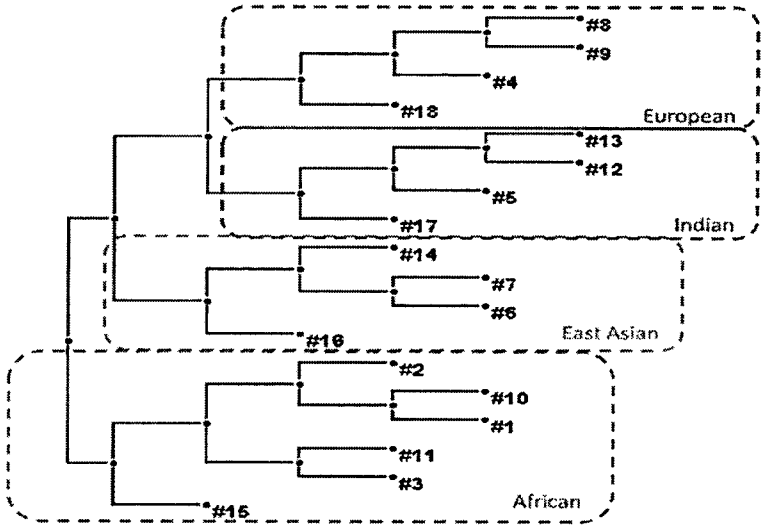


Figure 5.7 Population tree of the Human 27 populations dataset.

5.3.2 The iNJclust algorithm with fixation index difference stopping criterion

In this section we explore the performance of the iNJclust algorithm with the ΔF stopping criterion. Simulated data is useful in investigating the efficacy of the algorithm since the inherent number of subpopulations in a real genotype dataset is usually unknown or self-reported.

5.3.2.1 Simulation datasets

We use two simulated datasets to explore the effects of evolution time and number of subpopulations on the optimal value of ΔF and the robustness of the clustering results. The datasets are generated from the GENOME tool [37] using the following parameters and tree files:

Dataset 1:

```
-pop 20 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 -c 20 -s 500 -N UD1model.txt
```

Dataset 2:

```
-pop 10 330 150 60 60 60 300 60 60 60 60 -c 20 -s 500 -N UD2model.txt
```

The first dataset contains 20 clusters $S = \{S_1,...S_{20}\}$ of 60 individuals each (for a total of 1,200 individuals), and 10,000 SNPs per individual. The second dataset contains 1,200 individuals separated into 10 clusters $S_1,...S_{10}$ of varying sizes ranging

from 60 to 330 individuals per cluster, also genotyped at 10,000 SNPs. Their population history trees are shown in Fig. 5.8 where the branch lengths represent the evolution time of subpopulations in terms of the number of generations they evolve.

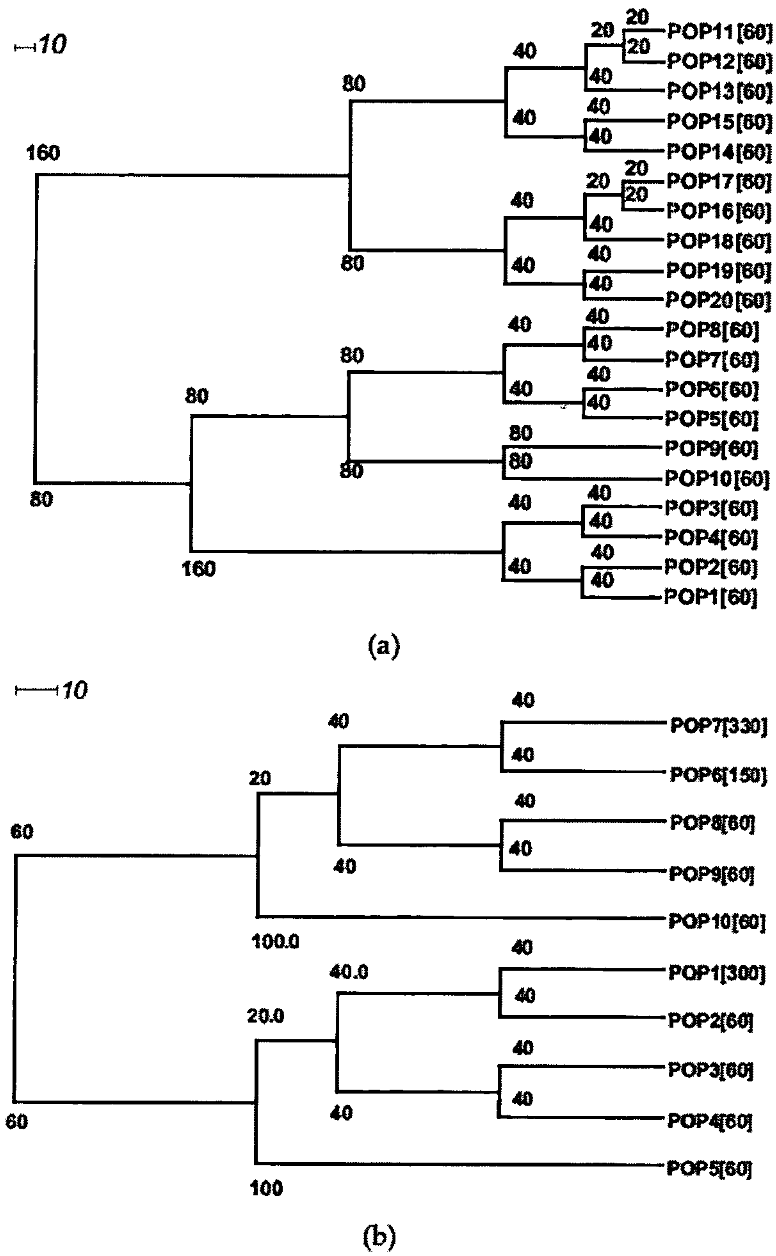


Figure 5.8 Simulated population history trees. The edge weights represent the generations of subpopulations. (a) Dataset 1. (b) Dataset 2.

An optimal values of the ΔF stopping criterion for clustering of Dataset 1 and 2 are determined by scanning the iNJclust algorithm over a range of possible ΔF values ranging from 10^{-5} to 10^{-1}, i.e., the differences of 10% to 0.001% in the fixation indices. We compare the iNJclust clustering results $C = \{C_1, ...C_P\}$ at each value of ΔF to the ground truth S and compute the F-measure [28]. The higher the F-measure values, the closer they are to the simulated clusters. An F-measure value of 1 occurs when the iNJclust clustering result is exactly the same as the true simulated clusters. Figure 5.9 depicts the F-measure value as a function of ΔF threshold value for both simulated datasets. It is observed that if the ΔF threshold is too high, the iNJclust process undersplits the clusters, where as a too-low value of the threshold oversplits the cluster. The step-like behavior of the F-measure value also implies that the optimal value for ΔF is not a single point but rather a range, making selecting an appropriate value for the threshold slightly flexible. From the graph, we selected the threshold of 0.001 for Dataset 1 and 0.003 for Dataset 2.

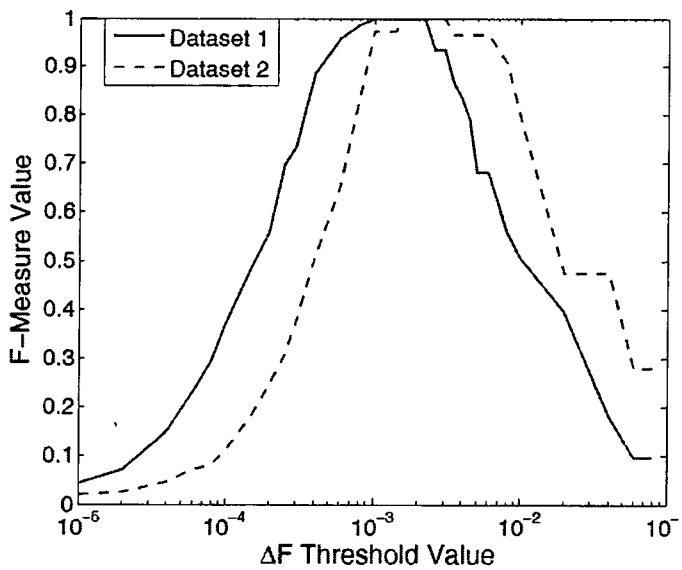


Figure 5.9 F-Measure as a function of ΔF threshold value.

We investigate how two other factors—the relationship between evolution time, which manifests in the number of generations in the simulation, and the number of subpopulations—may affect the optimal value of the ΔF threshold. We vary the genetic distance between subpopulations by simulating data ranges between 40 to 500 generations (corresponding to approximately 80 to 10,000 years

of genetic evolution). We also vary the number of subpopulations from 2 to 64 subpopulations in the data. The result is depicted in Fig. 5.10. Data with larger number of generations are further apart genetically, so the threshold increases with generations as expected. On the other hand, when the number of subpopulations increases the threshold value decreases, since more resolution is needed to differentiate between subpopulations. For large, complex dataset, the default threshold value for the iNJclust algorithm is set at 0.003. The threshold should be adjusted if some prior information about the data is available.

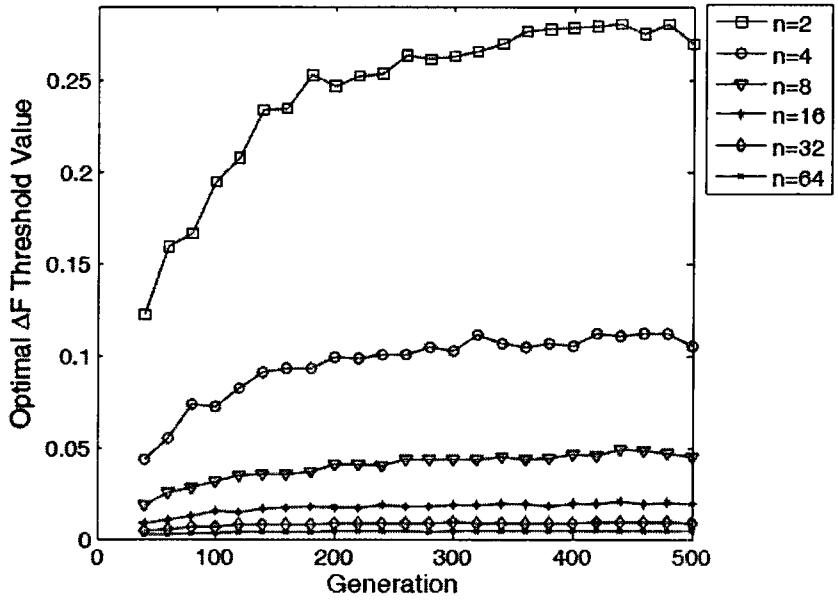


Figure 5.10 Optimal ΔF threshold values as a function of the number of generations, for varying number of subpopulations.

We infer the relationships among populations from the hierarchical population trees generated by the iNJclust algorithm for the two simulated datasets, as depicted in Fig. 5.11 and 5.12. The branch lengths on the trees correspond to the calculated ΔF values at each iteration. Observe that the ΔF value decreases at later iterations. The terminal nodes of the tree within the figures also show the iNJclust individual assignments, where the individual are labeled by their true cluster number. The numbers in the square brackets represent the number of individuals within each cluster.

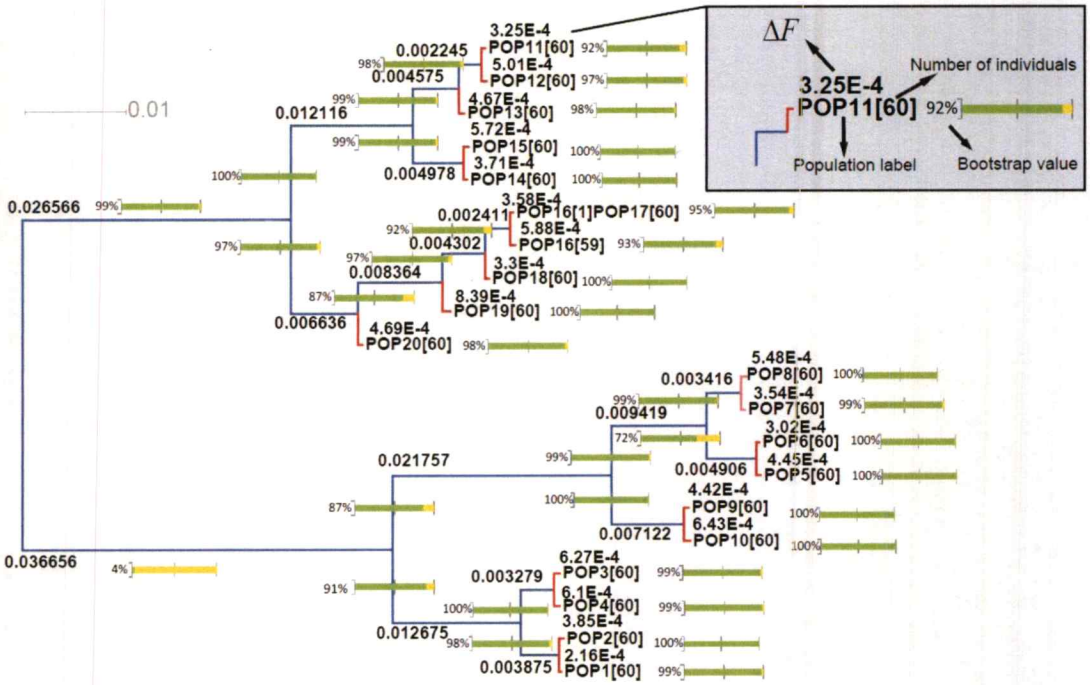


Figure 5.11 Hierarchical populations tree of dataset 1 generated by the iNJclust algorithm. The threshold is set at 0.001.

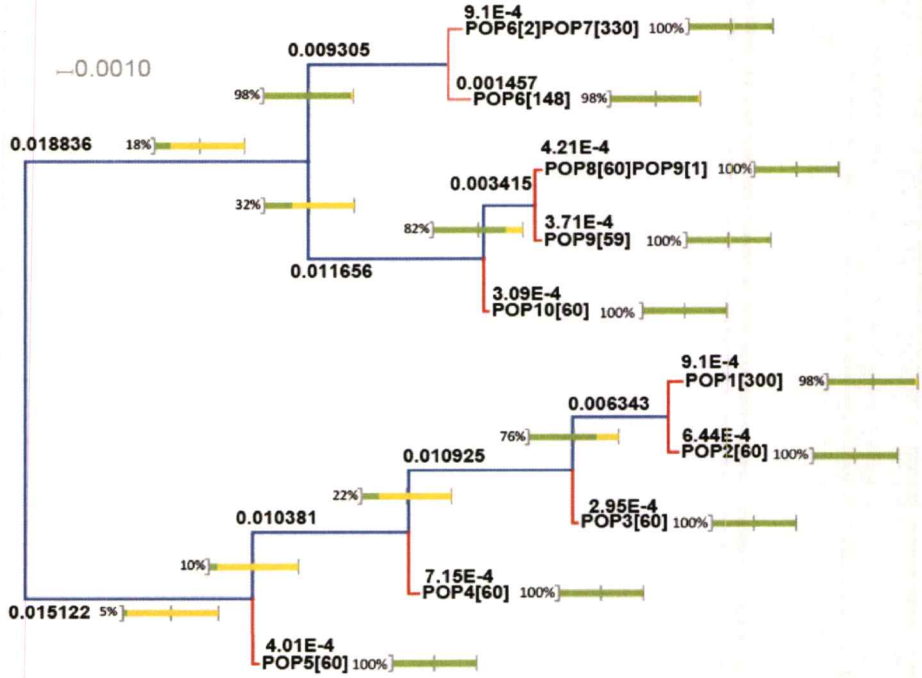


Figure 5.12 Hierarchical populations tree of dataset 2 generated by the iNJclust algorithm. The threshold is set at 0.003

A careful examination of the individual assignment results in Fig. 5.11 confirms that the iNJclust algorithm is able to assign most individuals to their respective cluster correctly. For Dataset 1, one individual from population POP16 is grouped with POP17. This is possible since the pair of subpopulations POP16 and POP17 only differs by 20 generations; they are closely related subpopulations in the dataset. The total individual assignment is 99.92% correct. The POP19 branching is in a different order from the simulated tree, which may be the consequence of the bisecting method we employ in the algorithm. Nonetheless, it does not affect the overall tree topology. Although the branch lengths on the top-half and bottom-half of the tree are not exactly identical, they follow a similar trend.

In Dataset 2, we vary the number of individuals in each subpopulation to illustrate the ability of our algorithm to handle varying cluster sizes. Clustering results in Fig. 5.12 show that the algorithm remains effective in this situation. The relationships among subpopulations are correctly translated to the iNJclust tree final results with slight variation in the tree topology. The individual assignment error is 3 individuals out of 1200 (0.25%).

Additionally, we employ the bootstrapping method in order to validate the consistency of the iNJclust results. Each simulated dataset is resampled 100 times with replacement to obtain bootstrap datasets with 400 individuals. We count how many times the same subtree topology, i.e., that subtree contains all of the same clusters, is generated at each hierarchy of the tree. The bootstrapping results are depicted in Fig. 5.11 and 5.12 as the slider bars on top of each tree branch. It is discovered that most of the branches have high bootstrap percentages; hence the iNJclust algorithm is fairly robust. Some branches near the root of the NJ tree may receive low bootstrap values, meaning that some terminal nodes may be on different branches early on, but eventually the individuals are split into their appropriate final clusters. To explain this phenomenon we adopt the Admixture method [38] to observe the ancestry distributions of different populations. The admixtures of Dataset 1 and Dataset 2 are depicted in Fig. 5.13. Different colors mean that the individuals are not related, i.e., they do not come from the same ancestry. Similar colors mean that the populations related genetically. If a subtree contains two groups of populations with totally distinct ancestries, for example {POP1, POP2, POP3, POP4} (red color of Dataset 1 on Fig. 5.13) and {POP5, POP6, POP7, POP8,

POP9, POP10} (green color of Dataset 1 on Fig. 5.13), either group can be swapped to a top branch of the NJ tree. If this happens, the bootstrap method views the tree topology as being different and the bootstrap score for that bootstrap dataset is 0.

In contrast, {POP11, POP12, POP13, POP14, POP15} (purple) and {POP16, POP17, POP18} (blue) also have two distinct ancestries. However there is {POP19, POP20} which has both purple and blue ancestries patterns. Hence, these two populations connect {POP11, POP12, POP13, POP14, POP15} and {POP16, POP17, POP18}, so they always appear on the same tree branch. Therefore, the top half of the tree for Dataset 1 has very high bootstrap values.

More evidence is depicted in the admixtures of Dataset 2 which has many low values of bootstrap. The ancestry patterns of POP6 and POP7 are similar. So they always have the same parent node, resulting in their root branch having a high bootstrap value. In contrast, {POP5} has two ancestries (blue and purple), so this population is related to the top half of the tree and often gets placed there. POP5 may also influence {POP1, POP2} to swap to the top branch as well since they have similar (purple) ancestry. This also causes the low bootstrap value for the root branch. In summary, a bootstrap value of our method relies on the different ancestry patterns within subtrees. Furthermore, when resampling, individuals from certain subpopulations may not be presented enough. Consequently, the branch representing that subpopulation is not generated, making the topology different.

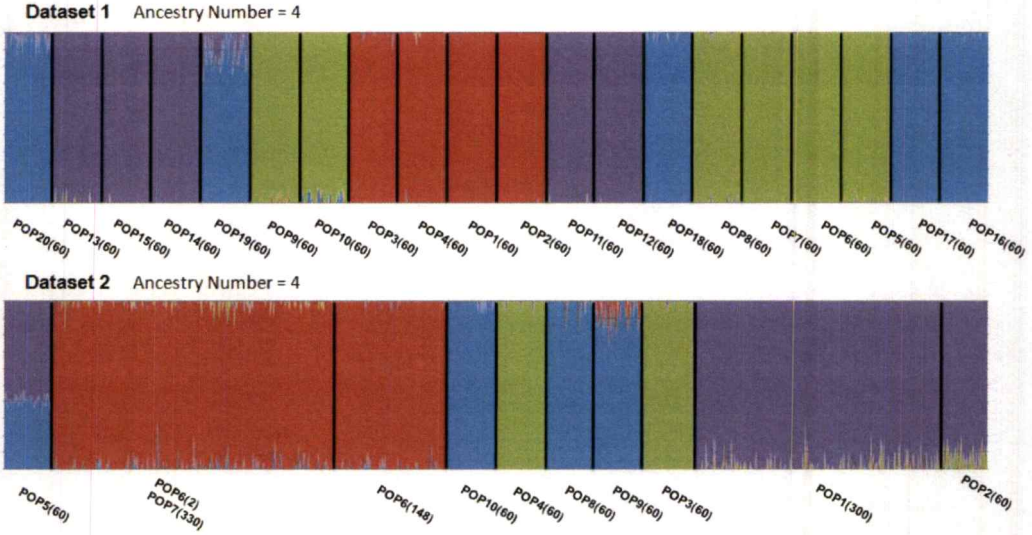


Figure 5.13 Admixture results of simulated Dataset 1 and 2. The ancestry parameter is set to 4.

5.3.2.2 Real datasets

We test the iNJclust algorithm on four large datasets with different complexities. The first three datasets are similar to the previous subsections. The first dataset is from 47 breeds of 1,089 bovines [34], genotyped at 44,706 SNPs. The second dataset is a 28-breed sheep dataset [35], which contains 392 individuals and 1,046 SNPs. The third dataset comprises of 27 human populations [36] for a total of 554 individuals and 243,855 SNPs. We add one additional human dataset containing 245 individuals and 54,794 SNPs from 13 tribes in Thailand [39]. Guided by the optimal thresholds from simulated Dataset 1 and 2, we choose the ΔF thresholds of 0.001 for the bovine 47 breeds, sheep 28 breeds, and human 27 populations since they seem similar in their complexities, and use the threshold of 0.003 from Dataset 2 for Thai 13 tribes datasets.

Again, we compare the clustering performance of the iNJclust algorithm with the AWclust and NJclust algorithms in terms of the F-measure values only for the datasets in which we have confidence in the labels, namely the two simulated datasets and the animal datasets.

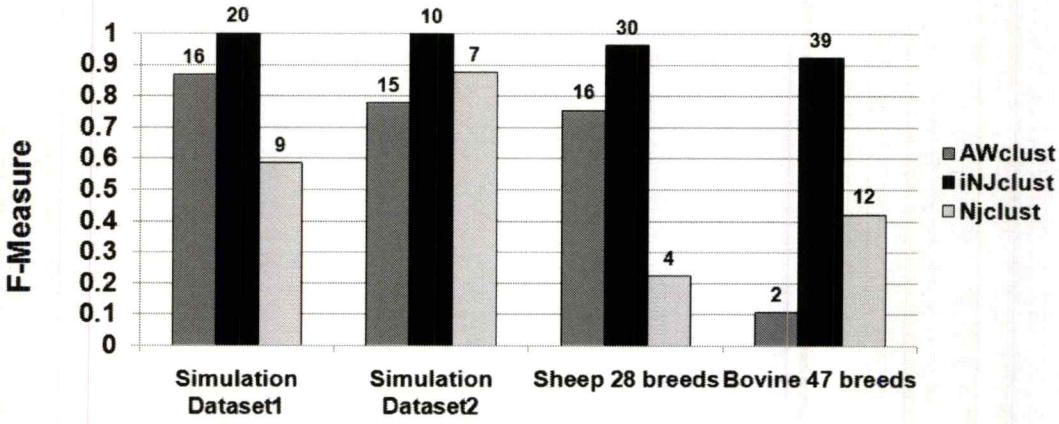


Figure 5.14 F-measures of the AWclust, iNJclust, and NJclust algorithms on simulated and animal datasets. The estimated numbers of clusters are displayed on top of the bar graphs.

The F-measures depicted in Fig. 5.14 imply that the iNJclust algorithm produce the best clustering results. The individual assignments of AWclust is worse than the assignments of iNJclust. We believe that NJ tree clustering is more

appropriate in distinguishing between subpopulations than hierarchical clustering for genetic data. The NJclust clustering result is much more erroneous than iNJclust and AWclust. This illustrates the benefit of reconstructing the NJ tree at each iteration. In addition, the computational time of the iNJclust algorithm is superior to the computational time for the AWclust algorithm. It would take longer times for the AWclust algorithm, which has complexity $O(n^6)$ due to use of Gap statistic [40], while iNJclust has complexity of only $O(n^3)$ to detect the number of clusters.

To alternatively verify the clustering results of the iNJclust algorithm on the human datasets, we again compare the clustering results with the admixture patterns. It is customary to plot the admixture patterns for different number of ancestries K and select the smallest possible value of K with the most distinctive patterns. The plots of admixtures for the human 27 populations dataset for K=2 *to* K=10 are depicted in Fig. 5.15. Figure 5.16 depicts the admixture ratios with K=2 *to* K=8 for the Thai 13 tribes dataset. We choose the best admixture parameters K=8 for the human 27 populations dataset and K=6 for the Thai 13 tribes dataset for our comparison purpose.

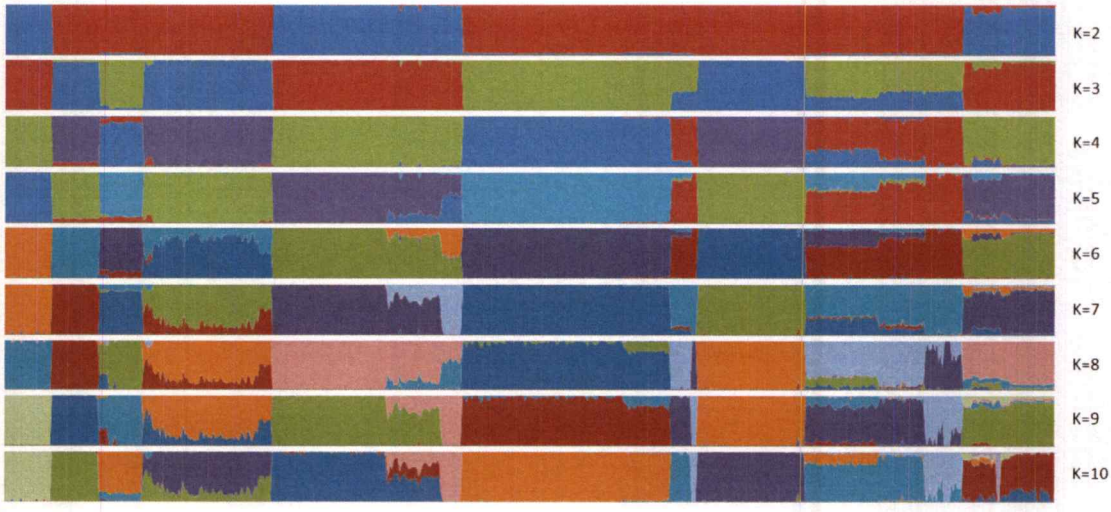


Figure 5.15 Admixtures of the human 27 populations dataset, with the ancestry parameter K=2 to K=10.

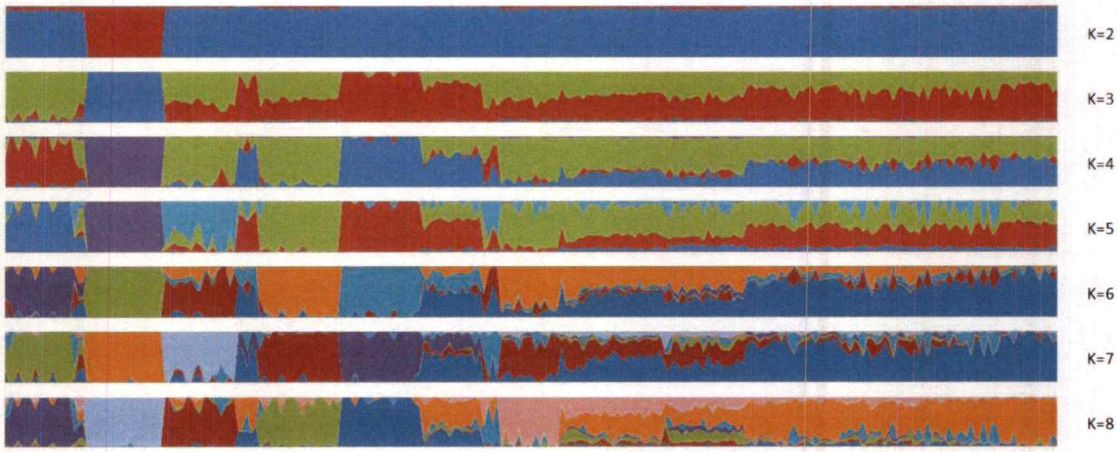


Figure 5.16 Admixtures of the Thai 13 tribes dataset, with the ancestry parameter K=2 to K=8.

In Fig. 5.17 and 5.18, we compare the iNJclust clustering results to the admixture patterns for the two human datasets. Each panel of admixture patterns separated by the black line is one cluster assigned by the iNJclust algorithm. The corresponding self-reported labels of individuals in each cluster are displayed below the panels, with the number of individuals from that label shown in the bracket. The admixture results are in very good agreement with the iNJclust individual assignments. That is, each assigned iNJclust cluster has distinct admixture patterns.

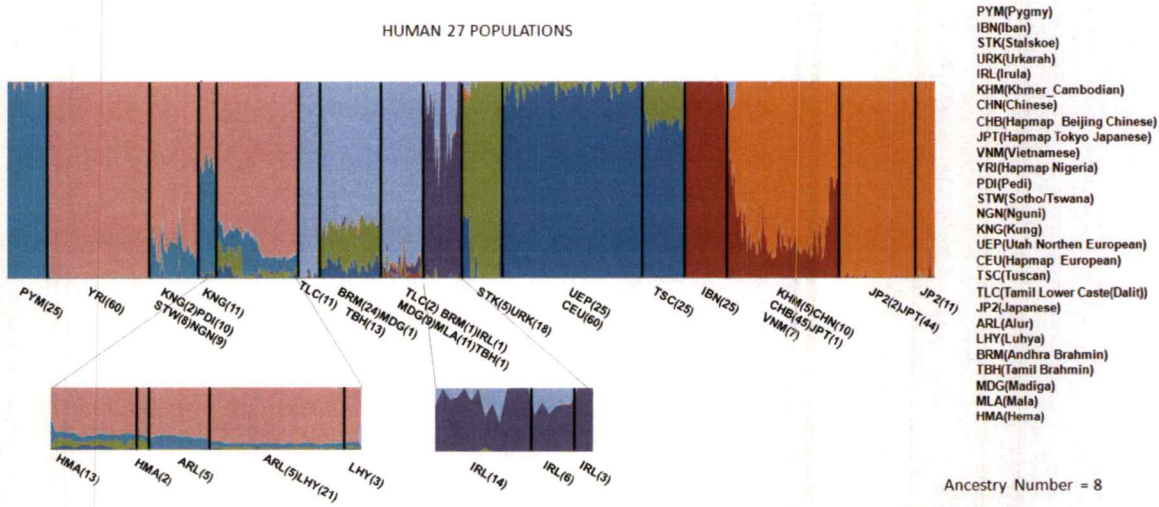


Figure 5.17 Comparison between admixture ratios (K=8) and the iNJclust clusters of the human 27 populations dataset.

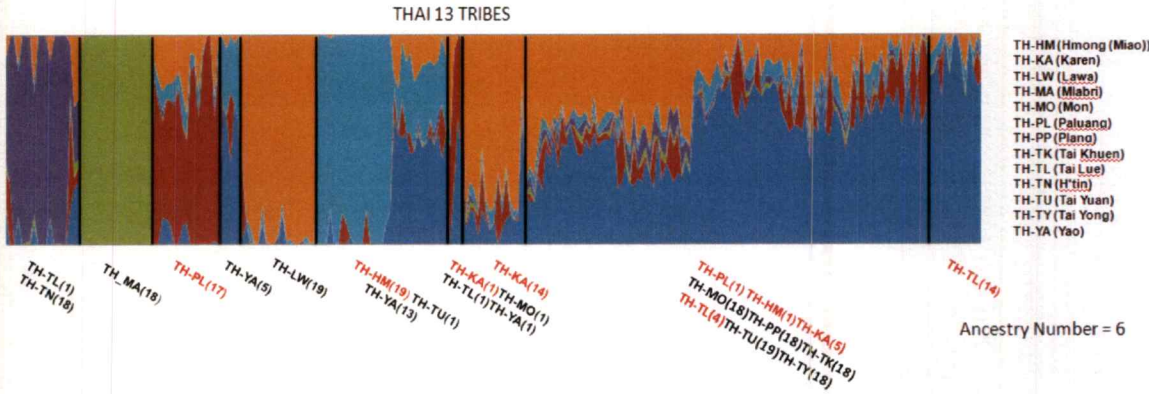


Figure 5.18 Comparison between admixture ratios (K=6) and the iNJclust clusters of the Thai 13 tribes dataset.

For both datasets, it is observed that there are many given labels that do not correspond to the genetic patterns. For example, in Fig. 5.17 subpopulation {UEP, CEU}, {STK,URK}, and {KHM,CHN,CHB,JPT,VNM} clusters contain individuals with various subpopulation labels, but they have similar admixture patterns. The iNJclust algorithm is able to cluster them into the same cluster. In contrast, individuals from KNG and HMA subpopulations, though carrying the same labels, are assigned to different clusters by the iNJclust algorithm. Their admixture patterns confirm their genetic differences. Similar results are observed in Fig. 5.18. For instance, members of the TH-PL, TH-HM, and TH-KA tribes (denoted in red) are assigned by the iNJclust algorithm according to their genetic differences into different groups of similar admixture patterns. Using these labels to calculate the F-measure would give the low F-measure values, which does not necessarily reflect the actual efficient clustering performance of the iNJclust algorithm in human datasets.

Chapter 6
Conclusion

6.1 Summary

This thesis applies graph-theoretic clustering, which converts a traditional clustering problem to a graph partitioning problem, to develop two iterative algorithms for clustering genotypic data. The graph-based algorithms can efficiently handle complex cluster shapes and provides excellent clustering results in terms of the estimated number of clusters (or subpopulations), individual assignments, and population tree representing genetic relationship between subpopulations.

For the first part of this thesis, we have proposed a modification to the clustering module of the EigenDev-ipPCA algorithm, which suffers with individual assignment errors from fuzzy c-mean clustering. We employ Ward's minimum variance hierarchical tree clustering and illustrates that the hierarchical clustering method provides better clustering results than fuzzy c-mean for complex genotype datasets. The improved algorithm is called the HiClust-ipPCA algorithm. It is particularly good at reducing outliers in each cluster. The accuracy of individual assignments and estimated number of clusters also increase significantly. Furthermore, we study the effect of using feature selection for dimensional reduction of the data. We find that the use of the feature selection technique is effective for reducing data dimension and increasing computational efficiency. It is observed that for high-dimensional datasets, we do not need to use all dimensions to achieve the best clustering results. From our example, only 33% of the available dimensions is enough to reach the same clustering quality. Nevertheless, for smaller datasets we have to use all of dimensions for clustering in order to reach the best clustering quality because all dimensions are informative.

In the second part of the thesis, we propose an unsupervised graph-based clustering algorithm called *iNJclust* which is capable of inferring three main quantities of interest in genetic data: estimated number of clusters, individual assignments, and

genetic relationships between subpopulations in the form of a population tree. We utilize the most widely used phylogenetic tree called a neighbor-joining tree to differentiate between subpopulations by their real genetic relationships. Similar to the HiClust-ipPCA algorithm, we adopt an iterative method to achieve high clustering resolution. Our iNJclust algorithm can differentiate arbitrary complex-shape clusters. The iterative tree reconstruction process of the algorithm is also essential in identifying the correct tree topologies and produce accurate clustering result. Moreover, we have proposed two measures for determining cluster homogeneity. One measure is ad hoc; it is based on the *UT1* topology pattern of the tree. The second measure ΔF based on the difference of the fixation indices at successive iterations has a solid mathematical proof.

The performance of the HiClust-ipPCA algorithm and the iNJclust algorithm have been tested extensively against existing clustering algorithms, namely the EigenDev-ipPCA algorithm and the AWclust algorithm, using both simulated and real datasets. Our algorithms are able to discriminate arbitrary complex-shape clusters more efficiently even though there are varying in size and numbers of populations. We investigate more evidence regarding the iNJclust performance by looking at the population structure inferred by the ancestry patterns. The iNJclust clustering result closely follows intrinsic ancestry patterns within the data.

6.2 Contribution

The research contribution in this thesis to the fields of bioinformatics, pattern recognition, and population studies is the utilization of graph-theoretic approaches to improve a genotypic clustering framework, such that it can cope with arbitrary cluster shape and increase its clustering accuracy. We substitute fuzzy c-means for a Ward's minimum variance hierarchical tree in the ipPCA framework. The accuracy of the algorithm increases, meaning that a hierarchical clustering approach can handle arbitrary-shape populations better than the model-based algorithms. Our investigation on the dimensional reduction of the data by feature selection suggests a way to increase the computational efficiently of an algorithm, since we find that

we do not need to use all dimensions in high-dimensional dataset. In the iNJclust algorithm, we believe to be the first to realize the usefulness of the NJ tree as a clustering technique. We discover that the tree rebuilding step is crucial for an iterative method. Moreover, we succeed in producing a hierarchical tree that represents the genetic relationship between subpopulations. We also introduced two novel methods to determine homogeneity of clusters and stop the iterative process of our algorithms. The first one depends on a particular tree topology called the *UT1* pattern. It contains no unknown parameter. The second one uses the behavior of the fixation index, which a real genetic distance among subpopulations. A threshold is needed, although selecting appropriate value a threshold is fairly flexible. Finally, our proposed algorithms is extendable for clustering other type of high dimensional signals such as corpus text datasets, images, or other biosignals simply by constructing an appropriate type of graph before clustering.

6.3 Future Works

Beside the clustering accuracy and relationships among clusters, another interesting topic is the dominant characteristics of each subpopulation. This investigation is key for disease association studies. We can use this information to distinguish between disease populations and normal populations, which may lead to the discovery of particular genes causing the disease, laying the foundation for disease protection and treatment. Therefore, finding the informative features is an open research topic worthy of future exploration. In addition to the hierarchical approach, there is another graph-theoretic method called spectral clustering [41]. It uses the eigenvalues of Laplacian matrix to split vertex groups and is able to perform dimensional reduction before clustering. This method may help generalize our framework to any type of multidimensional data.

References

[1] Grygorash O., Zhou Y., and Jorgensen Z. "Minimum spanning tree based clustering algorithms" **In Proc. of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence,** ICTAI '06, Washington, DC, USA , 2006. pp.73–81

[2] Xu Y., Olman V. and D. Xu. "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees" **Bioinformatics,** vol.18 no.4, 2002. pp.536–545

[3] Johnson S. "Hierarchical clustering schemes" **Psychometrika,** vol.32, 1967 pp.241–254

[4] Jain A.K., Murty M.N. and Flynn P.J. "Data clustering: A review" **ACM Computing Surveys,** vol.31, Sep. 1999. pp.264-323

[5] Intarapanich A. and et.al. "Iterative pruning PCA improves resolution of highly structured population" **BMC Bioinformatics,** vol.10, 2009. pp.382

[6] Limpiti T. and et.al. "Study of large and highly stratified population datasets by combining iterative pruning principal component analysis and structure" **BMC Bioinformatics,** vol.12 no.1, 2011. pp.255

[7] Saitou N. and Nei M. "The neighbor-joining method: a new method for reconstructing phylogenetic trees" **Oxford Molecular Biology and Evolution,** Vol.4 no. 4, 1987. pp. 406-425

[8] Gascuel O. and Steel M. "Neighbor-joining revealed" **Oxford Molecular Biology. and Evolution,** vol.23 no.11, 2006. pp.1997–2000

[9] Ward JH. Jr. "Hierarchical grouping to optimize an objective function" **Journal of the American Statistical Association,** vol. 58, 1963. pp. 236–244

[10] Paschou P., Lewis J., Javed A. and Drineas P. "Ancestry informative markers for fine-scale individual assignment to worldwide populations" **Journal of Medical Genetics**, vol.47 no.12, 2010. pp. 835–847

[11] Barreiro L.B., Laval G., Quach H., Patin E. and Quintana-Murci L. "Natural selection has driven population differentiation in modern humans." **Nature Genetics** vol.40, 2008 pp.340–345.

[12] Carlson B. "SNPs — A Shortcut to Personalized Medicine" **Genetic Engineering & Biotechnology News (Mary Ann Liebert, Inc.)**, vol.28, July 2008 pp.12.

[13] MCQUEEN J. "Some methods for classification and analysis of multivariate observations" **In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability**, CA. USA. 1967. pp. 281-297

[14] ZADEH L. A. "Fuzzy sets." **Inf. Control.,** Vol8, 1965. pp. 338-353.

[15] DEMPSTER A.P., LAIRD N.M., and RUBIN, D.B. "Maximum likelihood from incomplete data via the EM algorithm." **J. Royal Stat. Soc. B.**, Vol39, no.1, 1977 pp.1-38.

[16] KOHONEN T. **Self-Organization and Associative Memory**. 3rd Edition. Springer information sciences series. Springer-Verlag, New York, NY. USA, 1989.

[17] Gao X. and Starmer J. "AWclust: point-and-click software for non-parametric population structure analysis" **BMC Bioinformatics**, vol.9, 2008. pp.77

[18] Schaeffer S.E. "Graph clustering" **Elsevier Computer Science Review**, vol.1, 2007. pp.27-64

[19] Pearson K. "On Lines and Planes of Closest Fit to Systems of Points in Space" **Philosophical Magazine 2**, Vol.6, 1901. pp. 559–572.

[20] Gao X, and Martin ER. "Using Allele Sharing Distance for Detecting Human Population Stratification" **Human heredity**, Vol.68, 2009. pp.182-191

[21] Wright S. **Evolution and the genetics of populations.** Chicago, USA, University of Chicago Press, 1984

[22] Hartl DL., Clark AG. **Principles of population genetics** 4th Edition. Massachusetts. USA. Cambridge university press, 1997.

[23] Holsinger K.E. and Weir B.S. "Genetics in geographically structured populations: defining estimating and interpreting F_{st}" **Nature Reviews Genetics**, vol.10 no.9, Sep. 2009. pp.639-650

[24] Prim R.C. "Shortest connection networks and some generalizations" Bell System Technical Journal, vol.36, 1957. pp.1389-1401

[25] Kruskal J.B. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem" **Proceedings of the American Mathematical Society**, vol.7 no.1, Feb. 1956 pp.48-50

[26] Yang Z. and Rannala B. "Molecular phylogenetics: principles and practice" **Nature Reviews Genetics**, vol.13 no.5, May 2012. pp.303-314

[27] Li M., Reilly M.P., Rader D.J. and Wang L.S. "Correcting population stratification in genetic association studies" **Oxford Bioinformatics**, vol. 26 no. 6, 2010. pp.798-806

[28] Chinchor N. "Evaluation metrics" **In Proc.of the 4thMessage Understanding Conf.**, 1992. pp. 22–29

[29] Patterson N., Price AL. and Reich D. "Population structure and Eigenanalysis" **PLoS Genet**, vol.2 no.12, 2006. pp.e190.

[30] Press W.H. and et.al. **Numerical recipes** 3rd Edition. UK. Cambridge university press, 2007.

[31] Dongarra J.J. and et.al. "LAPACK++: A design overview of object-oriented extensions for high performance linear algebra" **In Proc. of the 18th IEEE Int. Conf. on Supercomputing '93.**, Oregon, USA, 1993. pp.162-171

[32] Lawson D. and Falush D. "Population identification using genetic data" **Annu.Rev. Genomics Hum. Genet** vol.13, Jun. 2012. pp.337-361

[33] Huson D.H., Richter D.C., Rausch C., Dezulian T., Franz M. and Rupp R. "Dendroscope: An interactive viewer for large phylogenetic trees" **BMC Bioinformatics**, Vol. 8, 2007 pp. 460

[34] Gautier M., Lalo¨e D. and Moazami-Goudarzi K. "Insights into the genetic history of french cattle from dense SNP data on 47 worldwide breeds" **PLoS ONE**, vol.5 no.9, Sep. 2010. pp.e13038

[35] Kijas J. and et.al. "A genome wide survey of snp variation reveals the genetic structure of sheep breeds" **PLoS ONE**, vol.4 no.3, Mar. 2009. pp.e4668

[36] Xing J. and et.al. "Fine-scaled human genetic structure revealed by snp microarrays" **Genome Res**, vol.19 no.5, 2009. pp.815–25

[37] Liang L., Zöllner S. and Abecasis G.R. "GENOME: a rapid coalescent-based whole genome simulator" **Oxford Bioinformatics**, vol.23 no.12, 2007 pp.1565-1567

[38] Alexander D.H., Novembre J. and Lange K. "Fast Model-based Estimation of Ancestry in Unrelated Individuals" **Genome Research**, vol.19, 2009. pp.1655-1664

[39] The HUGO Pan-Asian SNP Consortium "Mapping Human Genetic Diversity in Asia" **Science**, vol.326 no.5959, Dec. 2009. pp.1541-1545

[40] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 63, no. 2, pp. 411–423, 2001.

[41] Luxburg U. "A tutorial on spectral clutering" **Statistics and Computing**, vol.17 no.4, 2007. pp.395-416

[42] Jensen JLWV. "Sur les fonctions convexes et les inégalités entre les valeurs moyennes." Acta Math. Vol.30, pp.175-193, 1906.

Appendix A.

Jensen's inequality

Let $f(x)$ be a concave up function which has value on a real interval $[a,b]$. The second derivative of a concave up function is always a positive real number or $f''(x)>0$. In this thesis, we are interested in $f(x)=-x(1-x)$ function. Since its second derivative is $f''(x)=2$. Therefore, $f(x)=-x(1-x)$ is a concave up function.

The Jensen's inequality [42] states that, if $\lambda_1,...,\lambda_n$ are positive number, $\sum_{i=1}^{n}\lambda_i=1$ and $f(x)$ is a concave up function, then

$$\sum_{i=1}^{n}\lambda_i f(x_i)\geq f\left(\sum_{i=1}^{n}\lambda_i x_i\right) \qquad \text{(A1).}$$

In the specific case of $n=2$, we have

$$\lambda_1 f(x_1)+\lambda_2 f(x_2)\geq f(\lambda_1 x_1+\lambda_2 x_2) \qquad \text{(A2).}$$

Where $\lambda_1,\lambda_2>0$, and $\lambda_1+\lambda_2=1$,

Appendix B.

Publications

1. Amornbunchornvej C., Limpiti T., Assawamakin A., Intarapanich A., and Tongsima S., "Improved iterative pruning principal component analysis with Graph-theoretic hierarchical clustering", *Proceeding of the 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp.1-4, Phetchaburi, Thailand, 2012, ECTI association.

2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology.

ECTI-CON 2012

16-18 MAY 2012
PHETCHABURI,THAILAND

Author Index | Content

ISBN: 978-1-4673-2024-5
IEEE Catalog Number: CFP1206E-CDR

BACK | ECTI-CON 2012

CONTENT

Parallel and Bio Informatics

Improved iterative pruning principal component analysis with graph-theoretic hierarchical clustering

C. Amornbunchornvej*, T. Limpiti*, A. Assawamakin†, A. Intarapanich† and S. Tongsima†
*Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand
†National Center for Genetic Engineering and Biotechnology (BIOTEC), Pathumthani 12120 Thailand
‡National Electronics and Computer Technology Center (NECTEC), Pathumthani 12120 Thailand

Abstract—**Various unsupervised clustering algorithms have been used to infer population structure in genetic data. The goals are to separate individuals of similar genetic characteristics into clusters and to estimate the number of clusters within each dataset. Among them, a framework called iterative pruning principal component analysis (ipPCA) have been developed. It performs PCA iteratively on subsets of data samples and clusters them using fuzzy c-mean. We believe that the choice of model-based clustering method affects the individual assignments and cluster quality, as well as the estimated number of clusters. Thus, in this paper we introduce a hierarchical tree clustering concept from graph theory, whose performance is independent of cluster shapes, into the ipPCA framework. We also add a PCA-based feature selection technique as a data pre-processing step to reduce data dimension and increase computational efficiency. The resulting algorithm is called HiClust-ipPCA. We illustrate the improved clustering results of the HiClust-ipPCA algorithm using 47-breed bovine and 28-breed sheep datasets.**

I. Introduction

Various unsupervised clustering methods have been applied to genetic data to infer hidden characteristics such as population structures, ancestry information, or number of and relationships between clusters of individuals. One unsupervised clustering framework, which has originally been proposed for detecting population structure in large, complex genotypic datasets, is the iterative pruning principal component analysis or ipPCA [1]. It performs PCA iteratively on subsets of data samples and clusters them using the fuzzy c-mean algorithm. In contrast to other classical techniques where the number of clusters is needed *a priori*, the framework gives an estimated number of clusters and individual assignments as outputs. Although ipPCA is robust when dealing with closely-related clusters that may appear conglomerate and gives reasonable individual assignments, the estimated number of clusters is not accurate. This is due to the use of the Tracy-Widom statistic [2] to identify the stopping point in the iterative process. Subsequently, the second algorithm in the ipPCA framework called EigenDev-ipPCA has been proposed [3], which uses a new stopping criterion called EigenDev. It has been shown to produce better estimates of the number of clusters. Nevertheless, the errors in individual assignments remain. We suspect that these errors originate mainly from the choice of the clustering algorithm. This motivates us to improve the quality of individual assignments in the ipPCA framework by modifying its clustering technique.

Existing clustering methods are usually model-based, i.e., the cluster shape is assumed known. For example, in fuzzy c-mean the data clusters are assumed to be circular in shape. Errors of individual assignments often come from the wrong assumption of the cluster shape. One class of clustering methods which is robust to cluster shape is graph clustering method. By converting a data clustering problem to a graph partitioning problem, it does not depend on cluster shape. Graph clustering has been shown to outperform model-based clustering techniques for complex cluster shapes [4], [5].

In this paper, we introduce the hierarchical clustering concept from graph theory into the ipPCA framework. In addition, we have adopted an unsupervised feature selection method from [6] for reducing the data dimension in order to improve the computational efficiency of our algorithm. To assess the performance of the proposed algorithm, we apply it to bovine and sheep genotype datasets. We evaluate the results in terms of individual assignment errors and cluster quality.

II. Methods

Figure 1 depicts the improved ipPCA framework, where the proposed modifications are highlighted in gray boxes. In the original ipPCA [1], [3], the data sequences are projected onto the PCA subspace, then clustered using fuzzy c-mean. The process iterates until the stopping criterion is reached. For our improvements, the data vectors first enter the feature selection step, where the data dimension is reduced. Then the data is projected onto the principal component axes before clustering. The proposed clustering method is the hierarchical clustering [7], hence we termed the modified algorithm **HiClust-ipPCA** to emphasize the new technique. The details of the modified framework including our proposed changes are given below.

A. PCA-based feature selection

Given an N-dimensional data vector from M individuals, we concatenate all data vectors to form an $M \times N$ data matrix $\mathbf{X}$ as the input. Typically, $M << N$. For feature selection we adopt [6] as a method for reducing data dimension, keeping only dimensions with the most amount of information. This technique is based on the principal components, thus it is readily compatible with the ipPCA framework. The informative rank of the data matrix $\mathbf{X}$ can be computed from the singular value decomposition,

$$\mathbf{X} = \mathbf{USV}^T. \quad (1)$$

978-1-4673-2025-2/12/$31.00 ©2012 IEEE

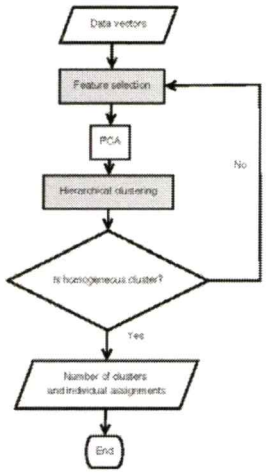


Fig. 1. Diagram of the proposed HiClust-ipPCA algorithm.

The importance score $P(j), j = 1, \ldots, N$ of the j^{th} data dimension is obtained by summing the j^{th} component of the right singular vectors from all individuals,

$$P(j) = \sum_{i=0}^{M} \mathbf{V}(j, i). \tag{2}$$

The larger the value of $P(j)$, the more significant that dimension is. We rank $P(j)$ and choose only the first R informative dimensions. Therefore, the data matrix $\mathbf{X}$ is reduced to $\tilde{\mathbf{X}}_{M \times R}$, where $R \leq N$. In the next step, $\tilde{\mathbf{X}}$ is projected onto its principal components. The homogeneity of these projected data points are determined using an EigenDev measure [3],

$$\text{EigenDev} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left(\log\left(\hat{\sigma}_i^2\right) - \log\left(\sigma_i^2\right) \right)} \tag{3}$$

$$\log\left(\hat{\sigma}_i^2\right) = \log\left(\sigma_1^2\right) + (i-1) \frac{\left(\log\left(\sigma_M^2\right) - \log\left(\sigma_1^2\right) \right)}{(M-1)} \tag{4}$$

and $\sigma_i^2, i = 1, \ldots, M$, are the M eigenvalues of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ ranked in descending order. If the EigenDev value exceeds a user-defined threshold, the data points are considered coming from more than one cluster and we proceed to the hierarchical clustering step. Otherwise, the cluster is considered homogeneous and the algorithm terminates.

B. Graph-theoretic hierarchical clustering

Hierarchical tree [7] is a well-known graph that is used for clustering genetic data. It is constructed by choosing the nearest pair of individuals and merging them to form a new node. The merging continues by joining the next nearest pair of nodes until all individuals are added onto the tree. The typical method in hierarchical clustering is cutting the tree edges from top down to separate the nodes into a desired number of clusters. This does not make biological sense because the order in which the nodes are combined does not always translate to the actual genetic distance between clusters. The number of clusters, K, also needs to be known. We eliminate the need for K by adopting the hierarchical tree into the iterative process, and use Ward's Minimum-variance [8] as a measure for genetic distance between clusters. The distance of Ward's minimum-variance method is the ANOVA sum of squares between pair of clusters,

$$W_{kl} = \frac{||\bar{y}_k - \bar{y}_l||^2}{1/n_k + 1/n_l} \tag{5}$$

where $\bar{y}_k$ and $\bar{y}_l$ represent means of cluster k and l and n_k and n_l represent the numbers of individuals within the clusters, respectively. After we construct the hierarchical tree, we cluster by cutting the edge between two nodes with largest distance W_{kl}, splitting the tree into 2 sub trees. To avoid the effect from outliers, the nodes which has less than 10% of the total number of individuals are excluded from consideration.

III. Results

A. Effect of the hierarchical clustering method

We first evaluate the effect of the new clustering method on the performance of the algorithm by comparing HiClust-ipPCA results with that of the EigenDev-ipPCA algorithm using two genotype datasets. Each data dimension is an integer value of either 0, 1, or 2, which represents the number of minor allele of the single nucleotide polymorphism (SNP). The first dataset comes from 47 breeds of bovine, having 1089 individuals and 44,706 SNPs [9]. The second dataset is a collection of 28 sheep breeds with 392 individuals and 1,406 SNPs [10].

The clustering results for the bovine dataset are shown in Fig. 2. The EigenDev threshold of 0.37 is used in both algorithms. The estimated cluster numbers are 34 for EigenDev-ipPCA and 35 for HiClust-ipPCA. Although the estimated numbers of cluster are lower than 47, it is hardly unexpected since some breeds are extremely close genetically. The results are also comparable to the original results reported in [9]. The cluster assignments in the bottom panel are similar between HiClust-ipPCA and EigenDev-ipPCA. Different cluster assignments are depicted in the top panel of Fig. 2. It is observed that the EigenDev-ipPCA clusters have more outliers. HiClust-ipPCA produces no outlier but there are clusters with mixed labels from breeds that are genetically similar. For example, the outliers from breeds BOR, BRU, and ZMA (highlighted in red) are eliminated by HiClust-ipPCA. The LMS breed which is initially separated into two clusters by EigenDev-ipPCA is combined into one cluster by HiClust-ipPCA.

Figure 3 depicts the clustering results for the 28-breed sheep dataset with the EigenDev threshold of 0.30. The estimated cluster number from EigenDev-ipPCA is 21; HiClust-ipPCA detects 27 clusters out of 28 clusters. It is evident that clusters

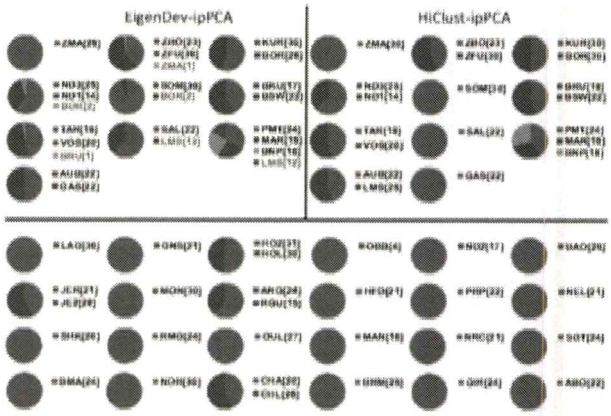


Fig. 2. Clustering results of the 47-breed bovine dataset using an EigenDev threshold of 0.37. Each circle represents a cluster. The colors on the circle represent subpopulations within the cluster. The numbers of individuals are denoted in the square brackets. Top-left panel: EigenDev-ipPCA; Top-right panel: HiClust-ipPCA. Bottom panel: Similar results between the two algorithms.

from EigenDev-ipPCA (left panel) contains many more outliers (highlighted in red). Half of the clusters are also mixtures of at least three breeds. In contrast, the HiClust-ipPCA clusters are mostly pure with exception of only three mixed clusters. This improvement is also evident visually from the graphs, where outliers presented in EigenDev-ipPCA clusters such as COM, USDA_DOS, SOA, TIB are assigned to the appropriate clusters by HiClust-ipPCA.

B. Measurement of clustering accuracy

We measure the performance of HiClust-ipPCA by comparing its clustering result to the given genetic labels using the F-measure from [11]. Let $D_1 = \{D_1^1, D_2^1, D_3^1, \ldots, D_P^1\}$ and $D_2 = \{D_1^2, D_2^2, D_3^2, \ldots, D_Q^2\}$ be two clustering results of dataset D with P and Q clusters, respectively. The similarity of two clustering results D_1 and D_2 is

$$F(D_1, D_2) = \sum_{i=1}^{P} \frac{|D_i^1|}{|D|} \max_j \left(\frac{2 \cdot \frac{|D_i^1 \cap D_j^2|}{|D_i^1||D_j^2|}}{\frac{|D_i^1 \cap D_j^2|}{|D_i^1|} + \frac{|D_i^1 \cap D_j^2|}{|D_j^2|}} \right) \quad (6)$$

where $|\cdot|$ denotes the number of members within a cluster and $|D|$ is the total number of individuals in the dataset. The possible value of F-measure ranges between 0 and 1. $F(D_1, D_2) = 1$ when D_1 and D_2 are exactly the same and zero when they are most dissimilar. We define D_1 as the genetic labels of the dataset and D_2 as the estimated clusters from an algorithm. Hence, the larger F-measure value, the better the performance of a clustering algorithm. The F-measures of the clustering results from EigenDev-ipPCA and HiClust-ipPCA are shown in Table I for both datasets. It is observed that the F-measures for HiClust-ipPCA results are higher than the measures for EigenDev-ipPCA results, confirming the improvement in clustering results achieved after applying our hierarchical clustering technique.

TABLE I
QUALITIES OF CLUSTERING RESULTS FROM EIGENDEV-IPPCA AND HICLUST-IPPCA ALGORITHMS USING F-MEASURES.

	EigenDev-ipPCA	HiClust-ipPCA
Bovine dataset	0.889291	0.914681
Sheep dataset	0.716138	0.778155

C. Effect of the feature selection technique

We also investigate the effect of the feature selection procedure on clustering quality. The HiClust-ipPCA algorithm is applied to both bovine and sheep datasets with varying numbers of data dimensions (number of SNPs). The F-measures of the clustering results in comparison with the given genetic labels are shown in Fig. 4. As expected, the F-measure increases when the data dimension increases, because there is more information in the data matrix. For bovine dataset, the clustering quality stabilizes when the number of SNPs reaches approximately 35% (15,000 SNPs). In sheep dataset, the clustering quality keeps increasing with data dimensions. This result corresponds to the original data dimensions. The bovine dataset has 44,706 SNPs, so it is likely that some SNPs are either correlated or noise. On the other hand, the sheep dataset only has 1,406 SNPs, so all SNPs are informative.

IV. Conclusion

We have proposed a modification to the clustering module of the ipPCA framework based on a graph-partitioning technique and illustrates that the hierarchical clustering method provides

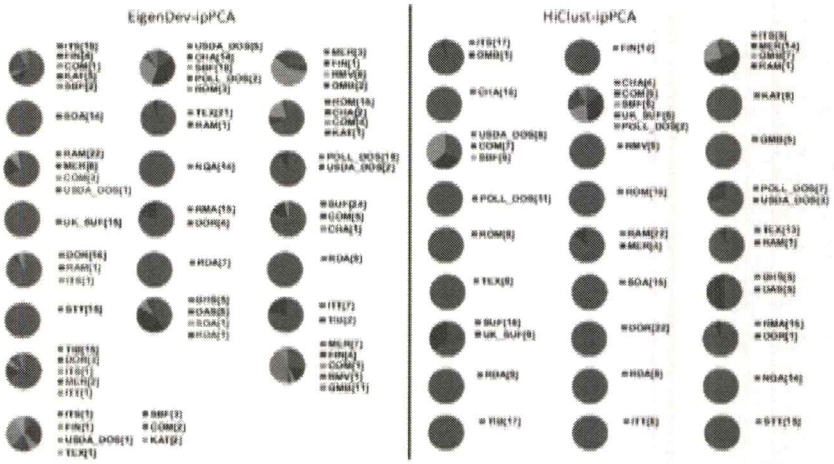


Fig. 3. Clustering results of the 28-breed sheep dataset. Each circle represents a cluster. The colors on the circle represent subpopulations within the cluster. The numbers of individuals are denoted in the square brackets. Left: EigenDev-ipPCA. Right: HiClust-ipPCA. The EigenDev threshold is 0.30.

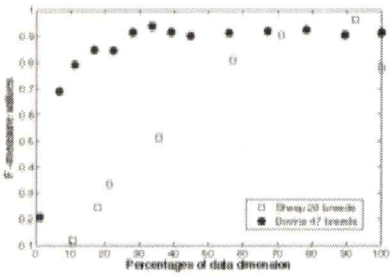


Fig. 4. F-measures of the HiClust-ipPCA clustering results vs. percentages of data dimensions.

better clustering results than fuzzy c-mean for complex genotype datasets. The HiClust-ipPCA algorithm is particularly good at reducing outliers in each cluster. We also find that the use of the feature selection technique is effective for reducing data dimension and increasing computational efficiency. However, the optimal dimension with maximum information is dependent on the dataset and is an open research topic worthy of future investigation.

ACKNOWLEDGMENT

The authors thank the National Science and Technology Development Agency (NSTDA) for funding supports through research grants JSTP-06-54-07 E to C. Amornbunchornvej and SCH-NR 2010-22-05 to T. Limpiti. S. Tongsima is supported by the Thailand Research Fund (TRF) under project no. RSA5480026. A. Assawamakin is supported by BIOTEC postdoc grant. We are also grateful for the permission from Dr. Mathieu Gautier to access the 47-breed bovine data.

REFERENCES

[1] A. Intarapanich, et.al., "Iterative pruning PCA improves resolution of highly structured populations," *BMC Bioinformatics*, vol. 10, pp. 382, 2009.

[2] N. Patterson, AL. Price, and D. Reich, "Population structure and Eigenanalysis," *PLoS Genet*, vol. 2, no. 12, pp. e190, 2006.

[3] T. Limpiti, et.al., "Study of large and highly stratified populations datasets by combining iterative pruning principal component analysis and structure," *BMC Bioinformatics*, vol. 12, no. 1, pp. 255, 2011.

[4] Y. Xu, V. Olman, and D. Xu, "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees," *Bioinformatics*, vol. 18, no. 4, pp. 536–545, 2002.

[5] O. Grygorash, Y. Zhou, and Z. Jorgensen, "Minimum spanning tree based clustering algorithms," in *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC, USA, 2006, ICTAI '06, pp. 73–81, IEEE Computer Society.

[6] P. Paschou, J. Lewis, A. Javed, and P. Drineas, "Ancestry informative markers for fine-scale individual assignment to worldwide populations," *Journal of Medical Genetics*, vol. 47, no. 12, pp. 835–847, 2010.

[7] S. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, pp. 241–254, 1967, 10.1007/BF02289588.

[8] JH. Jr. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.

[9] M. Gautier, D. Laloe, and K. Moazami-Goudarzi, "Insights into the genetic history of french cattle from dense SNP data on 47 worldwide breeds," *PLoS ONE*, vol. 5, no. 9, pp. e13038, 09 2010.

[10] JW. Kijas, et.al., "A genome wide survey of snp variation reveals the genetic structure of sheep breeds," *PLoS ONE*, vol. 4, no. 3, pp. e4668, 03 2009.

[11] N. Chinchor, "MUC-4 evaluation metrics," in *Proc. of the Fourth Message Understanding Conference*, 1992, pp. 22–29.

Publications

2. Amornbunchornvej C., Limpiti T., Assawamakin A., Intarapanich A., and Tongsima S., "Iterative Neighbor-Joining Tree Clustering Algorithm for Genotypic Data", Proceeding of the 21st International Conference on Pattern Recognition (ICPR2012), pp.1827-1830, Tsukuba Science City, Japan, 2012, International Association for Pattern Recognition.

Welcome to 2012 ICPR

November 11-15, 2012, Tsukuba, Japan

ICPR2012 is the twenty-first conference of the International Association for Pattern Recognition (IAPR), which will be held during November 11-15, 2012 at the Tsukuba International Congress Center, Tsukuba, Japan. Tsukuba Science City is about 50 km northeast of Tokyo and 40 km northwest of the Narita International Airport. The Haneda International Airport is 10 km south of Tokyo. The Tsukuba Express Line connects Tsukuba/Akihabara in 45 min.

Technically Co-Sponsored by IEEE IEEE computer society

21st International Conference on Pattern Recognition

November 11-15, 2012, Tsukuba International Congress Center, Tsukuba, Japan

Program at a Glance Monday Tuesday Wednesday Thursday Author Index Keyword Index

Last updated on November 7, 2012. This conference program is tentative and subject to change

Technical Program for Wednesday November 14, 2012

To view the keywords and abstract of a paper (if available), click on the paper title

08:30-09:00, Paper WePSAT1.35

PDF *Iterative Neighbor-Joining Tree Clustering Algorithm for Genotypic Data*

Amornbunchornvej, Chainarong	King Mongkut's Inst. of Tech. Ladkrabang
Limpiti, Tulaya	King Mongkut's Inst. of Tech. Ladkrabang
Assawamakin, Anunchai	National Center for Genetic Engineering and Biotechnology
Intarapanich, Apichart	National Electronics and Computer Tech. Center
Tongsima, Sissades	National Center for Genetic Engineering and Biotechnology

21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Iterative Neighbor-Joining Tree Clustering Algorithm for Genotypic Data

C. Amornbunchornvej*, T. Limpiti*, A. Assawamakin†, A. Intarapanich† and S. Tongsima†
*Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand
†National Center for Genetic Engineering and Biotechnology, Thailand
‡National Electronics and Computer Technology Center, Thailand
*kltulaya@kmitl.ac.th, ‡anunchaiice@gmail.com, ‡apichart.intarapanich@nectec.or.th

Abstract

Issues to explore in genotypic datasets include the number and characteristic patterns of subpopulations and, possibly, relationships among them. Model-based clustering methods have been adopted to find a number of clusters and the individual assignments. However, they cannot infer genetic relationships among subpopulations the way phylogenetic trees, e.g., the widely-used Neighbor-Joining (NJ) tree, can. In this paper we propose an unsupervised, iterative clustering framework called iNJclust. It performs clustering on an NJ tree with a graph-based partitioning technique. The iterative process enhances the zooming ability and corrects the topology of the final NJ trees. Inference on genetic similarities between subpopulations is also possible. As final outputs, the iNJclust algorithm provides an estimate of the number of clusters, individual assignments, a population tree, as well as sub-trees of the terminal nodes. We illustrate the superior clustering performance of the proposed algorithm using Human 27 populations, bovine 47 breeds, and sheep 28 breeds datasets.

1. Introduction

In genetic data clustering problems, we often encounter clusters that cannot be differentiated correctly and efficiently using model-based methods due to their arbitrary shapes and minimal characteristic variations. An alternative approach is to convert the clustering problem into a graph partitioning problem. Data points are viewed as leaf nodes on a graph, the pattern of clusters as graph topology. Graph partitioning is performed by selectively cutting the graph edges until a desired number of clusters is achieved. Thus, it is independent of any assumptions on cluster shapes and is found to work reasonably well as a clustering method [5, 12].

A key problem with many graph clustering methods in genetics, e.g., hierarchical tree clustering [7] is that the edge lengths on the graph do not translate to actual genetic relationships among individuals. These relationships can only be inferred in phylogenetic trees. One well-known tree is the Neighbor-joining (NJ) tree [3], which is constructed from a distance matrix of the data. The edge lengths on an NJ tree are proportional to genetic distances between the tree nodes. Surprisingly, the NJ tree has been utilized only in interpreting genetic relationships, but not for the clustering problems.

The distinct advantages of graph partitioning and NJ tree inspire us to develop a new framework for genotypic data clustering by combining the power of the two–using graph partitioning on an NJ tree. This enables us to cluster complex patterns according to the similarities of their genetic codes. We design an iterative scheme similar to [6, 10]. Besides increasing the algorithm's ability to differentiate closely related clusters, the iterative process allows automatic correction of the NJ tree topologies. The resulting clustering algorithm is termed **iterative Neighbor-Joining tree clustering** or **iNJclust**. We investigate the performance of the iNJclust algorithm using three large, highly complex genotypic datasets from human, sheep, and bovine.

2 Methods

The diagram of the iNJclust algorithm is depicted in Fig. 1. Single-nucleotide polymorphism (SNP) sequences are used to compute the allele sharing distance (ASD) matrix for constructing an NJ tree from the data. The algorithm then determines if the cluster is homogeneous, i.e., all individuals come from the same subpopulation. If not, the algorithm performs an NJ tree clustering to bisect the tree into two sub-trees with corresponding ASD matrices. Next, new NJ trees are re-

978-4-9906441-1-6 ©2012 IAPR

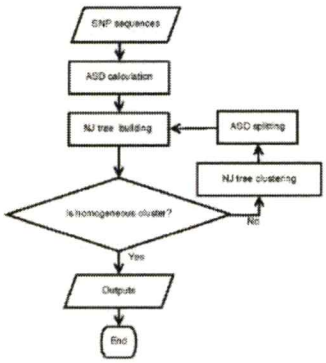


Figure 1. The iNJclust algorithm.

built from only individuals within each sub-tree. The process iterates until all sub-trees are deemed homogenous and finally terminates. As the final outputs, the iNJclust algorithm provides an estimate of the number of clusters, individual assignments, a population tree, as well as sub-trees of the terminal nodes.

Given $L \times 1$ SNPs sequences $\mathbf{x}_i$, $i = 1, \ldots, M$, from M individuals. Each $\mathbf{x}_i$ contains integer values of 0,1, or 2 representing the number of minor alleles at all loci. We compute the $M \times M$ ASD matrix $\mathbf{D}$ whose elements are the pairwise L1 distances between individuals, $\mathbf{D}(i,j) = \frac{1}{L} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel_1 \quad i,j = 1, \ldots, M$.

2.1 Iterative neighbor-joining tree clustering

The NJ tree is constructed from $\mathbf{D}$. Starting with all individuals as the leaf nodes, a pair of nodes that are nearest to each other as measured by the minimum evolution criteria [3] are merged to form a parent node. The Fitch-Margoliash distance between the newly created parent node u and its descendants f and g is

$$d(f,u) = \frac{d(f,g)}{2} + \frac{\left[\sum_{k=1}^{r} d(f,k) - \sum_{k=1}^{r} d(g,k)\right]}{2(r-2)}. \quad (1)$$

This equation gives the edge length (branch length) of the NJ tree. Then, we compute the distance between u and other nodes,

$$d(u,k) = \frac{1}{2}\left[d(f,k) - d(f,u)\right] + \frac{1}{2}\left[d(g,k) - d(g,u)\right]. \quad (2)$$

The pairwise distance between the new virtual node u to all other nodes are added to the ASD matrix. The tree construction process iterates until all nodes are merged onto the NJ tree.

To perform the NJ tree clustering, the edge between two nodes with largest length in Eq. (1) is cut, splitting the tree into 2 sub-trees. To avoid unwanted effects from noise or outliers, we do not permit edge-cutting that would result in a cluster with less than 10% of the individuals left in that iteration. After the split, new NJ trees for each subset of data points are reconstructed from the appropriate elements of the ASD matrix.

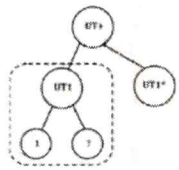


Figure 2. The homogeneous topology.

2.2 Homogeneity of cluster

To decide whether the algorithm should perform the NJ-tree clustering in each iteration, the homogeneity of the sub-tree is analyzed from its topological pattern. If the homogeneity criterion is met, the iNJclust algorithm terminates. This criterion stems from the NJ tree construction mechanism. During the early stage of tree building, nodes from the same subpopulation would be merged first before nodes from different subpopulations are combined. Consequently, it is observed that the bifurcating sub-tree of a homogeneous cluster (see Fig. 2) always contain one descendant with the so-called *UT1* topology pattern. The UT1 is an unbalanced bifurcating tree with a leaf node as one descendant, as depicted in the dashed-line box of Fig. 2.

Let $UT1_m^+$ be a tree topology at the m^{th} iNJclust iteration before NJ tree clustering. The iNJclust algorithm is to terminate when the homogeneous topology,

$$UT1_m^+ = UT1_{m+1} \cup UT1_{m+1}^* \quad (3)$$

is detected. $UT1_{m+1}^*$ can be of type UT1 or else. In other words, this stopping criterion indicates that a heterogeneous cluster (containing more than two subpopulations) should not be too close to the leaf nodes.

3 Results

Three large, complex SNPs datasets are used to evaluate the performance of the iNJclust algorithm. The first dataset contains 243,855 SNPs from 27 human populations, having 554 individuals [11]. The second comes from 47 breeds of bovine, having 1089 individuals and 44,706 SNPs in all [4]. The last dataset has 392 individuals and 1,406 SNPs from 28 breeds of sheep [8].

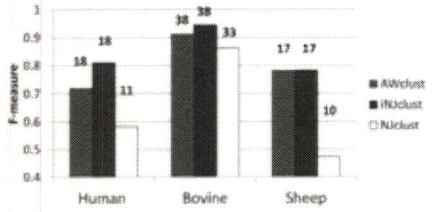


Figure 3. F-measures and numbers of clusters.

We compare iNJclust to an algorithm called AWclust [2], which is conceptually close to our proposed method. AWclust is the only existing method which uses the ASD matrix as input and performs clustering on a tree [9]. However, it uses a hierarchical tree [7] instead of an NJ tree, and does not include any iterative process. The main limitation of AWclust is that it can automatically detect subpopulations only up to 16 clusters. More than that, the number of clusters is required *a priori*. For fair comparison, the number of clusters for AWclust is set to equal the estimated cluster numbers from iNJclust in all datasets. To illustrate the benefit of iNJclust in correcting the NJ tree topology, we also investigate a simplified version of our algorithm (termed NJclust) where we iterate on the same stopping criterion, but do not reconstruct the new NJ trees on subsequent iterations.

The clustering assignment quality from each algorithm is compared against the given genetic labels of the datasets using F-measure [1]. The possible value of F-measure ranges between 0 and 1. The larger F-measure value, the better the performance of the algorithm. The F-measures from the AWclust, NJclust, and iNJclust algorithms are shown in Fig. 3 along with the estimated numbers of clusters at the top of the bars. The superiority of iNJclust over NJclust and AWclust is manifested in the F-measure values.

The detailed clustering results of the human dataset are depicted in Fig. 4-6 (due to space limitation, we only report the estimated numbers of clusters and F-measures for the bovine and sheep datasets). Each circle in Fig. 4 represents a cluster assigned by the algorithms. Different colors inside a circle represent distinct genetic labels. The number of individuals within each cluster is denoted in the square bracket. It is observed that the iNJclust algorithm assigns more pure clusters, although all three algorithms provide some clusters with mixed populations. This is unavoidable because some population groups, e.g., indians (Mala, Madiga, Tamil, Brahmin), east asians (Chinese, Vietnamese, Khmer),

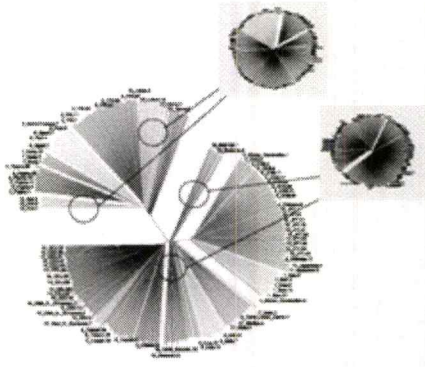


Figure 5. NJ trees with cluster labels.

are very similar genetically. However, AWclust result contains many more outliers (highlighted in red), which are corrected in iNJclust result. The cluster assignments from NJclust is the worst. This is because the NJ tree of the original dataset may have an incorrect topology, as illustrated in Fig. 5. We exemplify two genetic labels which have initially been separated in the topology of the NJ tree at the first iteration (main Fig. 5). However, the iNJclust is able to correct the tree topologies of these labels at the subsequent iterations (Fig. 5 insets) and eventually assign them to the right clusters.

Another advantage of the iNJclust algorithm is its ability to infer genetic similarities between clusters from the order at which each cluster is bifurcated in the iterative clustering process. This can be seen in Fig. 6 where the cluster numberings correspond to the assigned clusters in Fig. 4. African individuals are first separated from the rest because they are the most distinct subpopulation. Then East asians are separated out from Europeans and Indians. At terminal nodes, genetically similar subpopulations are finally differentiated.

4 Conclusion

We have proposed an unsupervised graph-based clustering framework capable of inferring three main quantities of interest in genetic data from an NJ tree: estimated number of clusters, individual assignments, and genetic relationships between subpopulations. Our iNJclust algorithm can differentiate arbitrary complex-shape clusters. The iterative process of the algorithm is beneficial in identifying the correct tree topologies. The bifurcating order of the final population tree also implies the genetic similarities between subpopulations.

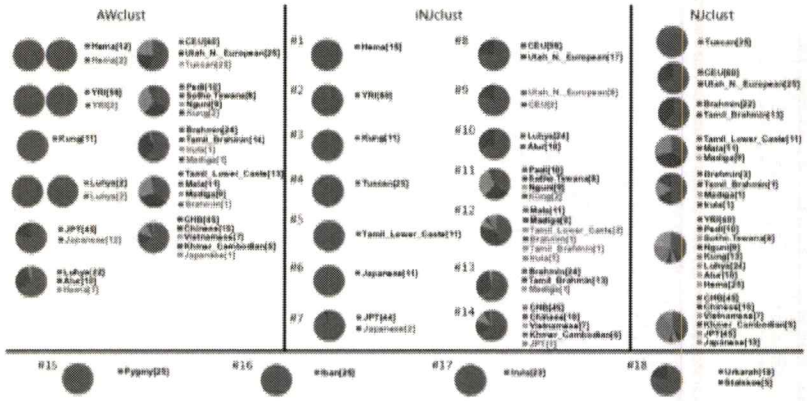


Figure 4. Clustering results of the human dataset. Bottom panel: Similar clusters among 3 algorithms.

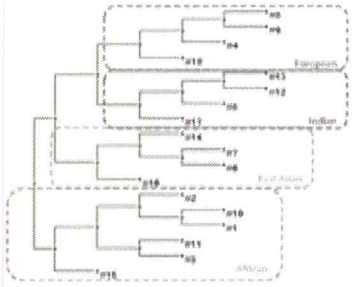


Figure 6. Population tree of Human 27.

Acknowledgment

C. Amornbunchornvej thanks the National Science and Technology Development Agency (NSTDA) for funding supports through research grants JSTP-06-54-07E. The authors also thank Dr. Mathieu Gautier for the 47-breed bovine genotype data.

References

[1] N. Chinchor. Evaluation metrics. In *Proc.of the 4th Message Understanding Conf.*, pages 22–29, 1992.

[2] X. Gao and J. Starmer. AWclust: point-and-click software for non-parametric population structure analysis. *BMC Bioinformatics*, 9:77, 2008.

[3] O. Gascuel and M. Steel. Neighbor-joining revealed. *Molecular Biol. and Evol.*, 23(11):1997–2000, 2006.

[4] M. Gautier, D. Laloë, and K. Moazami-Goudarzi. Insights into the genetic history of french cattle from dense SNP data on 47 worldwide breeds. *PLoS ONE*, 5(9):e13038, 09 2010.

[5] O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *Proc. of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence*, ICTAI '06, pages 73–81, Washington, DC, USA, 2006. IEEE Computer Society.

[6] A. Intarapanich, et.al. Iterative pruning PCA improves resolution of highly structured populations. *BMC Bioinformatics*, 10:382, 2009.

[7] S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32:241–254, 1967. 10.1007/BF02289588.

[8] J. Kijas, et.al. A genome wide survey of snp variation reveals the genetic structure of sheep breeds. *PLoS ONE*, 4(3):e4668, 03 2009.

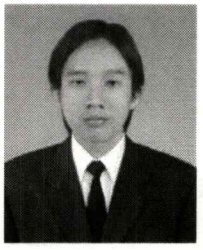
[9] D. Lawson and D. Falush. Population identification using genetic data. *Annu.Rev. Genomics Hum. Genet.*, 13, June 2012. Epub ahead of print.

[10] T. Limpiti, et.al. Study of large and highly stratified population datasets by combining iterative pruning principal component analysis and structure. *BMC Bioinformatics*, 12(1):255, 2011.

[11] J. Xing, et.al. Fine-scaled human genetic structure revealed by snp microarrays. *Genome Res*, 19(5):815–25, 2009.

[12] Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, 18(4):536–545, 2002.

BIOGRAPHY



Name Mr.Chainarong Amornbunchornvej

Birth date 19 December 1988

Hometown Bangkok Thailand

Address 15, Soi Serithai67, Rd.Serithai, Kannayao, Bangkok, 10230, Thailand

Education Bachelor of Engineering, Major Computer Engineering (Honor), 2010
King Mongkut's Institute of Technology Ladkrabang

Interested Researches:

1.) Data clustering

2.) Graph theory

3) Pattern recognition and machine learning

Publications:

[1] Amornbunchornvej C., Limpiti T., Assawamakin A., Intarapanich A., and Tongsima S., "Improved iterative pruning principal component analysis with Graph-theoretic hierarchical clustering", *Proceeding of the 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp.1-4, Phetchaburi, Thailand, 2012, ECTI association.

[2] Amornbunchornvej C., Limpiti T., Assawamakin A., Intarapanich A., and Tongsima S., "Iterative Neighbor-Joining Tree Clustering Algorithm for Genotypic Data", *Proceeding of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp.1827-1830, Tsukuba Science City, Japan, 2012, International Association for Pattern Recognition.