# SUPER PARAMAGNETIC CLUSTERING OF DNA SEQUENCES

SUGIARTO RADJIMAN

NATIONAL UNIVERSITY OF SINGAPORE

2005

# SUPER PARAMAGNETIC CLUSTERING OF DNA SEQUENCES

**SUGIARTO RADJIMAN**

(B.Sc. (Hons.), NUS)

**A THESIS SUBMITTED**

**FOR PARTIAL FULLFILMENT OF**

**THE DEGREE OF MASTER OF SCIENCE**

**DEPARTMENT OF COMPUTATIONAL SCIENCE**

**NATIONAL UNIIVERSITY OF SINGAPORE**

**2005**

# Acknowledgements

I would like to thank my supervisor Prof. Wang Jian-Sheng for his guidance, advice, tremendous patience, and for his continuous support and encouragement from the beginning till the completion of this work. My co-supervisor A.P Chen Yu Zong for fruitful discussions; I have much benefited from his input and suggestions.

I also would like to express gratitude to my classmates, especially Han Lianyi; I am thankful for his kind assistance throughout this work. Not to forget Li Hu for helping with some technical problems, and Ung Chung Yong for giving a biological perspective to the result.

# Contents

# Summary

We performed data clustering on a set of DNA sequences with active promoter regions. Super Paramagnetic Clustering method which is inspired by statistical physics model of a disordered ferromagnet is employed. With this method, we were able to mine some important clusters and capture correlations contained within the clusters. Besides successfully separating arthropod and vertebrate class, we found two human viral genome clusters: EBV and HSV-1. Their members were gene sequences which expressed proteins in lytic and latent cycles of infection. Another important result is the separation of the vertebrate class into two big clusters. We deduced that these two clusters correspond to housekeeping and tissue-specific genes by conducting a rigorous analysis of consensus octa-nucleotides of transcription factor binding sites (some part of the results of this thesis are also submitted for publication). The biological significance of these clusters are discussed.

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

In recent years, huge amount of data are generated due to dramatic rise of biotech industry, rapid development in information technology, and improvement in communication in general. It has been estimated that information will double every 20 months [54]. The outburst in biotechnology itself can be seen from the growing number of biological databases and the enormous data generated, e.g. from human genome project. These caused rapid accumulation of biological data, and in such a way made useful information difficult to extract. As a result, there is a strong urge in scientific community to develop new methods for dealing with avalanche of available data.

## 1.2 Aim of This Work

The study of DNA sequences as building blocks of living multicellular organisms provides opportunities to understand potentially any biological activity in the body system. Residing inside the nucleus of cells, DNA carries genetic information necessary for the organization of cells and controls the inheritance of characteristics. Nevertheless, even though massive amounts of genome-related information is now available due to the increasing number of genes sequenced and the use of new technologies such as DNA microarrays, extracting information from such data is still a challenging task. New reliable methods of analyzing a wide variety of genomic data are therefore preferred. In this work, we apply a novel clustering technique to analyze and extract the correlated behavior of gene sequences based on their expression function and classes division.

## 1.3 Thesis Outline

This thesis consists of 6 chapters and is organized as follows:

**Chapter 2** covers some basic biological background about our work such as central dogma of molecular biology. Introduction of DNA as the source of genetic information and nucleotides as its building block are discussed in this section. Finally, we talk about the importance of transcription process as the initial step in gene expression as well as promoter regions and transcription factors as two major components in the transcription process.

**Chapter 3** describes the big picture of data mining and knowledge discovery. In particular, it defines the connection between them. Paradigm of machine learning as the basis for information extraction is explained. We summarize the concept of cluster analysis in this chapter by defining the clustering task and the taxonomies of clustering methods.

**Chapter 4** covers detail explanation about Super Paramagnetic Clustering (SPC) algorithm, especially the main idea behind the method, which is inhomogenenous ferromagnetic Potts model; how this model is incorporated into data clustering problem, and how the Swendsen-Wang cluster algorithm is utilized to simulate the model. All details are discussed in this chapter.

**Chapter 5** presents the applications of SPC method. First, we applied SPC to Iris dataset; a well known problem in cluster analysis to demonstrate how the method works. In the second part, which is our main work, we test SPC to analyze a large dataset of active DNA sequences.

**Chapter 6** concludes our work. We restate the algorithm that is used and main clustering result. Some suggestions for possible future research improvement is also included.

# Chapter 2

# Biological Background

## 2.1 Central Dogma of Molecular Biology

The flow of genetic information is explained by the central dogma, the paradigm in molecular biology. It is based on the principle that genetic information travels from DNA to RNA, and finally to protein. Basically, the whole process consists of two main stages.

The first step is the transfer of genetic information from DNA to messenger RNA (mRNA) by *transcription* process that takes place in cell's nucleus. During transcription one template of DNA strand is copied into a complimentary sequence of RNA. This process, which is facilitated by enzyme RNA polymerase, creates an assembled piece of mRNA. In prokaryote cells, the mRNA that being produced is ready to be used due to the non-existent of a membrane that separates nucleus and cytoplasm. However, in eukaryote cell, the primary transcript (pre-mRNA) must undergo complex post-transcriptional processing. Pre-mRNA is transcribed from genes that contain introns and exons; prior to be biologically active, introns must be removed from the primary transcript. The process of removing intron and combining exons is called RNA splicing (Figure 2.1). The mRNA that carries the coded information leaves the nucleus and

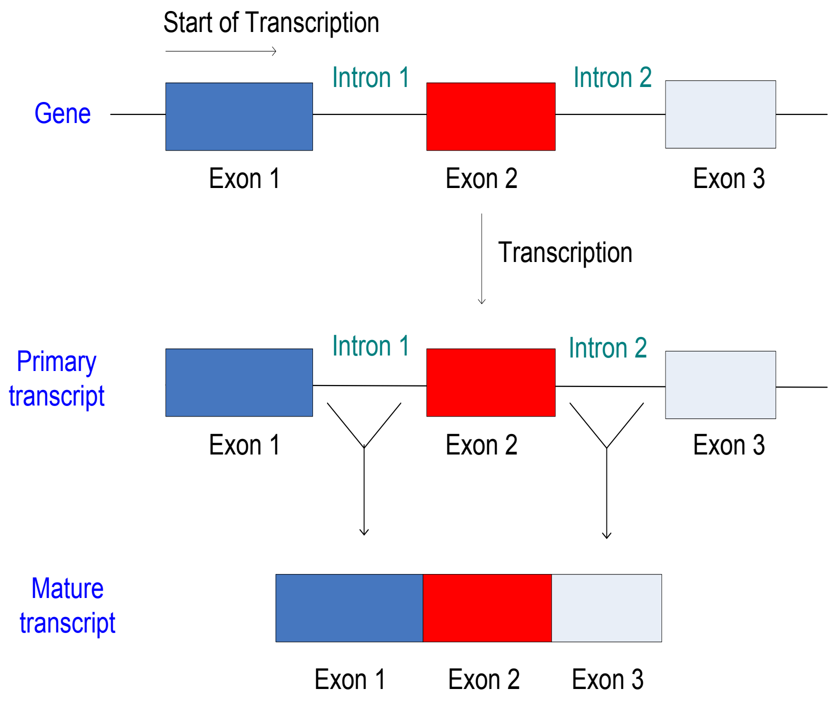

Figure 2.1: RNA splicing in eukaryote cells.

migrates to the cytoplasm. The second important stage is *translation*; in cytoplasm, one of cell's organelle, the ribosomes decode information from the combined exons and translate them into a sequence of amino acids as protein is formed. The proteins that

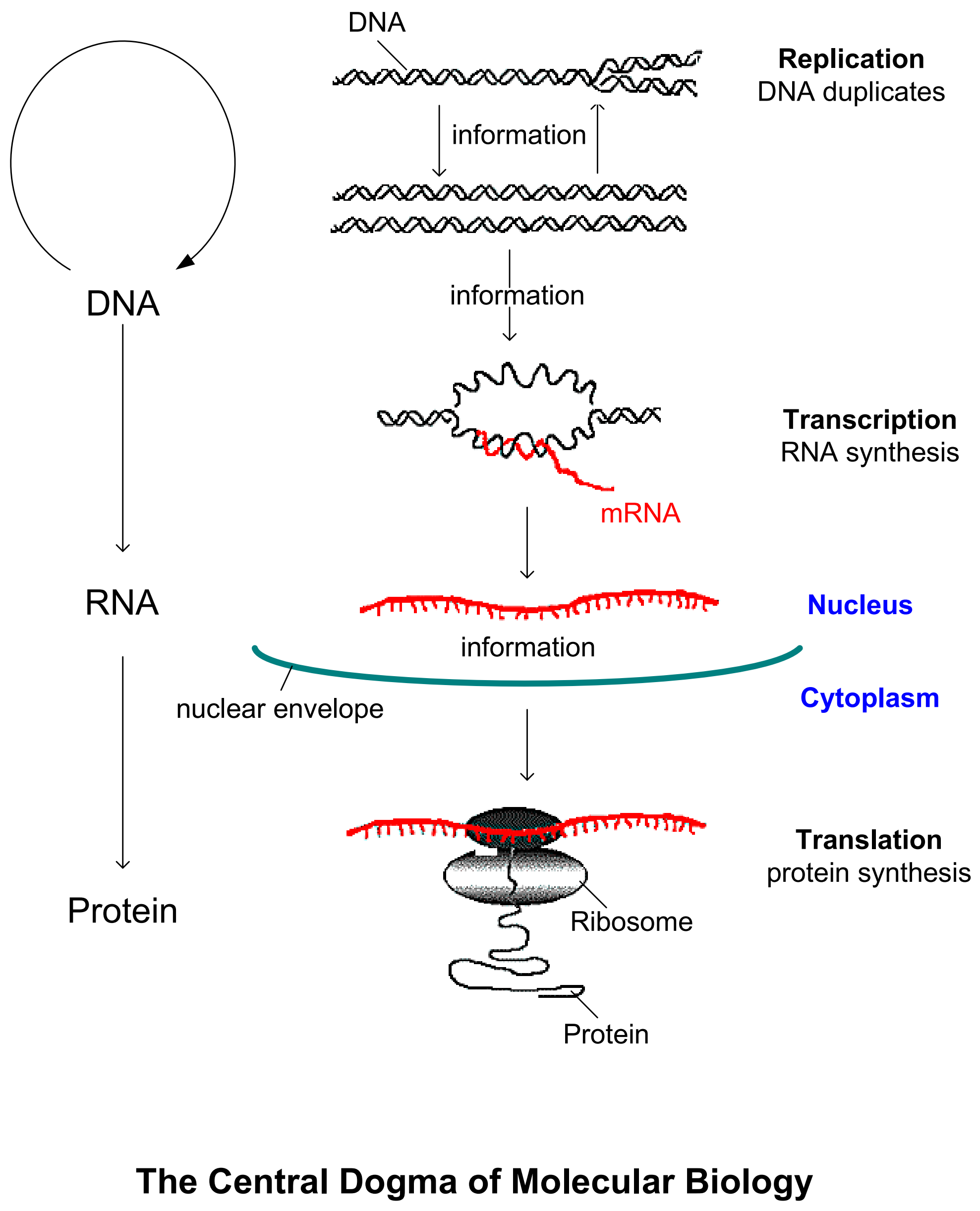

Figure 2.2: Main principle in the flow of genetic information as stated by Francis Crick in 1958.

play important roles in biological process will then perform most cell functions. Nevertheless, not all exons will be translated into protein. These exons are wholly or part of the 5' untranslated region (5' UTR) or 3' untranslated region (3' UTR) located in both end of the sequence.

Finally, to transmit the genetic information from one cell to another or from parents to progenies, DNA must duplicates its information in a replication process, in which a double stranded DNA is duplicated to produce an identical copy of DNA. This process involves many enzymes that unwind and copy the DNA strands. The cycle then continues in a new generation of cells or organisms. Figure 2.2 illustrates the central dogma (Figure is taken from http://www.accessexcellence.org/RC/VL/GG/central.html [2]).

## 2.2 Deoxyribonucleic Acid (DNA)

DNA contains the complete genetic information that defines the characteristics of an organism. Since the biochemical techniques for DNA sequencing were first developed more than three decades ago, the genomes of dozens of organisms have been sequenced. For example, one of the most important project in genome sequencing was human genome project, which took 13 year to complete the entire human genome (estimated to be 3 billion letters long) and finished in April 2003. Even though the human genome project has been completed, analysis of data generated will continue for many years.

DNA is formed by long linear polymer made up of nucleotides (nucleic acids).

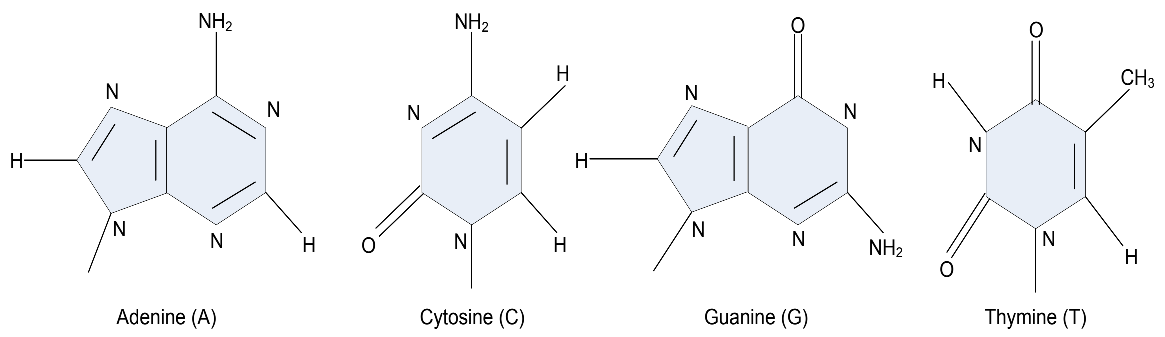

Figure 2.3: Four different nucleotides as bases for the construction of DNA sequences.

Each nucleotide has three parts: deoxyribose (a five carbon sugar), a phosphate group, and a nitrogenous base. Alternating of sugar and phosphate group creates a backbone for nucleic acids. There are four different types of nucleotide bases: adenine, thymine, guanine, and cytosine. The difference takes place in the nitrogenous base that is attached to the sugar-phospate unit. Adenine (A) and guanine (G) which have similar two-ring heterocyclic bases are called *purines*, whereas thymine (T) and cytosine (C) are called *pyrimidines* because they have similar one-ring structure.

In 1953 James Watson and Francis Crick proposed three dimensional structure of DNA molecule. The form is double helix composed of two twisted chains of nucleotides such that the backbone lies on the outside and the nitrogenous base lies on the inside. The key to this structure is specific base pairs (bp) arrangement between adenine with thymine (A-T), and guanine with cytocine (G-C) which is stabilized by hydrogen bonds. Because of this special property, the nucleotides sequence along one strand can completely determine the nucleotides sequence along the other strand.

## 2.3 Ribonucleic Acid (RNA)

RNA molecule acts as direct template for protein synthesis. As an intermediate agent in the flow of genetic information, RNA has same building block as DNA which consists of long linear polymer of nucleotides. The main difference is RNA only has single-stranded chain. RNA also contains the bases adenine (A), cytosine (C) and guanine (G); however, thymine base is absent and replaced by uracil (U). The uracil base lacks the methyl group and has an hydrogen atom instead (Figure 2.3). Cells contain three different classes of RNA based on its function:

1. Messenger RNA (mRNA)

   is the main coding template for protein synthesis or translation. It operates as a carrier of genetic information from nucleus to cytoplasm.

2. Transfer RNA (tRNA)

   is a type of RNA molecule that facilitates translation by carrying amino acids in activated form to the ribosome for peptide-bond formation as instructed by the mRNA template. Typically about 75–95 nucleotides long, tRNA is considered as one of the smallest RNA molecules.

3. Ribosomal RNA (rRNA)

   is a major component of ribosomes. It has several roles in protein synthesis, such as acts as a catalyst, recognizes as well as brings mRNA and tRNA to the correct position. The number is the most abundant among three types of RNA molecules.

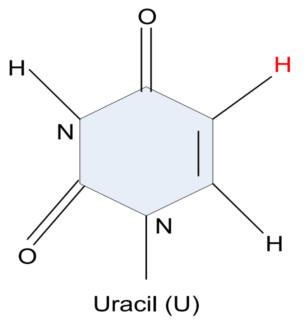

Figure 2.4: nucleotide base in RNA that replaces thymine base

## 2.4 Promoter Sites and Transcription Factors

Recall that transcription is the synthesis of RNA from a DNA template and is an important process in the flow of genetic information. During transcription, the initial step of gene expression, the transformation of DNA information into functional molecules, occurs. It works as the central mechanism by which the cell regulates proteins that are to be produced.

Viewing the mechanism in a more detailed scale, DNA template contains regions of initiation sites or promoter sites which work as the simplest regulatory element in the process of transcription. The promoters function as switches and code words that direct enzyme RNA polymerase to the proper transcription initiation site. In prokaryotes, RNA is synthesized by a single type of RNA polymerase, whereas RNA polymerase in eukaryotes can be categorized into three classes: type I, II, and III (Table 3.1). The RNA

| Type | Location | Synthesizes |
|---|---|---|
| I | Nucleolus | precursors of rRNA |
| II | Nucleoplasm | precursors of mRNA and small RNA |
| III | Nucleoplasm | tRNA and 5S rRNA |

Table 2.1: Eukaryotic RNA polymerases.

polymerase act as a catalyst in transcription and perform multiple tasks such as search promoter sites, select the correct ribonucleoside triphosphate, and detect a termination signal that specify end of transcript. Additionally, RNA polymerase also interact with activator and repressor proteins that control the rate of transcription (sigma factor in prokaryotes or transcription factor in eukaryotes). Transcription factors (TF) or also called DNA binding proteins are principally responsible for recognizing promoter regions and essentially required by RNA polymerase for binding to its promoter. This scheme (Figure 2.5) ensures that RNA polymerase in cooperation with TF specifically recognizes the corresponding region of the genome that needs to be decoded.

In biology, promoters and transcription factors are considered as cis-acting and trans-acting regulatory elements in the regulation of gene expression [12]. A comprehensive study of DNA sequences together with promoters and its TFs can therefore help us to understand gene co-regulation on the transcription level.

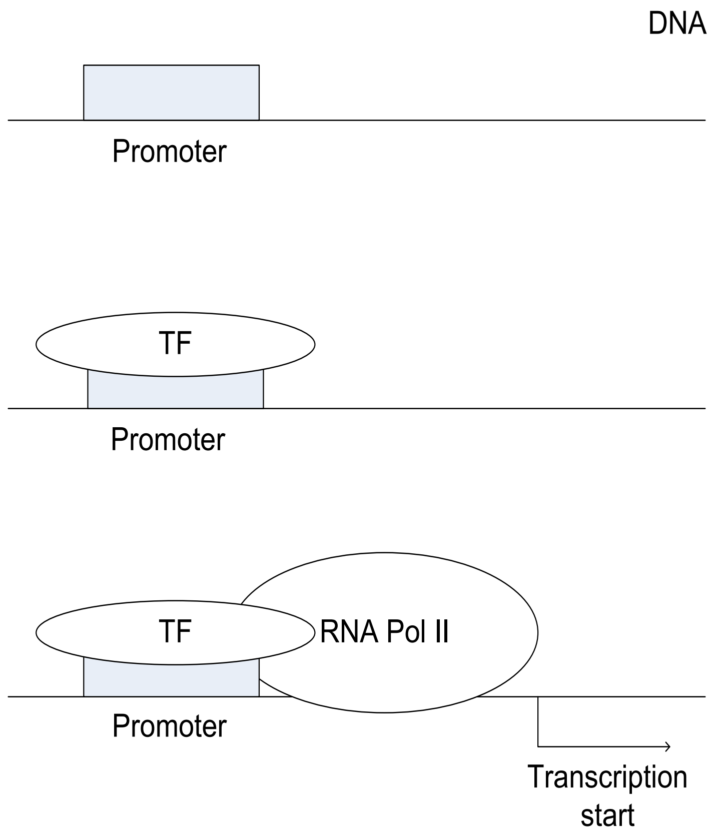

Figure 2.5: Interaction of TF, promoter sites, and RNA polymerase provide basic mechanism of transcription.

# Chapter 3

# Data Clustering Methods

## 3.1 Data Mining and Knowledge Discovery

Before we talk about Super Paramagnetic Clustering and its application in DNA sequence analysis in more detail, first we discuss the essence of this work which is mainly associated with data mining and knowledge discovery. Knowledge discovery is the task of identifying, exploring data to extract potentially useful information or discover patterns. Basically, its purpose is to help make sense of the data in a more understandable and applicable format. Data mining on the other hand is a process in knowledge discovery. It is developed as a means of identifying and extracting knowledge from data that is not evident [21]. The relationship between data mining and knowledge discovery is illustrated in figure 3.1.

Currently data mining techniques have drawn lots of attentions from industry

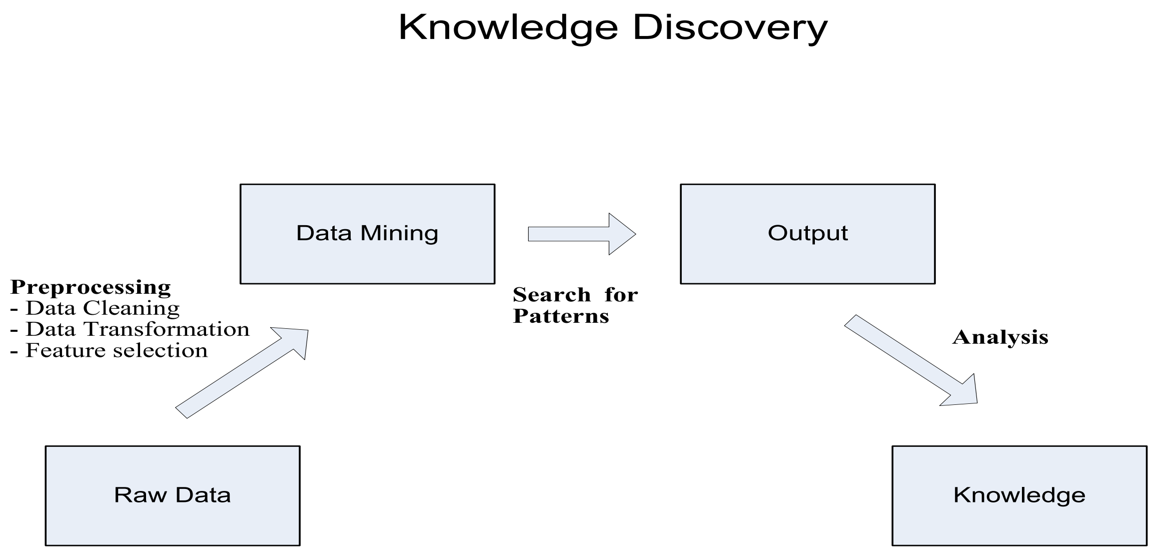

Figure 3.1: Main general scheme in knowledge discovery and data mining as one crucial process in understanding data.

and scientific communities. Both practitioners and researchers are trying to fully utilize sophisticated data mining techniques in order to have better understanding of raw data. Moreover, the abundance of raw data available has boosted data mining's popularity as a method of discovery. Very often it is used to support human decision-making or to explain observed phenomena. Data mining covers extensive range of well established methods, among them are evolutionary computing, neural network, machine learning, fuzzy sets, and Bayesian methods. This work involved with cluster analysis, one of the most fundamental technique in data mining. In the next section, a brief overview about machine learning is given to describe how clustering fit into the bigger framework.

## 3.2 Machine Learning

As an approach in data analysis and processing, machine learning provides the technical basis of data mining [57]. In the paradigm of machine learning, methods are generally divided into three sub-fields: supervised, unsupervised and reinforcement learning.

### 3.2.1 Supervised Learning

In supervised learning or learning by examples, the algorithm is provided with training set S, consists of both the cases and the predefined labels (response), that represent the concept to be learned.

$$S = \{(x_i, c_j) | i, j = 1, \ldots, N\}.$$

These $N$-training pairs are often called examples, where $x_i$ is an $d$-dimensional pattern vector, whose component are called features, and $c_j$ is a known label. The algorithm searches for a mapping function $f$, which approximates the unknown target function $c = f(x)$. It specifies relationship between the cases and the labels as in regression analysis. The goal is when applied to new cases, the function will be able to predict their corresponding labels. In other words, the function works as a classifier, which assigns a class to an object. The key challenge for supervised learning is the problem of generalization: After analyzing only a (usually small) sample of cases, the learning system should output a classifier that works well on all possible cases [7]. In order to achieve good function approximation, sufficient amount of training examples are therefore required. To test the quality of the function, the outcomes of this class of algorithms are usually evaluated on a disjoint set of examples from the training set, called the testing set. Methods fall into this category range from traditional statistics approaches, neural network, Bayesian network, nearest-neighbors method, and support vector machines.

### 3.2.2 Unsupervised Learning

This type of machine learning method is called unsupervised because there is no explicit teacher involved. In unsupervised learning or learning from observation, the algorithm receives only input data and uses an objective function (such as a distance function) to extract particular patterns. In complement to supervised learning, the system explores structure of data and directly discovers the classes on its own. Unsupervised technique is therefore more desirable if the classes are unknown or new classes are expected in the data. This learning method has been applied in wide variety of fields, few most common of which are cluster analysis, self-organizing map, and hebbian learning.

### 3.2.3 Reinforcement Learning

The idea of reinforcement learning system is learning by direct interaction with the environment. Similar with supervised learning, reinforcement learning approximates a mapping function from situations to actions by trial-error interactions with a dynamic environment [31]. Given a scenario, the learner makes actions and receives rewards from environment based on his actions. The task is to select action according to a strategy that maximizes the total amount of payoff. One central idea in reinforcement learning is temporal-difference method, which is introduced by Sutton [48]. It combines both Monte Carlo and dynamic programming ideas. The differences of supervised, unsupervised, and reinforcement learning are summarized in table 3.1.

| | Supervised | Unsupervised | Reinforcement |
|---|---|---|---|
| Trainer | User input data | – | Dynamic environment |
| Approximate Function | Classifier | – | Strategy |
| Assigned class | User | Method | Method |
| Data | Incremental | Non-incremental | Incremental |

Table 3.1: Differences of unsupervised, supervised, and reinforcement machine learning methods.

## 3.3 Definition of Clustering

Cluster analysis is an important technique in exploratory data analysis. Within the framework of machine learning, cluster analysis falls into category of unsupervised learning and is regarded as a fundamental technique in data mining. Cluster analysis has long track record and has been applied successfully in wide range of areas and disciplines such as economy, biology, medicine, computer vision, database, pattern recognition and remote sensing. Its broad range of applications reflects clustering's appeal and usefulness in many different fields.

Cluster analysis or simply clustering is a generic term for a range of numerical methods for exploring multivariate data. Its aim is to assign a group of objects (usually represented as a vector of measurements, or a point in a multidimensional space) into several compact groups (classes) called clusters. Since the classes are unknown a priori, the goal is to discover these classes from the data. This differentiates cluster analysis with classification (supervised learning), where the classes are predefined and the goal is to approximate a function for future unlabelled observations. Within a cluster, ideally objects should have similar patterns with high degree of observations, whereas objects that belong to different clusters indicate distinct category in measurement. As stated in [27], the problem of clustering can be mathematically formulated as follows: let $X \in R^{m \times n}$ a set of data items representing a set of $m$ patterns $x_i$ in $R^n$. Determine the partition of $X$ into $K$ different groups $C_k$ called clusters, such that every data that belong to the same cluster are more alike than data in different groups. The result of the algorithm is an injective mapping $X \rightarrow C$ of data items $x_i$ to cluster $C_k$.

## 3.4 Clustering Algorithms

There are many different approaches to clustering available in the literature; essentially partitional clustering methods can be divided into two different classes: parametric and non-parametric approach.

### 3.4.1 Parametric Methods

Parametric methods require some knowledge or assumptions about the data. For example knowledge that the data follows normal distribution or has specific clusters structure represented by a center of mass points. Based on this information, we may apply a more specific and suitable method that fit data requirement. In many cases, these assumptions are incorporated into a global criterion which subject to minimization by parametric methods [9]. For instance, statistical method $K$-means algorithm assumes that each cluster structure has spherical shape with a centroid, which is the mean of all the samples in that cluster. The algorithm iteratively assigning objects to $K$ clusters by using the distances to the centroids until an optimum value of criterion is found. The difficulty in this class of algorithms is for user in supplying parameter details such as a set of $K$ seed points. These variables become crucial since the results are sensitive to the initial partition and total number of centroids supplied [29]. Other classes of parametric methods include variance minimization, maximum likelihood, deterministic annealing, and fitting gaussian mixtures.

### 3.4.2 Non-Parametric Methods

Non-parametric methods make no assumptions about the underlying data structure and literally follow some local criterion for the construction of the clusters. When there is no a priori knowledge about distribution of data, it is more suitable to employ nonparametric approaches. Typical example of non-parametric method is agglomerative and divisive algorithm. Both algorithms are similar and produce a hierarchical structure. Agglomerative algorithm proceeds with original data and treats each point as a separate cluster. It merges each point repeatedly based on its relationship with other patterns. The divisive algorithm works in the opposite direction: one starts with a single cluster and continuously splits it into smaller clusters.

# Chapter 4

# Super-Paramagnetic Clustering

In recent years, many physicists have shown significant interest to domains that were not part of traditional physics. Sophisticated tools of experimental and theoretical physics were applied to investigate new fields. As the result, new interdisciplinary field such as biophysics and econophysics emerge [55]. An outstanding development in data mining is Super-Paramagnetic Clustering (SPC), a method using statistical physics approach to clustering problem.

The SPC is proposed by Blatt *et al* [9] and is a novel hierarchical clustering method inspired by physical behavior of inhomogeneous granular ferromagnet. It belongs to a non-parametric class algorithm based on a cost function, where no structure of the underlying distribution of data is assumed. Clustering of data is achieved by solving the physical problem of Potts ferromagnetic model. A Potts spin variable is assigned to each pattern in finite $D$-dimensional space. Short range interactions are introduced between neighboring spins, whose strength is a decreasing function of the distance. Two thermodynamic quantities: the susceptibility, and the spin-spin correlation function are calculated from Monte Carlo simulation and used to determine major phase transition and to partition patterns in cluster. This method has wide advantages, namely its robustness against noise and initialization, ability to generate automatic hierarchical structure, and most importantly the fact that no prior knowledge about distribution of data or cluster structure is needed.

This method has been successfully tested in various problems, for instance Iris data, Landsat data, yeast gene expression profiles. Recently, SPC is applied in clustering protein sequences from SwissProt and SCOP databases [50]. Coupled Two-Way Clustering (CTWC), another variation of SPC algorithm is also introduced for analyzing gene expression data of breast cancer, colon cancer, and leukemia [16, 22, 23, 24].

## 4.1 Potts Model

The Potts model was introduced as a generalization of Ising model. It was proposed by Domb to his research student R. Potts in 1952 and has been extensively studied for many years [58]. The idea came from the representation of Ising model as interacting spins which can be parallel or antiparallel. Considered as one of the most influencing model in statistical mechanics, the Ising model is a simple model of ferromagnet system. It mimics the behavior of granular ferromagnet in a lattice. Each lattice site, which represent an

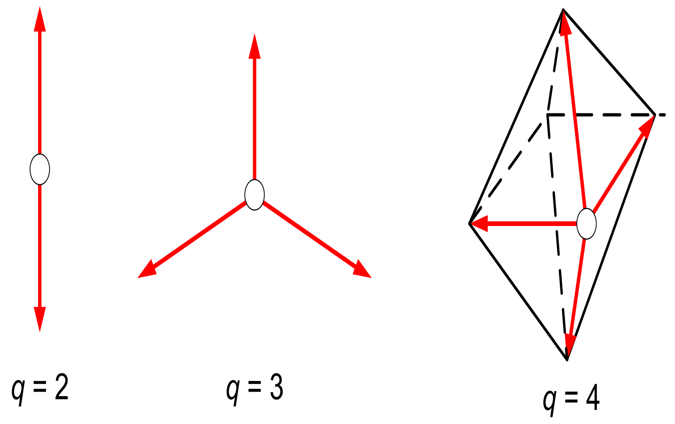

Figure 4.1: $q$-state Potts model pointing in the $q$ symmetric directions.

atomic magnet, is assigned with a spin variable $s$. The spin variable has two possible states; the spin point to either up $(+1)$ or down $(-1)$ direction. The Potts model extends the number of possible spin variables into $q$ number of states: $s = 1, 2, 3, \ldots, q$. The spin point to one of the $q$ equally spaced directions specified by the angles $\theta_n = 2\pi n/q$, where $n = 0, 1, \ldots, q-1$.

In a magnetic Potts model, the spins are located at points $v_i$ that reside on (or off) the sites of some lattice. Denote a configuration of spins for the system by $S = \{s_i\}_{i=1}^{N}$. The energy of such system is given by the Hamiltonian

$$\mathscr{H}(S) = -\sum_{<i,j>} J_{ij}\delta_{s_i,s_j} \tag{4.1}$$

where $J_{ij}$ represents the strength of coupled interacting spins $i$ and $j$, and $< i, j >$ stands for neighboring sites $v_i$ and $v_j$. The interaction between pairs of spins are therefore restricted to nearest neighbors with $J_{ij} > 0$ and 0 otherwise. Additionally, the energy contribution of a pair $< i, j >$ to the system $\mathscr{H}$ is only when $s_i = s_j$ due to Kronecker delta function term; that is when the two spins are aligned.

To investigate the behavior of the model, some thermodynamic quantities from the Hamiltonian need to be calculated. For example, in order to obtain the desired average quantity $A$, one need to calculate

$$< A >= \sum_{S} A(s)P(s), \tag{4.2}$$

where the Boltzmann factor,

$$P(s) = \frac{1}{Z}\exp(-\frac{\mathscr{H}(s)}{T}), \tag{4.3}$$

plays the role of a probability density function, which describes the statistical weight of each spin configuration $S = \{s_i\}_{i=1}^{N}$ in thermal equilibrium. $Z = \sum_S \exp(-\mathscr{H}(s)/T)$ is called the partition function which works as a normalization constant summing all $q^N$ possible energy states of spin configurations.

An important quantity to measure is the magnetization associated with a spin configuration $S$, the ordering parameter for magnetic system, which is defined as

$$m(S) = \frac{qN_{\max}(s) - N}{(q-1)N} \tag{4.4}$$

with

$$N_{\max} = \max N_1(s), N_2(s), \ldots, N_q(s),$$

where $N_\mu(s) = \sum_i \delta_{s_i,\,\mu}$ is the total number of spins with state $\mu$. From here, we also can calculate the susceptibility, which is proportional to the variance of magnetization

$$\chi = \frac{N}{T}\Big(\langle m^2\rangle - \langle m\rangle^2\Big), \tag{4.5}$$

to detect the thermodynamic phases of the system. Another quantity of interest which reflect the ordering properties of the system is the thermal average of $\delta_{s_i,s_j}$ or the spin-spin correlation function,

$$G_{ij} = \langle \delta_{s_i,s_j} \rangle, \tag{4.6}$$

which is the probability of two spins $s_i$ and $s_j$ being aligned.

### 4.1.1 Phase Transition in Ferromagnetic Potts Model

A phase transition is the change of a thermodynamic system from one phase to another. The phase transition is characterized by an abrupt sudden change in the physical properties measured, in particular with a small change in a thermodynamic variable such as the temperature. For a homogeneous Potts model where spins reside on a lattice, neighboring sites have equal distance $a$ with the same interaction strength $J_{ij} = J$. In this framework the Potts system exhibits two phases. At low temperature, individual spins have strong interactions with their neighbors; most spins are aligned with high probability and have high spin-spin correlation function. At very low temperature, pairs of spins are strongly coupled and interacted as a giant spin with high spin-spin correlation ($G_{ij} \approx 1$) and average magnetization $\langle m \rangle = 1$. In this state, the system is fully magnetized in ferromagnetic phase (lowest energy state). In contrast, at high temperature the system is in a disordered paramagnetic phase. As the temperature is raised, the system starts to undergo phase transition at the critical temperature $T_c$, in which the magnetization starts to vanish. At very high temperature, $\langle m \rangle = 0$, with $N_{\max}(s) \approx N/q$ indicating weak interactions among neighboring spins. The independent spins change its state randomly, thus pairs of neighboring spins have low spin-spin correlation function with only $\frac{1}{q}$ probability of being aligned ($G_{ij} \approx \frac{1}{q}$).

For a finite system in $D$-dimensional metric space where the local density varies (*i.e.* distances between points differ), it is more suitable to adapt inhomogeneous Potts model. In inhomogeneous Potts model, the spins form magnetic grains, with very strong coupling between neighbors within the same grain and weak interactions between the grains. Each site $i$ has different local distance with its neighbor $j$. The strength of interacting spin $J_{ij}$ must be carefully constructed as a decreasing function of distance, such that pairs of spins with close distance have stronger interaction and more likely to be in the same grain. Inhomogeneous Potts model has more interesting properties than its regular homogeneous part. The system exhibits three phases: ferromagnetic, super-paramagnetic, and paramagnetic states. Firstly at low temperature, the system also displays the behavior of ferromagnetic state: strong magnetization and high spin-spin correlation function among neighboring spins. However, as the temperature is increased, an intermediate super-paramagnetic phase appears. An interesting property of super-paramagnetic phase is that only strongly coupled spins are aligned (they are in their respective ferromagnetic phase), whereas there are no interaction among different grains. Using mean field approximation, the spin-spin correlation function in super-paramagnetic phase is estimated to be $G_{ij} \approx 1 - \frac{2}{q}\ O(\frac{1}{q^2})$ [11]. As the temperature is raised further, interactions become weak, magnetic grains disintegrate, and spins changing state independently, thus paramagnetic phase is reached.

The temperature $T_{fs}$ at which the transition from ferromagnetic to super-paramagnetic occurs is indicated by a pronounced peak in susceptibility, on the other hand, temperature $T_{sp}$ that signaled the transition from super-paramagnetic to paramagnetic region is indicated by abrupt decrease in susceptibility. Both $T_{fs}$ and $T_{sp}$

**Homogeneous System**

| Ferromagnetic | Paramagnetic |
|---|---|
| $T$-low | $T$-high |
| $\langle m \rangle = 1$ | $\langle m \rangle = 0$ |
| $G_{ij} \approx 1$ | $G_{ij} \approx \frac{1}{q}$ |

**Inhomogeneous System**

| Ferromagnetic | Super-Paramagnetic | Paramagnetic |
|---|---|---|
| $T$-low | $T$ | $T$-high |
| $\langle m \rangle = 1$ | $\langle m \rangle \neq 1$ | $\langle m \rangle = 0$ |
| $G_{ij} > 1 - \frac{2}{q}O(\frac{1}{q^2})$ | $G_{ij} \approx 1 - \frac{2}{q}O(\frac{1}{q^2})$ | $G_{ij} \approx \frac{1}{q}$ |

Table 4.1: Phase transitions in Homogeneous and Inhomogeneous Potts model

serve as lower and upper bounds for super-paramagnetic phase. Take note that there is a possibility that the super-paramagnetic region has several transitions (peaks in susceptibility). These happen whenever the spin magnetic grains experience some major separation.

## 4.2 Monte Carlo Simulation of Potts model

An efficient way in simulating the Potts model and calculating the thermodynamics average of quantities is by Monte Carlo simulation. A direct evaluation of equation 4.2 is impractical since the number of configuration $S$ increases exponentially with the system size $N$ (number of possible configuration is $q^N$), thus it can only be evaluated for small number of $N$. The purpose of Monte Carlo is to generate sample with a proper weight instead calculating direct thermodynamics average of physical quantity. A series of important spin configuration $\{S_1, S_2, \ldots, S_M\}$ is generated according to the Boltzmann probability distribution (equation 4.3). The calculation of a thermodynamic quantity $A$ is then reduced to a simple arithmetic average,

$$< A >= \frac{1}{M} \sum_{i}^{M} A(S_i), \tag{4.7}$$

where the total number of sample $M$ is much smaller than $q^N$ for large $N$.

### 4.2.1 Swendsen-Wang Algorithm

To do the simulation of Potts ferromagnet model, Swendsen-Wang (SW) algorithm is employed. Their method (also known as cluster algorithms) is very effective in reducing critical slowing down. The reason is because the local standard update of Metropolis algorithm merely flips one spin at a time, but SW flips a magnetic grain in one Monte Carlo step, which helps exploring a wide configuration space rapidly.

The algorithm starts with a initial spin configuration $S = \{s_i\}_{i=1}^{n}$. The initial state can be set randomly or by assigning each spin variables with a fixed integer number (i.e 1). Suppose after generating the $n$-th configurations, we want to generate the $(n+1)$-th configuration. We begin by visiting all neighboring pairs $\langle i, j \rangle$, that is pairs of spins with $J_{ij} > 0$. If the pair of spins have the same state ($s_i = s_j$), create a bond with probability

$$p = 1 - \exp(-\frac{J_{ij}}{T}\delta_{s_i,s_j}), \tag{4.8}$$

and no bond will be present otherwise. Having gone through all the interacting spins, we have a number of SW clusters formed by connected bonds. Update the spin configuration by assigning a new spin value independently to all cluster with equal probability from 1 to $q$. Members of the same cluster should be given the same spin value. Finally, erase all the bonds and we have a new configuration $S_{n+1}$. This defines one Monte

Carlo step. These procedures are ergodic and satisfy detailed balance [49, 52], as a result the simulation will produce the desired equilibrium distribution. Iterating this procedures $M$ times, we omit the first few number of configurations till the memory of initial configuration vanishes and equilibrium state is reached. From the simulation we measure all the thermodynamics quantity of interest, namely the magnetization, the spin-spin correlation function, and the susceptibility.

## 4.3 Clustering Data with SPC

Our goal is to incorporate the inhomogeneous Potts model into clustering problem. For a given set of data consist of $N$ number of feature vectors reside in finite real space, $\vec{v_1}, \ldots, \vec{v_N} \in R^D$, the detailed clustering procedures are as follow:

1. **Construct The Physical Analog of Potts Spin**

   a.) Assign a spin variable to each point $\vec{v_i}, i = 1, \ldots, N$.

   We choose an integer number for the number of Potts state and associate randomly one of $q$ possible Potts spin variables $s_i$, $i = \{1, 2, \ldots, q\}$ to each point $\vec{v_i}$. Blatt *et al* have found that the clustering result is insensitive to the parameter $q$ [8, 11]. However, it becomes necessary to run longer simulation as the number of Potts state increases to attain a good statistical accuracy. The parameter $q$ mainly determines the sharpness of transition and the temperature at which the transitions occur.

b.) Define the neighbors of each point based on selected criterion.

Since our data can be located anywhere in $D$-dimensional space, we need to define some conditions for interacting neighbors. Instead of using all $\vec{v_j}$, $j = \{1, 2, \ldots, N\}$, choose only a selected number of $\vec{v_j}$ as neighbors to limit the number of interactions according to the mutual neighborhood criterion. With this criterion, $\vec{v_i}$ and $\vec{v_j}$ are considered to be neighbors if and only if $\vec{v_i}$ is one of $K$-nearest neighbors of $\vec{v_j}$, and $\vec{v_j}$ is one of $K$-nearest neighbors of $\vec{v_i}$. Additionally, we impose the edges from minimum spanning tree to make sure that a connected graph which links all the data exists. These procedures lessen the number of interacting spins from $O(N^2)$ to $O(N)$; consequently computational efficiency increases and simulation running time is greatly reduced.

c.) Calculate the interaction $J_{ij}$ between neighboring points.

Introduce a short range interaction to determine the strength of interaction between neighbors $\vec{v_i}$ and $\vec{v_j}$,

$$J_{ij} = \begin{cases} \frac{1}{\hat{K}} \exp(-\frac{d_{ij}^2}{2a^2}) & \text{if } v_i \text{ and } v_j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

Here $\hat{K}$ is the average number of neighbors and the local length scale $a$ is the average of all distances $d_{ij}$ between neighboring pairs $\vec{v_i}$ and $\vec{v_j}$. This cost function of distance decreases exponentially fast such that points in high density regions have stronger interactions than those in low density regions. To calculate $J_{ij}$, the required input for the algorithm is therefore $N \times N$ dissimilarity matrix.

2. **Locating The Super-Paramagnetic Regions**

   a.) Calculate the thermodynamic quantities

   As has been explained before, the Swendsen-Wang algorithm is used to sample the thermodynamic physical quantities of interest. For a given range of temperature $0 \leq T \leq T_{max}$ with a temperature increament $T_{inc}$, we calculate the susceptibility $\chi$ (equation 4.5) and the spin-spin correlation function as a function of temperature. The spin-spin correlation function $G_{ij}$ for neighboring pairs of spins will be used to decide whether two spins belong to the same grain. Instead making use of equation 4.6, the SW provides an improved way to estimate this function [43] by averaging the indicator function

$$c_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \in \text{SW cluster} \\ 0 & \text{otherwise} \end{cases}$$

   for each spin configuration. The spin-spin correlation function is estimated by

$$G_{ij} = \frac{(q-1)C_{ij} + 1}{q}, \tag{4.10}$$

where $C_{ij} = \langle c_{ij} \rangle$ is the probability of $\vec{v_i}$ and $\vec{v_j}$ belonging to the same Swendsen-Wang cluster; it is the total number of times $\vec{v_i}$ and $\vec{v_j}$ are in the same grain divided by the total number of Monte Carlo iterations $M$.

b.) Analyze the susceptibility as function of temperature

The system self-organizes into different partitions in various temperatures. Location of the three regimes (ferromagnetic, super-paramagnetic, and paramagnetic) and its transitions can be identified by looking at the susceptibility as a function of temperature $\chi(T)$. The region of interest is super-paramagnetic phase, which is characterized by nonvanishing susceptibility. The susceptibility serves as a detector to monitor changes in phase transitions and major cluster break up, especially at temperatures where prominent peaks and sudden decreases in susceptibility take place. The first peak signals a transition from ferromagnetic to super-paramagnetic, in which a large cluster breaks into few macroscopic clusters. On the other hand, abrupt decrease in susceptibility corresponds to transition state from super-paramagnetic to paramagnetic region, at where major clusters break up into many smaller one.

3. **Constructing Clusters**

At selected temperatures of interest clusters are created by:

a.) linking each point $\vec{v_i}$ via its neighbors $\vec{v_j}$ with $G_{ij} > 0.5$

b.) connecting each point $\vec{v_i}$ to its neighbors $\vec{v_{j'}}$ with maximum correlation $G_{ij'}$ such that for any point $\vec{v_i}$: $G_{ij'} \geq G_{ij} > \frac{1}{q}$, $j = \{\text{neighbors of } i\}$

c.) Clusters are identified as those connected subgraphs obtained in steps a.) and b.).

# Chapter 5

# Applications

In this chapter we applied SPC algorithm to two different real data sets. The first dataset is a well studied case in cluster analysis. It is used as an example on how SPC can be applied for clustering problem. The second dataset is more challenging and is the main work in which SPC is employed to a set of selected DNA sequences.

## 5.1 Iris Data Analysis

The Fisher's Iris flower dataset is a popular multivariate data that has been used widely as a benchmark in discriminant analysis and cluster analysis. The data consists of 150 observations from three different species of Iris flowers: *Iris setosa*, *Iris versicolor*, and *Iris virginica.* Each specimen has four features including sepal length, sepal width, petal length, and petal width measured in millimeters. The 150$\times$4 data matrix is normalized by dividing each entry with summation of each feature,

$$\hat{x}_{ij} = x_{ij} / \sum_{i=1}^{150} x_{ij}, \ j = 1, 2, \ldots, 4.$$

We measure dissimilarity with Euclidean distance, then follow the steps as described in the previous chapter by choosing the number of Potts state $q = 20$, the number of mutual-neighbors $K = 5$, and impose edges from minimum spanning tree. Simulation is done with $10^5$ Monte Carlo sweeps for each sampled temperature. We then obtained the susceptibility density $\chi T/N$ curve as in figure 5.1a. The system experience two major cluster breaks. Initially all 150 data are in a single cluster due to the presence of a connected graph. All spins flip and change state as one big magnetic grain. However, at small temperature slightly above $T = 0$, a sudden increase in susceptibility density is observed. The system immediately cross ferromagnetic and enter super-paramagnetic region. The big cluster separates into two clusters of size 50 and 100. The smaller cluster correspond to the species *Iris setosa* and another cluster belong to *Iris versicolor* and *Iris virginica.*

The scatterplot graph (Figure 5.2), which compares each feature with the classification of flower, shows that there are overlapping in sepal length and sepal width, especially for *Iris versicolor* and *virginica.* Projection on the plane of Iris data spanned by its first and second principal component verifies that Iris setosa is well separated than the other two species (Figure 5.3). The SPC clearly handles the hierarchical organization of the Iris data very well, due to the fact that the two species are much closer

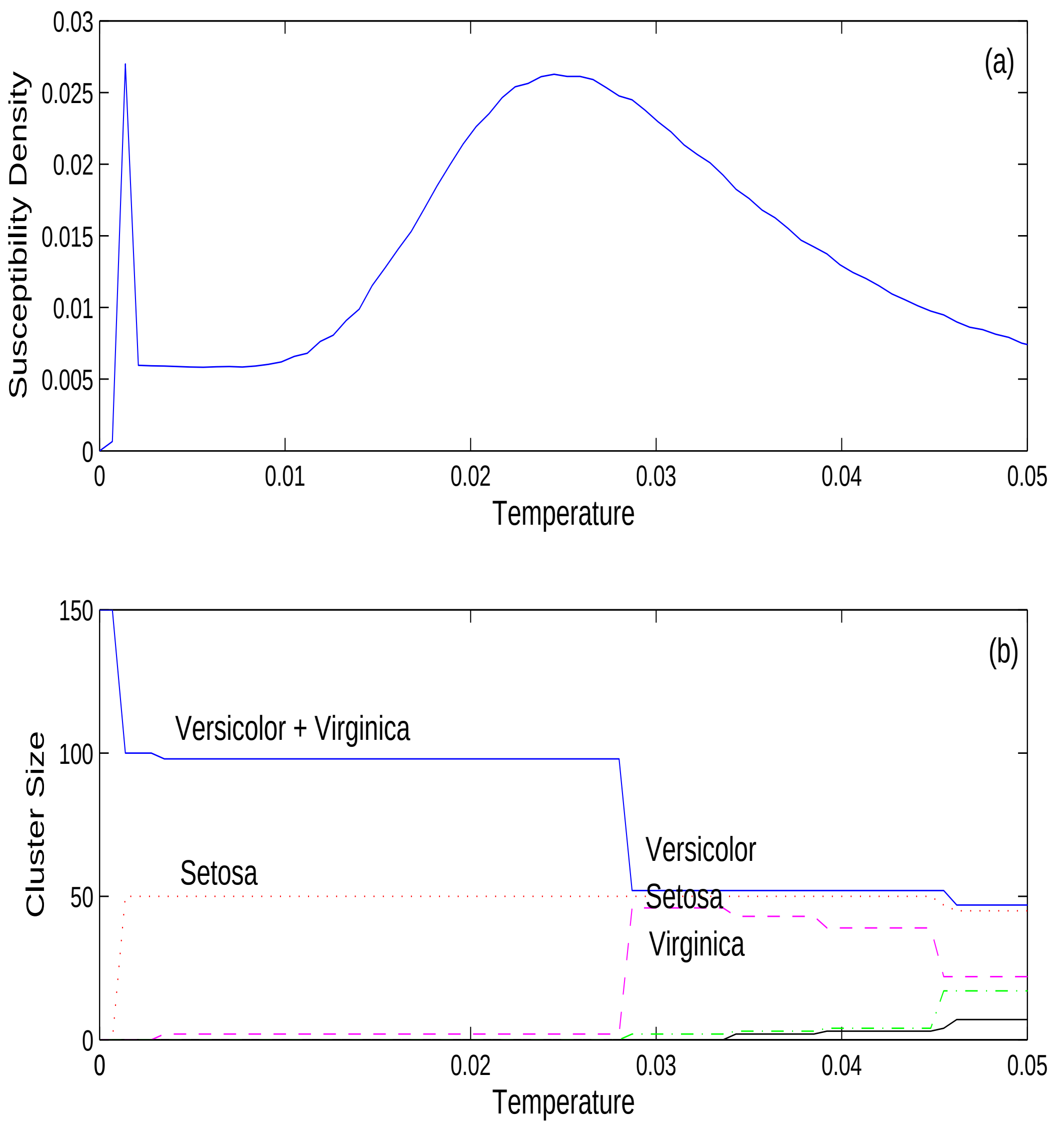

Figure 5.1: (a) The susceptibility density $\chi T/N$ as a function of temperature and (b) Cluster separation reflects hierarchical organization of Iris data.

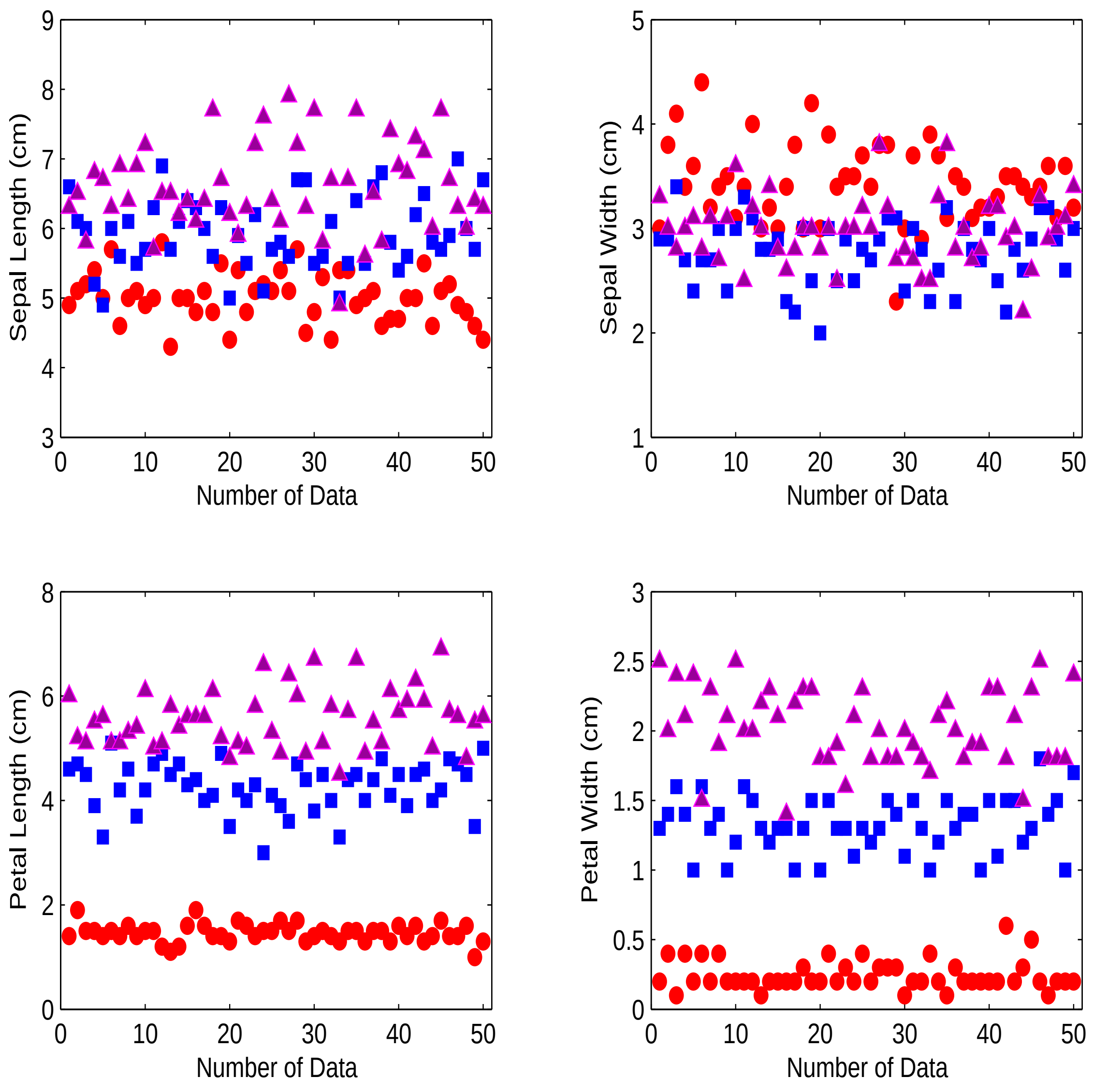

Figure 5.2: Features comparison of three different Iris flower species: Setosa (circle), Versicolor(square), and Virginica(triangle).

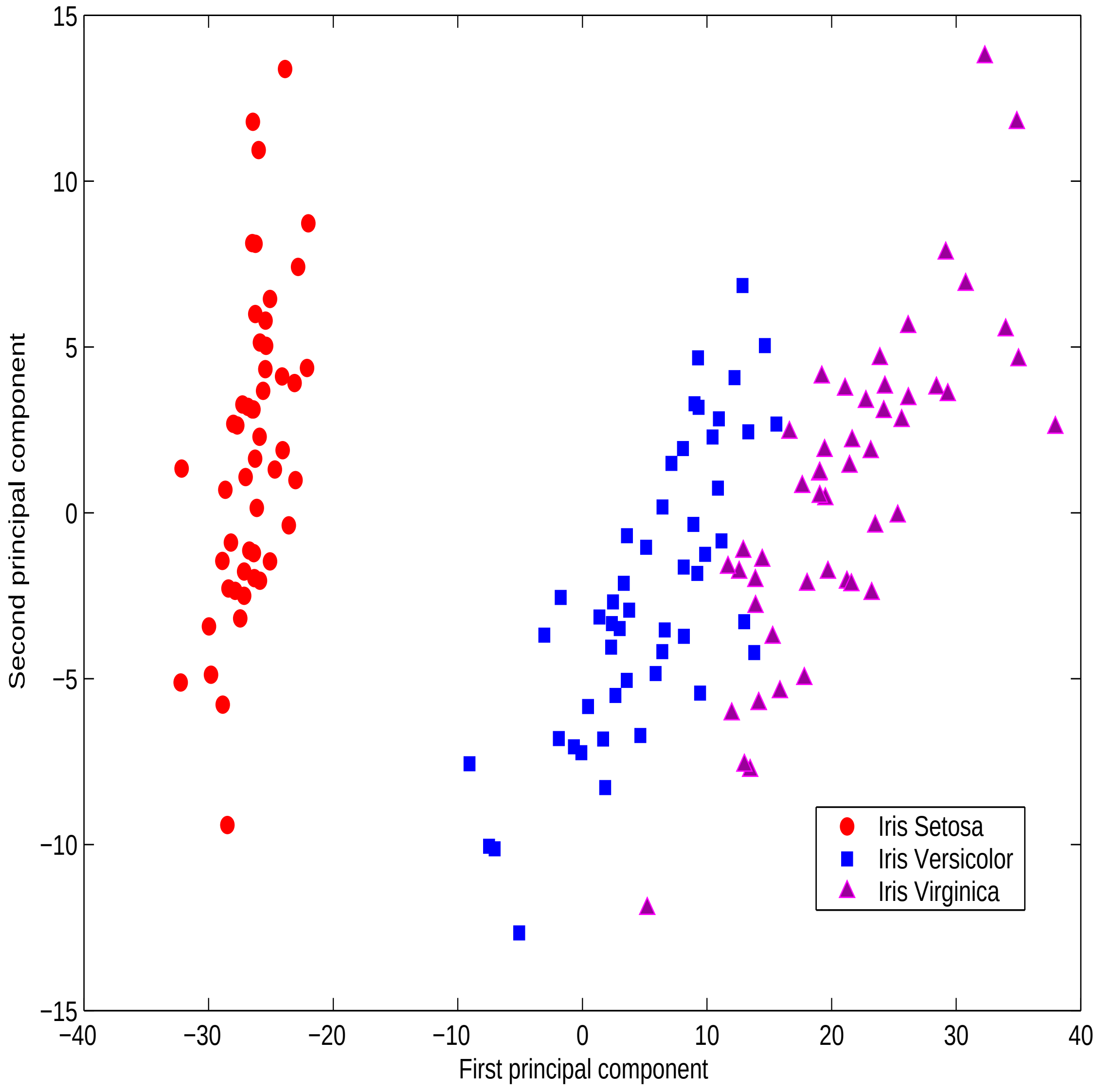

Figure 5.3: Principal Component Analysis of Iris data

than the third one.

As expected, later at $T \approx 0.3$ the bigger cluster is further divided into smaller clusters. The transition, however, is not as sharp as the first one. Strong interaction and overlapping among the other two Iris species create a smooth continuous dynamic transition in the susceptibility density. The success rate of SPC method is reflected from the result that does not deviate too much from manual classification. We identified three clusters of size 50, 52, and 46 belong to *Iris setosa*, *versicolor*, and *virginica* respectively. From the samples, 142 data ($\approx 95\%$) were correctly classified, 6 were wrongly classified (4 from *virginica* and 2 from *versicolor* ), and 2 (*virginica*) were left unclassified.

Comparatively, this result is better than previous results obtained by other classification and clustering techniques such as decision tree, genetic algorithm, valley seeking, complete link, directed graph, $k$-means, single link, and mutual neighborhood value [5, 10, 20, 45]. The most accurate result obtained so far is by minimal spanning tree procedure which divides the Iris data into three equal 50-size clusters [10]. However, unlike SPC method, to determine which edges in minimum spanning tree should be cut to split the correct clusters may be a difficult task. In most cases, cutting by edge length alone may be insufficient and some additional constrains must be imposed on the edges to be cut.

## 5.2 DNA Sequences Analysis

We now test the SPC algorithm on a set of eukaryotic DNA sequences. Since we would like to extract meaningful information from the gene function, only biologically active DNA sequences are included. Recall that the most important stage for controlling gene expression of eukaryotes organisms starts at transcription initiation, which is determined by the presence of promoter element within a given gene. Therefore, by selecting DNA sequences that contain strong promoter regions, we collect non-redundant genes for our dataset.

The set of gene sequences is taken from The Eukaryotic Promoter Database (EPD), a rigorous genome database containing a collection of eukaryotic RNA Polymerase II (POL II) promoters [47]. These promoters have been experimentally determined to be active in higher eukaryotes with an accuracy of ±5 base pair (bp) for the transcription initiation sites. We assembled $N = 4541$ sequences as a data source by combining 2000 sequences from arthropods (about 99% from Drosophila families) with 2541 from vertebrates (about 74% from Homo sapiens). The sequences also include a number of viral genes from arthropod and vertebrate. Other divisions, which contributed only a small number of DNA sequences, were removed to eliminate bias in the data. We then aligned all the sequences beforehand relative to a putative Transcription Starting Site (TSS).

### 5.2.1 Quantitative Measurement of DNA Sequences

Before we apply the SPC method, a suitable genetic distance measure need to be provided. Recall that a DNA sequence can be represented by string of letters taken from four symbol alphabet A, C, G, T namely the four nucleotide bases (adenine, cytosine,guanine, and thymine) which the DNA is composed from. A genetic distance between two independent organisms is commonly defined as the number of differences in nucleotides at specific sites of DNA fragment. The disadvantage of such measure is that it does not take into account displacement of nucleotides fragment and variation rate of nucleotides at different sites [55]. To obtain some numerical measurements from the data, we thus followed Abe *et al*' [1] approach by counting frequencies of $k$-nucleotides. This kind of measurement is useful for characterizing a sequence. Research has shown that nucleotide frequencies are similar for related organisms and different for unrelated organisms [1, 13, 32, 33, 42, 44]. Furthermore, the frequencies of nucleotides are relatively constant over the whole genome, in both coding and non-coding regions of DNA.

We then measure frequencies of tri-nucleotides and tetra-nucleotides by shifting one nucleotides for each sequence $s_i, i = 1, 2, \ldots, N$ from $-9999$ to 6000 bp regardless of its position. Take note that a number of sequences have shorter length than 16000, and the mean length is 13258. Frequencies of higher nucleotides were not used so as to keep the problem feasible. This is required to reduce computational complexities because each combination of nucleotides of length $L$, which has a one-to-one mapping with the number of features (dimension), increases exponentially as function of length. In addition, by using less stringent conditions redundancies and noise in feature measurement are removed. Altogether we have a total of 321 features, from where 64 are evaluated from 3-mers, 256 from 4-mers, and one additional feature which measures the length of each DNA sequence. This provides a 4541×321 input matrix of measurement. This is a large and complex multi-dimensional data set. Standard statistical clustering algorithm will face great difficulty to obtain reasonably good results, namely in determining optimal number of clusters, and initial/cut-off parameter.

Firstly, data entries are normalized for each column, such that all values fall between zero and one,

$$\hat{m}_{ij} = m_{ij}/\max(m_{ij}),\ j = \{1, 2, \ldots, 321\}.$$

The Euclidean distance $d_{mn} = \parallel \vec{v_m} - \vec{v_n} \parallel$ is calculated, and fed into the symmetric distance matrix in the SPC algorithm. For this particular problem, we still use the number of Potts state $q = 20$, set mutual-neighbors $K = 10$ and impose the edges from minimum spanning tree for the simulation.

### 5.2.2 Simulation Results and Analysis

The system behavior at different temperatures is monitored by plotting susceptibility density. Starting the simulation from initial temperature $T = 0$, the temperature is increased gradually till it reaches 0.16, where the value of susceptibility vanishes (Figure 5.6). For every sampled temperature we did $2 \times 10^5$ Monte Carlo sweeps with the Swendsen-Wang algorithm. Focusing only on main clusters in our analysis, a threshold is set and only those connected points with at least 20 members are considered as a cluster. Those clusters with sizes below the cut-off value are left as unclassified points. At low nonzero $T$ we detected two stable clusters of human viral genes Herpes Simplex Virus type 1 (HSV-1) and Epstein-Barr Virus (EBV), near $T \approx 0$ and $T = 0.035$ respectively. The DNA sequences in each cluster share same homology and exhibit more than 50% similarity between $-79$ and 20 bp [47]. SPC manages to mine all 29 (100%) HSV-1 genes and 22 out of 23 ($\approx$96%) EBV genes. Taking into account their sizes and threshold value, both clusters are stable because they only dissolve when $T$ reaches 0.06 and 0.05.

We observe two peaks in the susceptibility density curve (Figure 5.4). Notice how the cluster sizes changed drastically right after susceptibility density curve reached maximum position. The first sharp peak around $T = 0.05$ signals how the system undergoes major divisions into two separate clusters of arthropod and vertebrate classes with size 1704 and 2548. We spot 185 entries (9.25%) of arthropod sequences being misclassified into the vertebrate group. From this group, 36 entries ($\approx$19%) come from DNA sequences of polyphemus moth and silkworm, whose function is to regulate the synthesis of chorion related proteins. Interestingly the chorion, secreted by the follicle cells, is either a protective membrane which surrounds the eggs of insects and fishes, or the extra embryonic membrane derived from the trophoblast which surrounds the embryo of amniote vertebrates [3]. The others ($\approx$15%) include heat shock protein, ribosomal, histone, and alcohol dehydrogenase family proteins which are also commonly found in vertebrate cell system.

The first major breaking of clusters suggests that vertebrate and arthropod genes have strong dissimilarities in their genome pattern arrangement. Our study suggests that some of the normalized frequencies of tri-nucleotides are the determining factors which separate vertebrate and arthropod gene sequences. We observed high concentrations of AGG, CCC, CCT and low concentrations of ACG, CAA, CGA, CGT in the vertebrate class. On the contrary, high concentrations of AAC, ACG, ATC, CGA, TCG and low concentrations of AGG, CCC, CCT were present in the arthropod class (Figure 5.5). This result agrees well with Abe *et al*'s result in utilizing self-organizing map (SOM), an unsupervised neural network algorithm, for analyzing prokaryotic and eukaryotic genomes [1].

The second phase transition near $T = 0.09$ has a lower peak; it signals the breaking of the vertebrate cluster into two separate clusters of size 910 and 1414. The smaller cluster (Cluster VTB1) displayed high positive correlations within its sequence expression patterns, whereas the bigger cluster (Cluster VTB2) has diverse variations in functionality, and is largely dominated by unclassified chromosomal genes. To further examine correlations inherent in each respective cluster, we searched for consensus Transcription Factor Binding Sites (TFBS) motifs [18, 56] in the region $-1000$ to 1000,

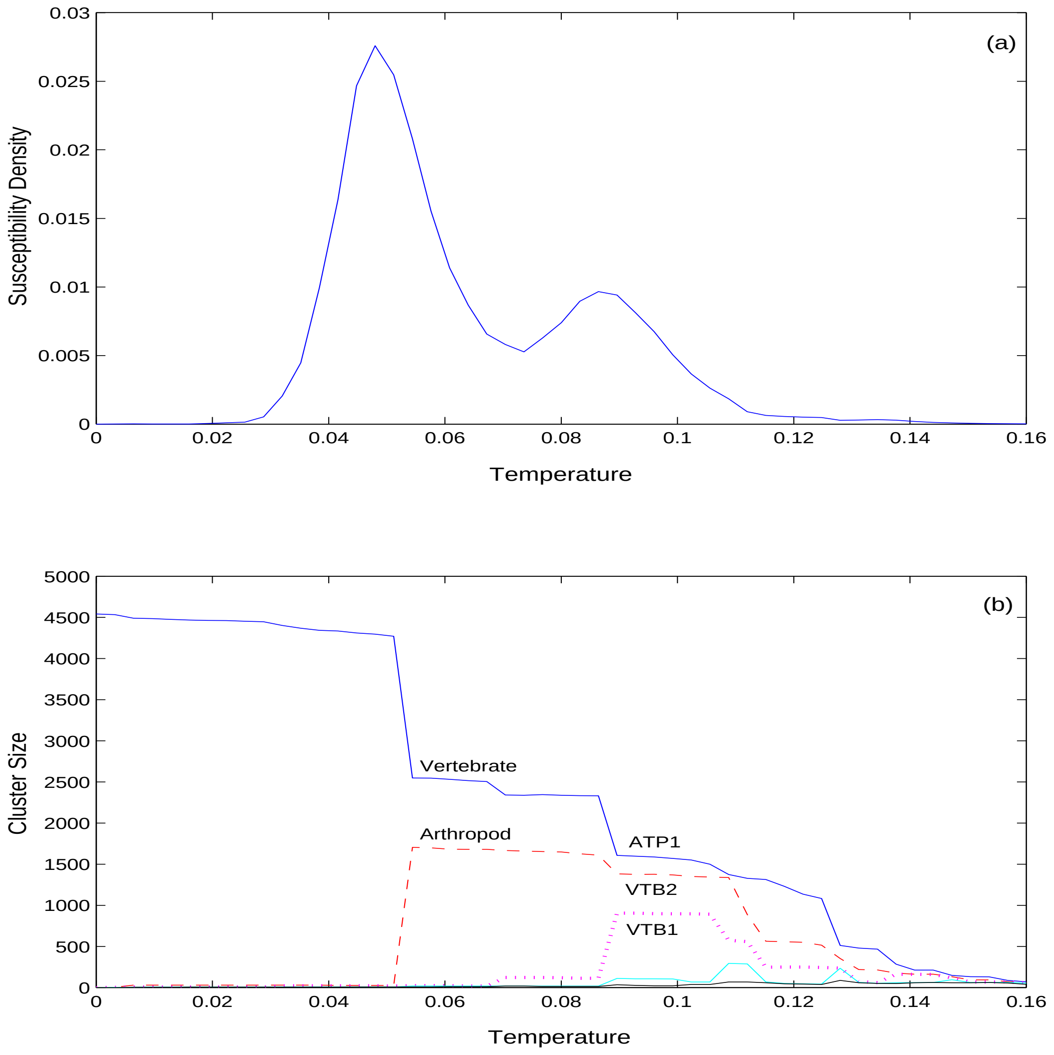

Figure 5.4: (a) Two peaks were obtained in the measurement of susceptibility density as an indication of major cluster split. (b) The size of four biggest clusters plotted as function of temperature.

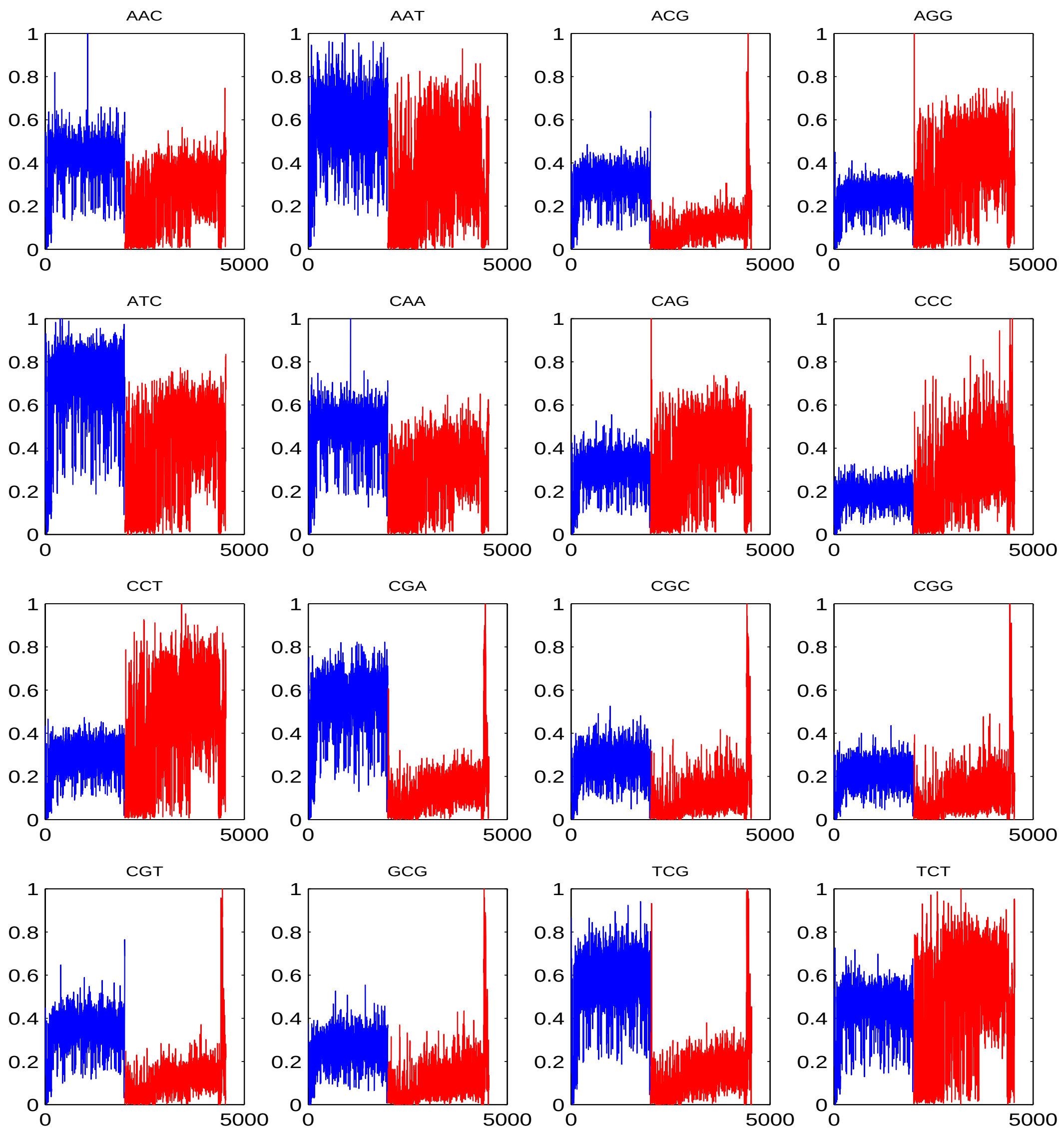

Figure 5.5: Distinct levels in normalized frequencies of arthropod (blue) and vertebrate (red) class that contribute significantly to their separation are shown in 16 out of 64 tri-nucleotides.

given the fact that most binding sites tend to cluster together and take place near TSS [19, 38]. All chunks of gene sequences with length 2000 bp are divided into 20 smaller fragments of length 100 bp each. The 1st DNA fragment starts from upstream region −1000 to −901. We then searched every fragment for consensus octa-nucleotides TFBS and filtered the number of occurrences so that only those consensus motifs with frequency above a threshold value 40 were kept for the analysis.

Our investigation shows that several TFBS motifs have been clustered together in certain positions of DNA sequences. A number of groups of Sp1 TFBS positioned from −200 to 100 bp and TATA box in the upstream region from −100 to TSS were identified in cluster VTB1. Indeed, the TATA box, as the most important cis-acting element for most genes transcribed by RNA polymerase II, is usually centered in the upstream region between −100 and −30 [4]. Similarly we identified Sp1, ETS, NRF-1, CpG island TFBS packed within region −400 to TSS in cluster VTB2 (Table 5.1).

The most important thing to point out from analysis of consensus motifs TFBS is the single occurrence of the TATA box motif in cluster VTB1 and the union of ETS, NRF-1, CpG in cluster VTB2. Firstly, although the TATA box is commonly found in most eukaryotic genes, it is absent in some genes, particularly housekeeping genes expressed in all tissues and in some tissue specific genes [12, 34, 53]. This attribute has been revealed in two independent studies by Fitz *et al* [19] and Murakami *et al* [40], which indicate TATA box association with promoters of tissue-specific genes. In contrast ETS, NRF-1, CpG motifs are predominantly found in promoters of housekeeping genes

[19, 35]. Another motif Clus 1: TCTCGCGA, an unknown TFBS which Fitz *et al* also pointed out has a strong relationship with promoters of tissue-specific genes, though not clustered in VTB2, still has very significant occurrence (102 times in total) compared with VTB1 (12 times in total).

On the other hand, Sp1 is the only motif observed in both clusters. The total number nevertheless is particularly large in the VTB2 cluster. Sp1, a 105-kDa transcription activator, usually binds with high affinity to promoters that contain CG rich regions (vertebrate class in this case). It is one of the most frequently found transcription activators required for the expression of large number regulated genes, and is effectively conserved in the promoters of most housekeeping genes [14, 15, 30, 46]. The absence of signature TATA box, the existence of GC rich region, and multiple Sp1 binding sites are characteristics of housekeeping genes [41, 51, 60]. All of these characteristics are found in the VTB2 cluster. Comparison of the clustering result with some known housekeeping genes [17, 28] also showed positive correlations. We are much convinced after searching for gene expression patterns related to ribosomal gene, a typical housekeeping gene which is essential for general cell function. Examining both clusters; there are 84 entries from ribosomal related gene in cluster VTB2, while only 23 entries are observed in cluster VTB1.

We attempt to classify the data further by increasing temperature parameter $T$ till it reaches 0.11 (Figure 5.6 shows the hierarchical organization of cluster break ups). This is when the number of unclassified points has gone above 38% and the system has reached the paramagnetic region. Beyond this temperature all clusters simply disintegrate, because strong correlations that characterize members of individual clusters have disappeared. Since each spin does random flips, we have small spin-spin correlation functions with neighbors ($G_{ij} \approx 1/q$), hence creating many solitary clusters. Between temperature 0.11 and 0.12 the number of unclassified data in main arthropod cluster (ATP1) increases drastically, causing this cluster to break up rapidly from 1 into 19 clusters within short temperature range. This phenomenon shows that the arthropod cluster only has one phase transition from ferromagnetic to paramagnetic and has no super-paramagnetic region. It merges naturally as single large cluster and therefore quickly disintegrates into many small grains after reaching a critical temperature. Domany *et al* encounter a similar phenomenon when applying the SPC method to two simulated datasets generated by uniform and Gaussian distribution. SPC demonstrates its superiority by not imposing a partition to the data when there are no natural classes present [10].

**Cluster VTB1**

**Sp 1**

| | | | | | |
|---|---|---|---|---|---|
| 8 | CCCGCCCC | 91 | 8 | CCCCTCCC | 73 |
| 8 | CCCCGCCC | 97 | 9 | CCCGCCCC | 154 |
| 9 | GCCCCGCC | 143 | 9 | CGCCCCGC | 93 |
| 9 | CCGCCCCC | 76 | 9 | CCCCGCCC | 146 |
| 9 | CCGCCCCG | 86 | 9 | CCGCCGCC | 41 |
| 10 | GCCGCCGC | 85 | 10 | CCGCCGCC | 86 |

**TATA**

| | | | | | |
|---|---|---|---|---|---|
| 9 | CTATAAAA | 48 | 9 | GTATAAAA | 54 |
| 9 | TATAAAAG | 71 | | | |

**Cluster VTB2**

**Sp 1**

| | | | | | |
|---|---|---|---|---|---|
| 6 | CCCGCCCC | 78 | 7 | CCCGCCCC | 117 |
| 7 | CCGCCCCC | 69 | 7 | GCCCCGCC | 101 |
| 7 | CCCCGCCC | 141 | 8 | CCCGCCCC | 277 |
| 8 | CCGCCCCC | 151 | 8 | CCGCCCCT | 98 |
| 8 | GCCCCGCC | 205 | 8 | CGCCCCGC | 109 |
| 8 | CCGCCCCG | 105 | 8 | CCCCGCCT | 72 |
| 8 | GGCCCCGC | 108 | 8 | CCCCTCCC | 100 |
| 8 | CGCCCCCT | 75 | 8 | CGCCCCCG | 74 |
| 8 | CCCCGCCC | 255 | 9 | CCCGCCCC | 361 |
| 9 | CCCGCCCT | 70 | 9 | CCGCCCCC | 192 |
| 9 | GCCGCCGC | 134 | 9 | CCGCCCCT | 141 |
| 9 | CGCCCCGC | 159 | 9 | CCCCGCCT | 103 |
| 9 | CCGCCCCG | 157 | 9 | CGCCGCCC | 65 |
| 9 | CCGCCGCC | 138 | 9 | TCCGCCCC | 75 |
| 9 | CGCGCCGC | 78 | 9 | CCCCGCCC | 373 |
| 9 | CGCCGCGC | 79 | 9 | CGCCGCCG | 127 |
| 9 | GCGCCGCC | 79 | 10 | GCCGCCGC | 226 |

| | | | | | |
|---|---|---|---|---|---|
| 10 | CCGCCGCT | 59 | 10 | CGCCGCCA | 94 |
| 10 | CCGCCGCC | 241 | 10 | CGCCGCCG | 156 |
| 11 | CCCGCCCC | 72 | 11 | CCCGCCGC | 69 |
| 11 | GCCGCCGC | 104 | 11 | CCGCCGCC | 106 |
| 12 | CCCGCCCC | 87 | 12 | GCCGCCGC | 79 |
| 12 | GCCCCGCC | 64 | 12 | CCGCCCCC | 67 |
| 12 | CCCGCCCC | 73 | 12 | CCCGCCCC | 62 |

**ETS**

| | | | | | |
|---|---|---|---|---|---|
| 9 | GCCGGAAG | 89 | 9 | CCCGGAAG | 62 |
| 9 | CCGGAAGC | 83 | 9 | CGGAAGTG | 81 |
| 9 | CCGGAAGT | 99 | | | |

**NRF-1**

| | | | | | |
|---|---|---|---|---|---|
| 9 | CGCCTGCG | 90 | 9 | GCCTGCGC | 94 |
| 9 | GCGCCTGC | 83 | | | |

**CpG**

| | | |
|---|---|---|
| 9 | GCGCGCGC | 56 |

Table 5.1: Comparison of TFBS motif occurrences within two vertebrate clusters. Start from left most column, it represents position of $n$-th DNA fragment which starts from $-1000$ to $-901$, its motif, and the total number of times the motif occur.

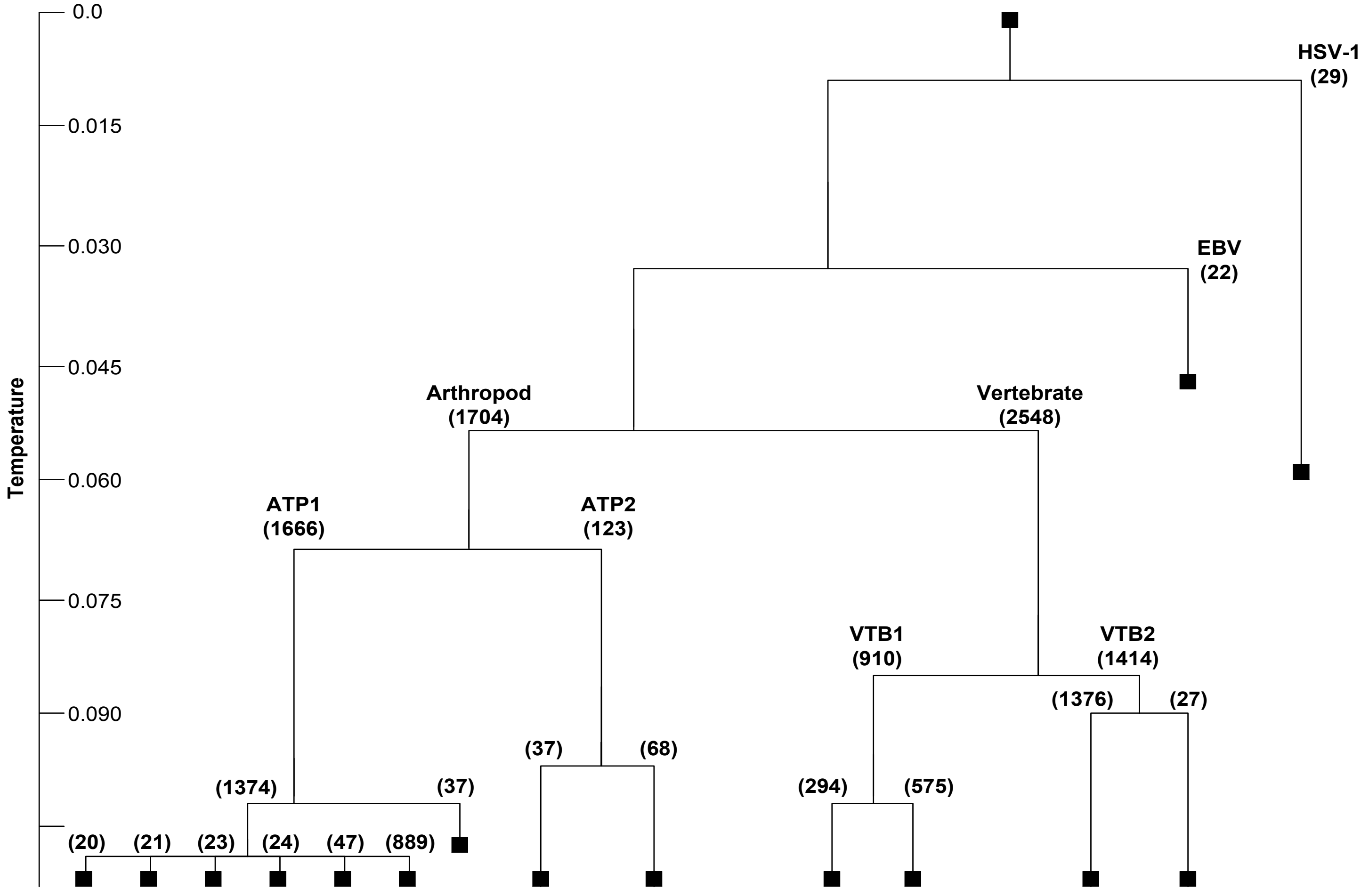

Figure 5.6: Dendrogram of clusters based on simulation with different temperature parameter.

### 5.2.3 Details of Clustering

#### Cluster HSV-1

With size 29 this cluster is completely formed by DNA sequences from human viral genome HSV-1, the pathogen that causes cold sores disease. The genes consist of sequences which are involved during the lytic cycle of HSV-1 infection from immediate early (alpha), delayed early (beta), intermediate late (beta/gamma), and late phases (gamma).

#### Cluster EBV

It comprises of 22 members from EBV viral genes. The EBV virus, which also falls under the herpesvirus family, has two types of genome that express proteins in either the productive lytic or the latent cycle of infection. During latent infection only a few viral genes are expressed [36]. As a result, the number of genes established in latent infection are much smaller (3 genes) instead those in lytic phases (19 genes).

#### Cluster VTB1

This cluster consists of DNA sequences from different species such as *Gallus gallus* (chicken) , *Mus musculus* (house mouse), *Rattus norvegicus* (rat), and *Homo sapiens* (human). The sequences can be categorized into: **a**.) small nuclear RNA, **b**.) structural protein, **c**.) storage, transport proteins and apoproteins, **d**.) enzymes, **e**.) hormones, growth factors, regulatory proteins, **f**.) proteins related to stress, antibody or pathogen defense, **g**.) all transposable elements and retroviruses, especially for long terminal repeats, **h**.) all human viral genes from adenoviruses, and. **i**.) a small number of unclassified chromosomal genes with various functions. The rest (≈4%) are chromosomal genes regulating synthesis of structural proteins, storage and transport proteins, enzymes, antibodies, and hormones from misclassified arthropods.

### Cluster VTB2

This cluster has many genes involved in basic functions needed for the sustenance of the cell. About 99% of the genes come from human. The sequences are dominated by unclassified function chromosomal genes, which cover a total of 1293 entries or approximately 92% of cluster density. Some of their functions which are constitutively expressed include those associated with ribosomal protein, apolipoproteins, molecules transport, molecules binding, different families of globin, hypothetical protein, inhibitors, protease, and zinc-finger protein.

### Cluster ATP1

This is the main cluster of the arthropod class, with almost 100% of the clusters members formed by DNA sequences from fruit fly (*Drosophila*) families. It has similar structure composition in gene expression function as VTB1 and VTB2 such as structural proteins, storage and transport protein, enzymes, hormones and growth factor, and unclassified function chromosomal genes. The members have combinations of a huge number of genes for general metabolism and a fair amount of genes for specific functionality.

**Cluster ATP2**

This cluster contains mostly DNA sequences from fruit fly (*Drosophila melanogaster*). It has 110 sequences of unclassified function, with diverse functionality such as those responsible for synthesis of histone, protein kinase, serine/threonine, and hypothetical protein. Others include genes regulating oxidoreductase in both adult and larva forms of the fruit fly; a type of alcohol dehydrogenase protein useful for catalysis of oxidation-reduction. These genes are absent in the main arthropod cluster and explain the reason behind ATP2 separation from the ATP1 cluster.

# Chapter 6

# Conclusion

In this work, we performed clustering with Super Paramagnetic Clustering (SPC), a novel clustering algorithm based on physical properties of an inhomogeneous granular ferromagnet. This method utilizes Swendsen-Wang cluster Monte Carlo simulations to distinguish clusters by measuring pairs of correlation function from different resolutions.

The method is first tested on the Iris data and reliability of the method is confirmed. We obtained three clusters from the simulation; each cluster corresponds to one of three distinct specimens of Iris flower (Setosa, Virginica, and Versicolor). The SPC correctly classified around 95% of the data with small error.

The main problem is to mine the correlated behavior of gene sequences. Applying SPC to 4541 DNA sequences containing active promoter regions from vertebrate and arthropod classes (including their viral genes), we manage to find high associations in terms of gene function and classes division in the revealed clusters that shows the success rate of this method. From the simulation result, two strongly separated clusters of human viral genes corresponding to the Epstein-Barr virus and the Herpes Simplex virus type 1 were revealed. In addition, vertebrate and arthropod sequences were successfully separated into two different classes with merely 9.25% of arthropod sequences being misclassified. By tuning a clustering parameter, Super Paramagnetic Clustering managed to classify vertebrate class even further into two major clusters, from where a number of housekeeping genes and tissue-specific genes were found respectively. The indications came from observation of gene expression function and consensus transcription factors which were found grouped together in specific positions of the DNA sequences.

## 6.1 Future Directions

- Due to limited number of gene sequences of certain class, we used only two different types of DNA sequences, arthropod and vertebrate class. In future work, more sequences may be included in the dataset as long as the number is sufficient to represent a particular class. A better performance could be achieved by modifying the way features being measured. Some statistical technique such as feature selection or weighting can be applied based on the importance of the features.

- We also did some test of SPC for certain families of protein on another learning method called semi-supervised learning. The idea of semi-supervised is to incorporate a working set of unlabelled data (unsupervised) and a training set of labeled data (supervised) to obtain better clustering with limited supervision. The output unlabeled data that is being clustered together by SPC is combined with labeled data and processed with support vector machine. Initial result was promising with accuracy 87.29% for sensitivity and 89.13% for specificity.

- Implementation wise, to speed up the performance of SPC method, it has been proposed that initial configuration of Potts spin at $T_{n+1}$ follow the last configuration at $T_n$. We did not implement this technique in our main code, instead for every new sampled temperature $T$, we randomly generate a new spin configuration and discard about 10% of initial data before we start collecting statistical sample.

# Bibliography

[1] Abe, T., Kanaya, S., Kinouchi, M., Ichiba, Y., Kozuki, T., and Ikemura, T. 2003. Informatics for Unveiling Hidden Genome Signatures. *Genome Research.* **13**: 693-702.

[2] Access Excellence @ the National Health Museum. 1999. <http://www.accessexcellence.org>

[3] Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale D.A., O'Donovan, C., Redaschi, N., and Yeh, L.S. 2005. The Universal Protein Resource (UniProt). *Nucleic Acids Res.* **33**:D154-159.

[4] Berg, J.M., Tymoczko, J.L., Stryer, L. 2002. Biochemistry, 5th edn. W.H. Freeman and Company. New York

[5] Borra, S., Rocci, R. Vichi, M., and Schuder, M. Advances in Classification and Data Analysis. Springer-Verlag. Heidelberg.

[6] Binder, K., and Heermann, D.W. 2002. Monte Carlo Simulation in Statistical Physics: An Introduction, 4th edn. Springer-Verlag. Heidelberg.

[7] Dietterich, T. Machine Learning. 2003. in Nature Encyclopedia of Cognitive Science. Macmillan. London.

[8] Blatt, M., Wiseman, S., and Domany, E. 1996. Super-paramagnetic Clustering of Data. *Physical Review Letters*. **76**: 3251-3255.

[9] Blatt, M., Wiseman, S., and Domany, E. 1996. Clustering Data through an Analogy to the Potts Model. *Advances in Neural Information Processing System*. **8**: 416-422.

[10] Blatt, M., Wiseman, S., and Domany, E. 1997. Data Clustering Using a Model Granular Magnet. *Neural Computation*. **9**: 1805-1842.

[11] Blatt, M., Wiseman, S., and Domany, E. 1998. Super-paramagnetic Clustering of Data. *Physical Review E*. **57**: 3767-3783.

[12] Butler, J.E.F., and Kadonaga, J.T. 2002. The RNA Polymerase II Core Promoter: A Key Component in The Regulation of Gene Expression. *Genes and Development*. **16**: 2583-2592.

[13] Chuzhanova, N. A., Jones, A. J., and Margetts, S. 1998. Feature Selection for Genetic Sequence Classification. *Bioinformatics*. **14**: 139-143.

[14] Courey, A. J., and R. Tjian. 1988. Analysis of Sp1 in Vivo Reveals Multiple Transcriptional Domains, Including a Novel Glutamine-Rich Activation Motif. *Cell.* **55**: 887-898.

[15] Dynan, W. S., and R. Tjian. 1983. The Promoter-Specific Transcription Factor Sp1 Binds to Upstream Sequences in The SV40 Early Promoter. *Cell.* **35**: 79-87.

[16] Domany, E. 2003. Cluster Analysis of Gene Expression Data. *J. Statistical Physics.* **110**: 1117-1139.

[17] Eisenberg, E., and Levanon, E.Y. 2003. Human Housekeeping Genes are Compact. *Trends in Genetics.* **19(7)**: 362-365.

[18] Faisst, S., and Meyer, S. 1992. Compilation of Vertebrate-Encoded Transcription Factors. *Nucleic Acids Res.* **20**: 3-26.

[19] FitzGerald, P.C., Shlyakhenko, A., Mir, A.A., Vinson, C. 2004. Clustering of DNA Sequences in Human Promoters. *Genome Research.* **14**: 1562-1574.

[20] Friedman, H.P., Rubin, J. 1967. On Some Invariant Criterion for Grouping Data. *Journal of The American Statistical Association* **63**:1159-1178.

[21] Fung,G. A Comprehensive Overview of Basic Clustering Algorithms.

[22] Getz, G., Levine, E., Domany, E., and Zhang, M.Q. 2000. Super-paramagnetic Clustering of Yeast Gene Expression Profiles. *Physica A.* **279**: 457-464.

[23] Getz, G., Levine, E., Domany, and Domany, E. 2000. Coupled Two-Way Clustering Analysis of Gene Microarray Data. *PNAS*. **97**: 12079-12084.

[24] Getz, G., Gal, H., Notterman, D.A., and Domany, E. 2003. Coupled Two-Way Clustering Analysis of Breast Cancer and Colon Cancer Gene Expression Data. *Bioinformatics*. **19**: 1079-1089.

[25] Everitt, B.S. Cluster Analysis. 1974. John Wiley & Sons. New York.

[26] Gnanadesikan, R., and Kettenring, J.R. 1995. Weighting and Selection of Variables for Cluster Analysis. *Journal of Classification.* **12 (1)**: 113-136.

[27] Graepel, T., Statistical Physics of Clustering Algortihms. 1998. Technical Report 171822, FB Physik, Institut fur Theoretische Physik.

[28] Hsiao, L.-L., Dangond, F., Yoshida, T.,*et al.* 2001. A Compendium of Gene Expression in Normal Human Tissues. *Physiol. Genomics* **7**: 97 104.

[29] Jain, A.K., and Dubes, R.C. 1988. Algorithms for Clustering Data. Prentice Hall. New Jersey.

[30] Kadonaga, J. T., K. R. Carner, F. R. Masiarz, and R. Tjian. 1987. Isolation of cDNA Encoding Transcription Factor Sp1 and Functional Analysis of The DNA Binding Domain. *Cell.* **51**: 1079-1090.

[31] Kaelbling, L.P., Littman, M.L., and Moore, A.W. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intellegence Research.* **4**: 237-285.

[32] Karlin, S. and Burge, C. 1995. Dinucleotide Relative Abundance Extremes: A Genomic Signature. *Trends Genet.* **11**: 283-290.

[33] Karlin, S. and Ladunga, I. 1994. Comparisons of Eukaryotic Genomic Sequences. *Proc Natl Acad Sci U S A.* **91**: 12832-12836.

[34] Latchman, D.S. 2004. Eukaryotic Transcription Factors 4th Ed. Academic Press, London.

[35] Larsen, F., Gundersen, G., Lopez, R., and Prydz, H. 1992. CpG Islands as Gene Markers in The Human Genome. *Genomics.* **13(4)**: 1095-1107.

[36] Laux, G., Perricaudet, M., and Farrell, P.J. 1988. A spliced Epstein-Barr Virus Gene Expressed in Immortalized Lymphocytes is Created by Circularization of The Linear Viral Genome. *EMBO Journal.* **7(3)**: 769-774.

[37] Ma, X.T., Qian, M.P., Tang, H.X. 2004. Predicting Polymerase II Core Promoters by Cooperating Transcription Binding Sites in Eukaryotic Genes. *Acta Biochimica et Biophysica Sinica.* **36(4)**: 250-258.

[38] Mariño-Ramírez, L., Spouge,J.L., Kanga, G.C., Landsman, D. 2004. Statistical Analysis of Over-Represented Words on Human Promoter Sequences. *Nucleic Acids Res.* **32(3)**:949-958.

[39] Milligan, G.W., and Cooper, M.C. 1988. A Study of Standardization of Variables in Cluster Analysis. *Journal of Classification.* **5 (2)**: 181-204.

[40] Murakami, K., Kojima, T., and Sakaki, Y. 2003. Detection of Tissue Specifiv Genes by Putative Regulatory Motifs in Human Promoter Sequences. *Genome Informatics.* **14**: 408-409.

[41] Nakamura, Y., Miura, K., Fujino, Y., Iwao, H., Ogita, S., and Yamanaka, S. 2000. Evolution, Structure, and Expression of GNPI/Oscillin Orthologous Genes. *Genomics.* **68**: 179-186.

[42] Nakashima, H., Ota, M., Nishikawa, K., and Ooi, T. 1998. Genes from Nine Genomes are Separated into Their Organisms in the Dinucleotide Composition Space. *DNA Res*, **5**: 251-259.

[43] Niedermayer, F. 1990. Improving the Improved Estimators in O(N) Spin Model. *Physical Review Letters.* **B237**: 473-475.

[44] Nussinov, R. 1984. Strong Doublet Preferences in Nucleotide Sequences and DNA Geometry. *J Mol Evol.* **20**: 111-119.

[45] Perner, P. 2002. Data Mining on Multimedia Data. Springer-Verlag. Heidelberg.

[46] Philipsen, S., and G. Suske. 1999. A Tale of Three Fingers: The Family of Mammalian Sp/XKLF Transcription Factors. *Nucleic Acids Res.* **27**: 2991-3000.

[47] Schmid, C.D., Praz, V., Delorenzi, M., Prier, R., and Bucher, P. 2004. The Eukaryotic Promoter Database EPD: The Impact of in Silico Primer Extension. *Nucleic Acids Res.* **32**: D82-85.

[48] Sutton, R.S., and Barto, A.G., Reinforcement Learning: An Introduction. 1988. MIT Press, Cambridge, MA

[49] Swendsen, R.H., and Wang, J.-S. 1987. Nonuniversal Critical Dynamics in Monte Carlo Simulation. *Physical Review Letters.* **58(2)**: 86-88.

[50] Tetko, I.V., Facius, A., Ruepp, A., and Mewes, H.W. 2005. Super Paramagnetic Clustering of Protein Sequences *BMC Bioinformatics.* **6**: 82.

[51] Trinklein, N.D., Aldred, S.J.F., Saldanha, A.J., and Myers, R.M. 2003. Identification and Functional Analysis of Human Transcriptional Promoters. *Genome Research.* **13**: 308-312.

[52] Wang, J.-S., and Swendsen, R.H. 1990. Cluster Monte Carlo Algorithms. *Physica A.* **167**: 565-579.

[53] Weis, L. and Reinberg, D. 1992. Transcription by RNA Polymerase II Initiator Directed Formation of Transcription Competent Complexes. *FASEB Journal.* **6**: 3300-3309.

[54] Whitson, D.L., and Amstutz, D.A. 1997. Accessing Information in a Technological Age. Krieger Publishing Company. Malabar, Fla

[55] Wille, L.T (ed). 2004. New Directions in Statistical Physics: Econophysics, Bioinformatics, and Pattern Recognition. Springer-Verlag. Heidelberg.

[56] Wingender, E., Chen, X., Fricke, E., Geffers, R., Hehl, R., Liebich, I., Krull, M., Matys, V., Michael, H., Ohnhuser, R., Pr, M., Schacherer, F., Thiele, S. and Urbach, S. 2001. The TRANSFAC system on gene expression regulation. *Nucleic Acids Res.* **29**: 281-283.

[57] Witten, I.H., and Frank, E. 1999. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.

[58] Wu, F.Y., The Potts Model. 1982. *Reviews of Modern Physics.* **54(1)**:235-268

[59] Valpola, H. 2000. Bayesian Ensemble Learning for Nonlinear Factor Analysis. Acta Polytechnica Scandinavica, Mathematics and Computing Series No.108. Finnish Academies of Technology.

[60] Yamabe, Y., Shimamoto, A., Goto, M., Yokota, J., Sugawara, M., and Furuichi, Y. 1998. Sp1-Mediated Transcription of the Werner Helicase Gene Is Modulated by Rb and p53. *Molecular and Cellular Biology.* **18(11)**: 6191-6200.

# Appendix

The SPC algorithm is implemented in C language and divided into two parts. First part of the program "nbors.c" is straightforward. It reads data from a file and finds the list of neighbors based on user specified input. (e.g number of mutual-neighbors $K$ and whether edges from Minimum Spanning Tree (MST) are imposed). If edges from MST are included, Kruskal algorithm is used to find a connected graph. Its edges are then added to initial list of mutual neighbors. The program then writes the list of neighbors in a separate output file.

The main part of SPC algorithm is executed in "SPC.c". From first output file, we have a list of neighbors for each data point. From here the interaction strength can be directly calculated and Monte Carlo simulation with Swendsen-Wang algorithm is initiated. To identify SW and final clusters, we implement Hoshen-Kopelman algorithm for assigning label to connected subgraphs.

```
/*************************************************************/
/* Written by : Sugiarto Radjiman                            */
/* File name  : nbors.c                                      */
/* Project    : Super Paramagnetic Clustering                */
/* Function   : Find List of Neighbors                       */
/*************************************************************/

#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#include "nrutil.h"
#include "nrutil.c"
#include "quicksort.c"

#define N   150              // # of Vectors
#define DM  4                // # of Dimensions
#define K   5                // # of Mutual Neighbor
#define H   50               // # of Additional List for Neighbors
int sort;                    // Sort Neighbors List
int mst ;                    // Include Neighboring Points from MST
int R;                       // Buffer for MST
int interval;                // Distance Interval Division


void find_neighbors(double **data, int **nbors);
void MST(int *idx1, int *idx2, double th_d, double th_u, double max_dist,
         double **data, int *graph, double *dist, int *mark, int **idx,
         int **edge);


main(void)
{

    int    i, j, k, idx,
           **nbors;
    double mindist, maxdist,
           **data, *temp;
    FILE *fp1, *fp2;
```

```
sort     = 0;
mst      = 1;
R        = 1000;
interval = 50;

/* READ DATA FROM FILE
   =================== */
data  = dmatrix(1,N,1,DM);
fp1 = fopen("irisN.dat","r");
if(fp1 == NULL){
   printf("Unable to open file for reading\n");
   exit(1);
}

i = 1; j=1;
while(fscanf(fp1,"%lf ", &data[i][j])!=EOF){
     j++;
     if(j==DM+1){
        i++;
        j = 1;
     }
}
fclose(fp1);

/* FIND NEIGHBORS
   =============== */
nbors = imatrix(1,N,1,K+H);
for(i=1; i<=N; i++){
    for(j=1; j<=K+H; j++){
        nbors[i][j] = 0;
    }
}
find_neighbors(data, nbors);
printf("Find Neighbors Complete\n");

/* SORT NEIGHBORS
   =============== */
temp = dvector(1,K+H);
for(i=1; i<=N; i++){
    for(j=1; j<=K+H; j++){
        temp[j] = nbors[i][j];
        nbors[i][j] = 0;
    }

    if(sort==0){    // Neighbors Not Sorted
       idx = 1;
       for(j=1; j<=K+H; j++){
           if(temp[j]!=0){
              nbors[i][idx] = temp[j];
              idx++;
           }
       }
    }

    if(sort==1){   // Neighbors Sorted
       quicksort(K+H,temp);
       for(j=1; j<=K+H; j++){
           if(temp[j]!=0)
              break;
       }
       idx = 1;
       for(k=j; k<=K+H; k++){
           nbors[i][idx] = (int) temp[k];
           idx++;
       }
       for(k=idx; k<=K+H; k++){
           nbors[i][k] = 0;
       }
    }
}
free_dvector(temp,1,K+H);


/* WRITE RESULT TO FILE
   ==================== */
k = K;
for(i=1; i<=N; i++){
    for(j=K+1; j<=K+H; j++){
        if(nbors[i][j]==0 && j>k){
           k = j-1;
           break;
        }
```

```
            if(nbors[i][j]==0 && j<k){
               break;
            }
        }
    }
    printf("MAX # of Neighbors is %d\n", k);
    fp2 = fopen("nbors_irisN.dat","w");
    for(i=1; i<=N; i++){
        for(j=1; j<=k; j++){
            fprintf(fp2, "%d ", nbors[i][j]);
        }
        fprintf(fp2, "\n");
    }
    fclose(fp2);

}

void find_neighbors(double **data, int **nbors)
{


     int    i, j, ii, jj, m, n, k,
            *idx1, *idx2, *graph, *vtx, *mark, *elist, **edge, **idx;
     double sum, th_d, th_u, inc, max_dist, temp,
            *dist;


     dist = dvector(1,N);
     vtx  = ivector(1,N);
     for(i=1; i<=N; i++){
         dist[i] = 0.0;
         vtx[i]  = 0;
     }


     sum = max_dist = 0.0;                  // Find K-Nearest Neighbors
     for(i=1; i<=N; i++){
         temp = 0.0;
         for(j=1; j<=N; j++){
             for(k=1; k<=DM; k++){
         sum += (data[i][k]-data[j][k])*(data[i][k]-data[j][k]);
        }
dist[j] = sqrt(sum);
      if(dist[j] > max_dist){
   max_dist = dist[j];
}
vtx[j]  = j;
sum = 0.0;
    }
    quicksort2(N, dist, vtx);

    if(vtx[1]!=i){
       for(j=2; j<=N; j++){
           if(vtx[j]==i){
              temp   = vtx[1];
              vtx[1] = vtx[j];
              vtx[j] = temp;
              break;
           }
       }
    }

    for(j=1; j<=K; j++)
        nbors[i][j] = vtx[j+1];

}
printf("Max distance = %f\n", max_dist);
free_dvector(dist,1,N);
free_ivector(vtx, 1,N);


for(i=1; i<=N; i++){                   // Find K-Mutual Neighbors
    for(j=1; j<=K; j++){
        m = nbors[i][j];
        for(k=1; k<=K; k++){
            if(i==nbors[m][k]){
               break;
            }
        }

        if(k==K+1){
           nbors[i][j] = 0;
        }
    }
}

if(mst == 1){
   graph = ivector(1,N);              // Find Minimum Spanning Tree
   dist  = dvector(1,R);              // w/ Kruskal's Algorithm
   mark  = ivector(1,R);
   elist = ivector(1,N);
   idx1  = ivector(1,1);
   idx2  = ivector(1,1);
   idx   = imatrix(1,2,1,R);
   edge  = imatrix(1,2,1,N-1);
   printf("Memmory Allocation Complete\n");


   for(i=1; i<=N-1; i++){
       graph[i] = 0;
       elist[i] = 0;
       edge[1][i] = 0;
       edge[2][i] = 0;
   }
   graph[N] = 0;
   elist[N] = 0;

   inc = max_dist/(double) interval;
   printf("inc = %f\n", inc);
   idx1[1] = idx2[1] = 1;
   for(k=1; k<=interval; k++){
       m = 0;
       th_d = inc*(k-1);
       th_u = inc*k;
       MST(idx1, idx2, th_d, th_u, max_dist, data, graph, dist,
           mark, idx, edge);
       printf("# Cluster Idx = %d # Edges = %d\n", idx1[1], idx2[1]-1);
       printf("MST %d complete \n\n", k);

       for(i=1; i<=N; i++){
           if(graph[i]!=1){
              m++;
           }
    }
    printf("Unconnected = %d data\n",m);

    if(m==0){                      // Graph are Connected (MST Found)
       printf("Connected Graph Found\n");
       break;
    }
}
free_ivector(graph,1,N);
free_dvector(dist,1,R);
printf("idx2 = %d N-1 = %d\n", idx2[1], N-1);


for(i=1; i<=N; i++){               // Find Empty Place in Array nbors
    for(j=1; j<=K+H; j++){
        if(nbors[i][j]==0){
           elist[i] = j;
           break;
        }
    }
}

for(i=1; i<=N-1; i++){             // Update List of Neighbors
    m = edge[1][i];
    n = edge[2][i];
    for(j=1; j<=K+H; j++){
        if(nbors[m][j]==n){        // Neighbors in The List
           break;
        }
    }
    if(j==K+H+1){                  // Neighbors not in The List
       ii = elist[m];
       nbors[m][ii] = n;
       for(k=ii+1; k<=K+H; k++){
           if(nbors[m][k]==0){
              elist[m] = k;
              break;
           }
       }
       if(k==K+H+1){
          printf("Set larger array for neighbors !\n");
```

```
                exit(1);
             }

             jj = elist[n];
             nbors[n][jj] = m;
             for(k=jj+1; k<=K+H; k++){
                 if(nbors[n][k]==0){
                    elist[n] = k;
                    break;
                 }
             }
             if(k==K+H+1){
                printf("Set larger array for neighbors !\n");
                exit(1);
             }

          }
       }
       free_ivector(mark,1,R);
       free_imatrix(idx,1,2,1,R);
       free_imatrix(edge,1,2,1,N-1);

    }
}

void MST(int *idx1, int *idx2, double th_d, double th_u, double max_dist,
         double **data, int *graph, double *dist, int *mark, int **idx,
         int **edge)
{

    int i, j, k, m, ii, jj, I, J,
        set, min;
    double sum, temp;


    for(i=1; i<=R; i++){
        mark[i] = i;
        dist[i] = 0.0;
        idx[1][i] = 0;
        idx[2][i] = 0;
    }
```

```
m = 1;
sum = temp = 0.0;
printf("Cluster Idx = %d # Edges = %d\n", idx1[1], idx2[1]-1);
for(i=1; i<=N; i++){
    for(j=i; j<=N; j++){
        for(k=1; k<=DM; k++){
    sum += (data[i][k]-data[j][k])*(data[i][k]-data[j][k]);
}
temp = sqrt(sum);
sum = 0.0;
if(temp>=th_d && temp<th_u && i!=j){
   dist[m] = temp;
   idx[1][m] = i;
   idx[2][m] = j;
   m++;
}
    }
    if(m > R) break;
}

printf("Buffer = %d R = %d\n", m, R);
printf("th_d = %f th_u = %f\n", th_d, th_u);
if(m > R){
   printf("Use Larger Array or Decrease Threshold Increament\n");
   exit(1);
}

for(i=m; i<=R; i++)
    dist[i] = max_dist;
quicksort2(R, dist, mark);

for(i=1; i<=m-1; i++){
    ii = idx[1][mark[i]];
    jj = idx[2][mark[i]];

    if(graph[ii]==0 && graph[jj]==0){
       graph[ii] = idx1[1];
       graph[jj] = idx1[1];
       edge[1][idx2[1]] = ii;
       edge[2][idx2[1]] = jj;
```

```
            idx1[1]++;
            idx2[1]++;
         }
         else if(graph[ii]==0 && graph[jj]!=0){
            graph[ii] = graph[jj];
            edge[1][idx2[1]] = ii;
            edge[2][idx2[1]] = jj;
            idx2[1]++;
         }
         else if(graph[ii]!=0 && graph[jj]==0){
            graph[jj] = graph[ii];
            edge[1][idx2[1]] = ii;
            edge[2][idx2[1]] = jj;
            idx2[1]++;
         }
         else if(graph[ii]!=graph[jj]){
            if(graph[ii] > graph[jj]){
               min = graph[jj];
               set = graph[ii];
            }
            else{
               min = graph[ii];
               set = graph[jj];
            }
            for(j=1; j<=N; j++){
                if(graph[j]==set)
                   graph[j] = min;
            }
            edge[1][idx2[1]] = ii;
            edge[2][idx2[1]] = jj;
            idx2[1]++;
         }
         else{
            continue;
         }
     }

}
```

```
/************************************************************/
/* Written by : Sugiarto Radjiman                          */
/* File name  : SPC.c                                      */
/* Project    : Super Paramagnetic Clustering              */
/* Function   : 1. MC Simulation w/ SW algorithm           */
/*              2. Measurement of Thermodynamics Quantities */
/*              3. Clustering Data                         */
/************************************************************/

#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "nrutil.h"
#include "nrutil.c"
#include "quicksort.c"

#define N  150             // # of vectors
#define DM 4               // # of dimensions
#define NB 6               // # of MAX Neighbors
#define B  5               // # of Biggest Clusters
#define S  100             // # of Total simulation (w/ different T)
#define M  500000          // # of Monte Carlo simulation
#define D  100000          // # of discarded simulation
#define EPS 1.0e-300

int    q;
double T, T0, Tmax, Tinc, th;


void interaction_matrix(int **nbors, double **J, double **dist, double **data);
double SW(int q, double T,
          int **nbors, int **bond, int *cluster, double **J, double **G);
int HK(int *cluster, int **nbors, int **bond);
int final_cluster(int k, int q, double th,
          int **nbors, int **bond, int *cluster, int **cluster_data, double **G);
void find_ClusterSize(int k, int *cluster, int *fcluster, int **cluster_size);
```

```
main(void)
{


    int    i, j, k, m, seed, idx,
           *cluster, *fcluster, **bond, **cluster_size, **cluster_data, **nbors;

    double **data, **dist, **J, **G, *G_max, *X, *temp;

    FILE *fp1, *fp2, *fp3, *fp4;


   /* PARAMETER
      ========= */
    q     = 20;                // # of different Potts State Variable
    T0    = EPS;               // Starting Temperature
    Tmax  = 0.07;              // Highest  Temperature
    Tinc  = (Tmax-T0)/S;       // Temperature Increament
    th    = 0.5;               // Threshold Value


    /* READ DATA FROM FILE
       ================== */
    data  = dmatrix(1,N,1,DM);
    fp1 = fopen("irisN.dat","r");
    if(fp1 == NULL){
       printf("Unable to open file for reading\n");
       exit(1);
    }

    i = 1; j=1;
    while(fscanf(fp1,"%lf ", &data[i][j])!=EOF){
         j++;
         if(j==DM+1){
            i++;
            j = 1;
         }
    }
    fclose(fp1);
    printf("Reading Data File Complete\n");
```

```
nbors  = imatrix(1,N,1,NB);
fp1 = fopen("nbors_irisN.dat","r");
if(fp1 == NULL){
   printf("Unable to open file for reading\n");
   exit(1);
}

i = 1; j=1;
while(fscanf(fp1,"%d ", &nbors[i][j])!=EOF){
     j++;
     if(j==NB+1){
        i++;
        j = 1;
     }
}
fclose(fp1);
printf("Reading Neighbor List File Complete\n");


/* CALCULATE INTERACTION MATRIX J
   ============================= */
dist   = dmatrix(1,N,1,NB);
J      = dmatrix(1,N,1,NB);

interaction_matrix(nbors, J, dist, data);
free_dmatrix(data ,1,N,1,DM);
free_dmatrix(dist ,1,N,1,NB);
printf("Calculate Interaction Matrix Complete\n\n");


X            = dvector(1,S+1);
fcluster     = ivector(1,S+1);
cluster      = ivector(1,N);
G            = dmatrix(1,N,1,NB);
bond         = imatrix(1,N,1,NB);
cluster_size = imatrix(1,S+1,1,B);
cluster_data = imatrix(1,N,1,S+1);


//srand48((unsigned int) time (NULL));
//printf("seed = %d\n", (unsigned int) time (NULL));
```

```
seed = 1083354197;
srand48(seed);


for(k=1; k<=S+1; k++){

    /* CLUSTERING ALGORITHM
       ==================== */
    T = T0 + Tinc*(k-1);
    X[k] = SW(q, T, nbors, bond, cluster, J, G);
    printf("T = %f  X[%d] = %f\n", T, k, X[k]);


    /* FINAL DATA CLUSTERS
       =================== */
    fcluster[k] = final_cluster(k, q, th, nbors, bond, cluster,
                                cluster_data, G);
    printf("# of clusters[%d] = %d\n", k, fcluster[k]);


    /* FIND BIGGEST CLUSTERS
       ===================== */
    find_ClusterSize(k, cluster, fcluster, cluster_size);

}


/* WRITE RESULT TO FILE
   ==================== */
fp2 = fopen("data_irisN.dat","w");
for(i=1; i<=S+1; i++){
    fprintf(fp2, "%f  %f  %d ", T0 + Tinc*(i-1), X[i], fcluster[i]);
    for(j=1; j<=B; j++){
        fprintf(fp2, "%d  ", cluster_size[i][j]);
    }
    fprintf(fp2, "\n");
}
fclose(fp2);

fp3 = fopen("cluster_irisN.dat","w");
for(i=1; i<=N; i++){
        for(j=1; j<=S+1; j++){
            fprintf(fp3, "%d  ", cluster_data[i][j]);
        }
        fprintf(fp3, "\n");
    }
    fclose(fp3);

    /*fp4 = fopen("G.dat","w");
    for(i=1; i<=N; i++){
        for(j=1; j<=NB; j++){
            fprintf(fp4, "%f ", G[i][j]);
        }
        fprintf(fp4, "\n");
    }
    fclose(fp4);*/


}


void interaction_matrix(int **nbors, double **J, double **dist, double **data)
{


     int i, j, k, m, n, num_nbors;
     double sum, temp, K_hat, a,
            sum_dist, min_dist, max_dist, Jmin, Jmax, Jmean;


     num_nbors = 0.0;
     sum_dist  = 0.0;
     max_dist  = Jmax = 0.0;
     min_dist  = Jmin = 100.0;
     Jmean = 0.0;

     for(i=1; i<=N; i++){
        for(j=1; j<=NB; j++){
            dist[i][j] = 0.0;
            J[i][j]    = 0.0;
        }
     }
```

```
for(i=1; i<=N; i++){
    for(j=1; j<=NB; j++){
        if(nbors[i][j]!=0){
           sum = 0.0;
           num_nbors++;
           n = nbors[i][j];
           for(k=1; k<=DM; k++){
       sum += (data[i][k]-data[n][k])*(data[i][k]-data[n][k]);
           }
   dist[i][j] = sqrt(sum);
           sum_dist += dist[i][j];
   if(dist[i][j] < min_dist)
      min_dist = dist[i][j];
   if(dist[i][j] > max_dist)
      max_dist = dist[i][j];
        }
    }
}
K_hat = (double) num_nbors/N;         // Average # of Neighbors
a     = (double) sum_dist/num_nbors;  // Average distances of
                                      // neighboring points

for(i=1; i<=N; i++){
    for(j=1; j<=NB; j++){
        if(nbors[i][j]==0){
           J[i][j] = 0.0;
        }
        else{
           J[i][j] = 1/K_hat*exp(-dist[i][j]*dist[i][j]/(2*a*a));
           Jmean += J[i][j];
           if(J[i][j]<Jmin)
              Jmin = J[i][j];
           if(J[i][j]>Jmax)
              Jmax = J[i][j];
        }
    }
}
Jmean = Jmean/num_nbors;
printf("Total # Neighbors    = %d\n", num_nbors);
printf("Total # Interactions = %d\n", num_nbors/2);
printf("Sum Distance = %f\n", sum_dist);
```

```
     printf("a = %f K_hat = %f \n",a, K_hat);
     printf("Min Dist = %e\n", min_dist);
     printf("Max Dist = %f\n", max_dist);
     printf("Jmin  = %e\n",Jmin);
     printf("Jmax  = %f\n",Jmax);
     printf("Jmean = %f\n",Jmean);


}


double SW(int q, double T,
          int **nbors, int **bond, int *cluster, double **J, double **G)
{


    int i, j, k, n, kk, kkmax, idx, itr, ns,
        Nmax, num_cluster, avg_cluster,
        *s, *Ns;
    double p, m, m_mean1, m_mean2, **C;

    s     = ivector(1,N);
    Ns    = ivector(1,q);
    C     = dmatrix(1,N,1,NB);

    m = p = 0.0;
    m_mean1 = m_mean2 = 0.0;
    num_cluster = avg_cluster = 0;

    for(i=1; i<=N; i++){                      // Initial Spin Variable
        for(j=1; j<=NB; j++){
            C[i][j]    = 0.0;
            G[i][j]    = 0.0;
            bond[i][j] = 0;
        }
        s[i] = drand48()*q + 1;
    }


    //==========================================
itr = 0;
kk = 1;
kkmax = M-D+1;
for(; ;){
    if(kk==kkmax) break;
    for(i=1; i<=N; i++){        // Bond exist with certain probability
        for(j=1; j<=NB; j++){  // for neighbors with same state Potts
            bond[i][j] = 0;    // variable
        }
    }
    for(i=1; i<=N; i++){
        for(j=1; j<=NB; j++){
            if(nbors[i][j]==0){
               break;
            }
            n = nbors[i][j];
            if(s[i]==s[n] && i<n){
               p = 1-exp(-J[i][j]/T);
               if(drand48()<=p){
                  bond[i][j] = 1;
                  for(k=1; k<=NB; k++){
                      if(nbors[n][k]==i){
                         bond[n][k] = 1;
                         break;
                      }
                  }
               }
            }
        }
    }


    num_cluster = HK(cluster, nbors, bond);  // Labelling Cluster

    for(i=1; i<=N; i++)                      // Assign New Spin Value
        s[i] = 0;
    for(i=1; i<=N; i++){
        if(s[i] == 0){
           s[i] = drand48()*q + 1;
           for(j=i+1; j<=N; j++){
               if(cluster[j] == cluster[i])
```

```
                  s[j] = s[i];
           }
        }
    }


    itr += 1;
    if(itr > D){
       kk++;
       Nmax = 0;                    // Calculate Magnetization Value
       for(i=1; i<=q; i++)
           Ns[i] = 0;
       for(i=1; i<=N; i++)
           Ns[s[i]]++;

       for(i=1; i<=q; i++){
           if(Ns[i] > Nmax)
              Nmax = Ns[i];
       }
       m = (double) (q*Nmax-N)/((q-1)*N);
       m_mean1 += m;
       m_mean2 += m*m;

       avg_cluster += num_cluster;
       for(i=1; i<=N; i++){
           for(j=1; j<=NB; j++){
               n = nbors[i][j];
               if(n==0){
                  break;
               }
               if(cluster[i] == cluster[n]){
                  C[i][j] += 1.0;
               }
           }

       }
    }

}
// ===========================================
    m_mean1 = m_mean1/(M-D);
    m_mean2 = m_mean2/(M-D);
    printf("avg_cluster = %d\n", avg_cluster);
    avg_cluster = avg_cluster/(M-D);
    printf("<m>  = %f\n", m_mean1);
    printf("<m2> = %f\n", m_mean2);
    printf("Average # of SW clusters = %d\n", avg_cluster);

    for(i=1; i<=N; i++){       // Calculate Spin Correlation Function
        for(j=1; j<=NB; j++){
            C[i][j] = C[i][j]/(M-D);
            G[i][j] = ((q-1) * C[i][j] + 1) / q;
        }
    }
    free_ivector(s,1,N);
    free_ivector(Ns,1,q);
    free_dmatrix(C,1,N,1,NB);
    return (m_mean2-m_mean1*m_mean1);


}


int HK(int *cluster, int **nbors, int **bond)
{



    int i, j, n, ns, idx, ncluster,
        *stack;


    stack = ivector(1,N);
    for(i=1; i<=N; i++){
        stack[i] = 0;
        cluster[i] = 0;
    }

    ncluster = 0;
    for(i=1; i<=N; i++){
        if(cluster[i]==0){        // Point i not labeled
           ns = 1;                // Stack Index
```

```
            ncluster++;
            cluster[i] = ncluster;
            stack[ns] = i;
            while(ns >=1){
                idx = stack[ns];
                ns--;
                for(j=1; j<=NB; j++){
                    n = nbors[idx][j];
                    if(n!=0 && bond[idx][j]==1 && cluster[n]==0){
                       cluster[n] = ncluster;
                       ns++;
                       stack[ns] = n;
                    }
                }
            }
        }
    }
    free_ivector(stack,1,N);
    return ncluster;

}


int final_cluster(int k, int q, double th,
        int **nbors, int **bond, int *cluster, int **cluster_data, double **G)
{


    int i, j, m, n, ii, jj, num_cluster;
    double *Gmax;


    Gmax    = dvector(1,N);


    for(i=1; i<=N; i++){
         for(j=1; j<=NB; j++){
             bond[i][j] = 0;
         }
         Gmax[i] = 0.0;
         cluster[i] = 0;
```

```
     cluster_data[i][k] = 0;
}

for(i=1; i<=N; i++){
    for(j=1; j<=NB; j++){
        if(nbors[i][j]==0){
           break;
        }
        if(G[i][j] >= th){
           bond[i][j] = 1;
           n = nbors[i][j];
           for(m=1; m<=NB; m++){
               if(nbors[n][m]==i){
                  bond[n][m] = 1;
                  break;
               }
           }
        }
        if(G[i][j] > Gmax[i]){
           Gmax[i] = G[i][j];
           ii = j;
           jj = nbors[i][j];
        }
    }
    if(Gmax[i] > (double)1/q){
       bond[i][ii] = 1;
       for(m=1; m<=NB; m++){
           if(nbors[jj][m]==i){
              bond[jj][m] = 1;
              break;
           }
       }
    }
}

 num_cluster = HK(cluster, nbors, bond);
 for(i=1; i<=N; i++){
     cluster_data[i][k] = cluster[i];
 }
 return num_cluster;
```

```
}


void find_ClusterSize(int k, int *cluster, int *fcluster, int **cluster_size)
{


     int i, j;
     double *temp;


     for(i=1; i<=B; i++)                // Find the Size of "B" Biggest
          cluster_size[k][i] = 0;        // Clusters

     if(fcluster[k]==1){
        cluster_size[k][1] = N;
        for(i=2; i<=B; i++)
            cluster_size[k][i] = 0;
     }
     else{
        temp = dvector(1,fcluster[k]);
        for(i=1; i<=fcluster[k]; i++)
            temp[i]  = 0.0;
        for(i=1; i<=N; i++)
            temp[cluster[i]] ++;

        quicksort(fcluster[k], temp);
        j = (fcluster[k]) > B ? B : fcluster[k];
        for(i=1; i<=j; i++)
            cluster_size[k][i] = (int) temp[fcluster[k]+1-i];

     }
     for(i=1; i<=B; i++)
          printf("cluster_size[%d][%d] = %d\n",k,i,cluster_size[k][i]);
     printf("\n");

}
```