Machine Learning Methods for Pattern Analysis and Clustering

By

Ji He

Submitted In Partial Fulfillment Of The
Requirements For The Degree Of
Doctor of Philosophy
at

Department of Computer Science
School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543

September, 2004

Name: Ji He

Degree: Doctor of Philosophy

Department: Department of Computer Science

Thesis Title: **Machine Learning Methods for Pattern Analysis and Clustering**

Abstract: Pattern analysis has received intensive research interests in the past decades. This thesis targets efficient cluster analysis of high dimensional and large scale data with user's intuitive prior knowledge. A novel neural architecture named Adaptive Resonance Theory Under Constraint (ART-C) is proposed. The algorithm is subsequently applied to the real-life clustering problems on the gene expression domain and the text document domain. The algorithm has shown significantly higher efficiency over other algorithms in the same family. A set of evaluation paradigms are studied and applied to evaluate the efficacy of the clustering algorithms, with which the clustering quality of ART-C is shown to be reasonably comparable to those of existing algorithms.

Keywords: Pattern Analysis, Machine Learning, Clustering, Neural Networks, Adaptive Resonance Theory, Adaptive Resonance Theory Under Constraint.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Pattern Analysis: the Concept

Pattern, originally as *patron* in Middle English and Old French, has been a popular word ever since sometime before 1010 [Mor88]. Among its various definitions listed in the very early Webster's Revised Unabridged Dictionary (1913), there are

- *Anything proposed for imitation; an archetype; an exemplar; that which is to be, or is worthy to be, copied or imitated; as, a pattern of a machine.*
- *A part showing the figure or quality of the whole; a specimen; a sample; an example; an instance.*
- *Figure or style of decoration; design; as, wall paper of a beautiful pattern.*

- *Something made after a model; a copy.*
- *Anything cut or formed to serve as a guide to cutting or forming objects; as, a dressmaker's pattern.*

Whereas more recently, the Cambridge Advanced Learner's Dictionary defines *pattern* as *something which is used as an example, especially to copy*, as well as *a recognizable way in which something is done, organized, or happens*. These definitions cover both individual entities (e.g. an apple, an alphabetic character, etc.) and descriptive concepts (e.g. how an apple looks like, how to spell the name "John", etc.).

Intuitively, *Pattern analysis* refers to the study of observing, discovering, organizing, discerning, perceiving and visualizing patterns of interests from the problem domain as well as making sound and reasonable decisions about the patterns. The analysis of patterns can be either spatial (e.g. What is the density of the elk in Asia?), temporal (e.g. When the population of the ibex in Tibet reached its peak?) as well as both spatial and temporal (e.g. What was the impact of the greenhouse effect to the world-wide geographical distribution of the wild swans in the past century?). Sharing the common points of a variety of scientific, social and economical researchers, the Nobel prize winner Herbert A. Simon emphasized the importance of "a larger vocabulary of recognizable patterns" in the experts' empirical researches for decision making and problem solving [Sim86].

1.2 Pattern Analysis in the Computer Science Domain

The advancement of computer science, which enables faster processing of huge data, has facilitated the use of elaborate and diverse methods in highly computationally demanding systems. At the same time, demands on automatic pattern analysis systems are rising enormously due to the availability of large databases and stringent performance requirements (speed, accuracy and cost) [JDM00]. In the past fifty years, numerous algorithms have been invented to handle certain types of pattern analysis tasks. Many computer programs have been developed to exhibit effective pattern analyzing capability. Significant commercial software has begun to emerge.

Watanabe [Wat85] refers a pattern in the computer science domain as

> ***Definition:*** A *pattern* is an opposite of a chaos; an entity, vaguely defined, that could be given a name.

In practice, instances of a pattern can be any representations of entities that can be processed and recognized by a computer, such as a fingerprint image, a text document, a gene expression array, a speech signal, as well as their derivatives, such as a biometrical identification, a semantic topic, and a gene functional specification, etc.

In the literature, pattern analysis is frequently mentioned together with *pattern recognition*, but the scope of pattern analysis greatly extends the limitation of the latter. As a comparison, the online Pattern Recognition Files [Dui04] refer the sub-disciplines of pattern recognition as follows:

> Discriminant analysis, feature extraction, error estimation, cluster analysis (together sometimes called statistical pattern recognition), grammatical inference and parsing (sometimes called syntactical pattern recognition).

whereas the journal *Pattern Analysis and Machine Intelligence* gives examples on the scope of pattern analysis studies as follows:

> Statistical and structural pattern recognition; image analysis; computational models of vision; computer vision systems; enhancement, restoration, segmentation, feature extraction, shape and texture analysis; applications of pattern analysis in medicine, industry, government, and the arts and sciences; artificial intelligence, knowledge representation, logical and probabilistic inference, learning, speech recognition, character and text recognition, syntactic and semantic processing, understanding natural language, expert systems, and specialized architectures for such processing.

The interests in the pattern analysis study keep renewing. The application domains of pattern analysis in the computer science literature include, but not limited to, computer vision and image processing, speech analysis, robotics, multimedia, document analysis, character recognition, knowledge engineering for pattern recognition, fractal analysis and intelligent control. Table 1.1 provides some examples of pattern analysis applications in various problem domains.

Table 1.1: Examples of pattern analysis applications.

Problem Domain	Application	Input Instances	Patterns Being Analyzed
Image document analysis	Optical character recognition	Scanned documents in image format	Characters and words
Bioinformatics	Sequence matching	DNA sequences	Known genes/patterns
Text document analysis	Associate the online news with predefined topics	Online news	Semantic categories/topics
Data mining	Investigating the purchasing habits of super market customers	Super market transactions	Well separated and homogeneous clusters / extracted rules
Speech recognition	Commanding the computer using human voice	Voice waveform	Voice commands
Temporal analysis	Predicting the trend of stock market	Stock quote data	The hidden function that the change of the stock price follows

1.3 Machine Learning for Pattern Analysis

The best pattern analyzer in the human's civilization, besides the almighty God, most likely is the human himself. In the age of two, a baby is able to name nearly all the toys and dolls scattered on the floor and pick up his/her favorite Barney. Recognizing more abstract entities like numbers and alphabets is not a difficult task for a six-year-old child. Gaining such recognition capability certainly involves a complicated and continuous learning process (as an example given by Figure 1.1). Yet ironically, we don't understand exactly how we analyze patterns and how we learn to do so.

Having the above limitation, generations of scientists ever since the creation of the world's first so-called *intelligent machine* which could be traced back to the syllogistic logic system invented by Aristotle in the 5th century B.C [Buc02], are far from being capable of reproducing a machine that thinks or acts exactly like a human. Fortunately, a machine is not necessarily to think and act exactly like a human before it can serve us quite well. As a matter of fact, given a human's natural solution to a task, finding alternative and simplified solutions that suite better to the machine's repetitive nature reflects the art of numerous inventive works. A good example in the industry is the washing machine, which substitute the human's complicated washing activity with repeated spins. Understanding this, rather than attempting to exactly replicate the human's thoughts during pattern analysis, it is more practical to study in favor of the nature of a machine.

Designing a pattern analysis machine/system essentially involves the following three aspects [JDM00]:

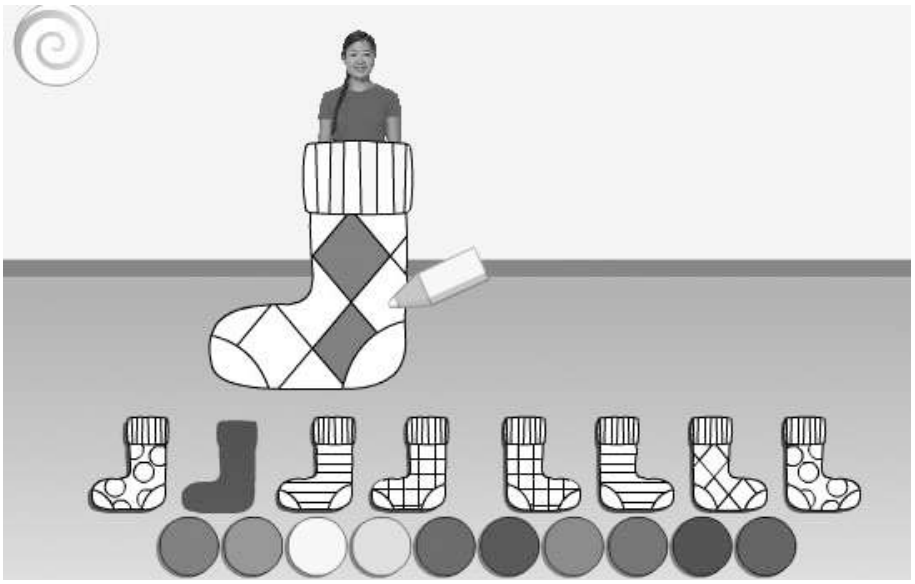


Figure 1.1: A coloring game for children on the PDSKids web site http://pbskids.org/ boohbah/socks.html. Completing this game requires pattern analysis knowledge in various aspects like close area identification, pen position tracking (both being confirmative analysis) as well as optimal color combination (being exploratory analysis), etc. Gaining these knowledge involves a complicated and continuous learning process.

1. Data acquisition and preprocessing,

2. Data representation, and

3. Decision making.

Through the first two steps, we are able to abstract the patterns from the problem domain and represent them in a normalized, machine understandable format for the further use of more general algorithms during decision making. The patterns are usually represented as vectors of measurements or points in a multidimensional space. With respect to the decision making process, it has been shown that algorithms based on *machine learning* outperform all other approaches that have been attempted to date [Mit97].

With reference to human's learning activities, we may say that a machine "learns" whenever it changes its structure, program or data (based on its inputs or in response to external information) such that its expected future performance improves [Nil96]. Tom M. Mitchell [Mit97] formalized this definition as

> ***Definition:*** A computer program is said to *learn* from experience E with respect to some class or tasks T and performance measure P, if its performance at tasks in T as measured by P improves with experience E.

Various learning problems for pattern analysis can be formalized in this fashion. Two examples from Table 1.1 are illustrated as follows:

An optical character recognition learning problem:

- Task T: Recognizing optical characters.
- Performance measure P: Percentage of characters correctly recognized by the computer.
- Experience E: A set of optical characters with corresponding alphanumeric characters that are correctly recognized by the human.

A data mining learning problem:

- Task T: Finding super market customers that have common purchasing habits.
- Performance measure P: The similarity among the customers being identified in the same group and the dissimilarity among the customers being identified in different groups.
- Experience E: A set of super market transactions.

While a machine is not necessarily to, and is far from being able to learn in the same way as what a human does, with no doubt, the study of machine learning algorithms is motivated by the theoretical understanding of human learning, albeit partial and preliminary. As a matter of fact, there are various similarities between machine learning and human learning. In turn, the study of machine learning algorithms might lead to a better understanding of human learning capabilities and limitations as well.

1.4 Supervised and Unsupervised Learning, Classification and Clustering

Depending on the nature of the data and the availability of appropriate models for the training source, the analysis of a pattern may be either confirmatory or exploratory (Figure 1.1).

A typical confirmatory pattern analysis task is the so-called *classification* problem. In a classification task, the input pattern is identified as a member of a *class*, where the class is predefined by the system designer. The classification task usually involves a *supervised* machine learning process, where the class labels of the training instances are given. The optical character recognition problem Section 1.3 is a typical supervised learning task.

On the other hand, one of the typical exploratory tasks is the *clustering* problem. In a clustering task, the input pattern is assigned to a class, which is automatically generated by the system based on the similarity among patterns. The clustering task usually involves an *unsupervised* machine learning process, in which the classes are hitherto unknown when the training instances are given. The data mining problem in Section 1.3 is a typical unsupervised learning task.

Readers shall note that the term *classification* (*categorization* in some cases) may refer to a broader scope in the literature. For example, Watanabe [Wat85] posed pattern recognition as a *classification* task, whereas the two different types of learning refer to the so-called *supervised classification* and *unsupervised classification* tasks. Similar terminology also appeared in [HKLK97, Rau99] etc.

While supervised and unsupervised learning are based on different models of

the training source. Studies have shown that they share a wide range of theoretical principles. Most significantly, the key element in both supervised and unsupervised learning is grouping, which in turn greatly involves the measurement of the similarity between two patterns. Given an unsupervised learning method proposed in the literature, one is most likely capable of finding its sibling in the supervised learning family; and vice versa.

1.5 Contributions of The Thesis

While the studies and applications of machine learning algorithms have been emerging in the past decades, due to the limited understanding of the human's learning behavior, the design of a general purpose machine pattern analyzer remains an elusive goal. In the meantime, the human's domain knowledge still plays an important role in designing a pattern analyzer and applying it in a specific problem.

This thesis mainly deals with unsupervised learning algorithms for cluster analysis. The application of the research is targeted for text mining and biological information mining. Data in these two domains are featured with high-dimension, large scale and high noisiness. More specifically, this thesis mainly attempts to answer the following two representative questions in cluster analysis:

- How to improve the efficiency of cluster analysis on high dimensional, large scale data with minimal requirements on the user's prior knowledge on the data distribution and system parameter setting, without losing clustering quality, compared with various slow learning, quality-optimized algorithms?

- How to evaluate the clustering results in a fairly quantitative manner, so that a clustering system can be fine-tuned to produce optimal results?

One of the major contributions of this thesis is the proposed novel artificial neural network architecture of Adaptive Resonance Theory under Constraint (ART-C) for cluster analysis. ART-C is an ART-based architecture [CG87b] capable of satisfying user-defined constraints on its category representation. This thesis will show that ART-C is more scalable than the conventional ART neural network on large data collections and is capable of accepting incremental inputs on the fly without re-scanning the data in the input history.

The capacity and the efficiency of the ART-C neural network will be examined through several case studies in the text and bioinformatics domains. The characteristics and the challenges of the studies in these two problem domains are thoroughly studied. For the benchmark purpose, two sets of clustering evaluation measures, namely evaluation measures based on cluster distribution and evaluation measures based on class conformation, are proposed and extensively studied. Experiments show the strength of these evaluation measures in various tasks including discovering the inherent data distribution for suggesting the optimal number of clusters, choosing a suitable pattern proximity measure for a problem domain and comparing various clustering methods for a better understanding of their learning characteristics. Experiments also suggest a number of advantages of these evaluation measures over existing conventional evaluation measures.

1.6 Outline of The Thesis

The rest of this thesis is organized as follows.

Chapter 2 reviews the unsupervised learning algorithms for cluster analysis in the literature through a comprehensive typology study.

Chapter 3 reviews existing neural network architectures and learning rules for a better understanding of the thesis. This chapter also briefly review two families of competitive learning neural networks, namely SOM and ART.

Chapter 4 proposes a novel neural network, Adaptive Resonance Theory under Constraint (ART-C), whose architecture and learning algorithm are described. Two variations of ART-C which correspond to the existing variations of ART are studied in more details.

Chapter 5 provides a literature review on the evaluation methodologies for clustering analysis, studies the difficulties of accessing the efficacy of a clustering system and proposes a set of evaluation measures for the clustering methods studied in this thesis.

Chapter 6 reports the application of clustering algorithms for pattern analysis in gene expression analysis and text mining domains. The characteristics of these two problem domains are studied. The performance of the clustering algorithms being studied in the thesis are accessed through statistical comparison work on a number of real-life problems.

The last chapter, Chapter 7 summarizes the thesis contents and proposes future work.

CHAPTER 2
CLUSTER ANALYSIS: A REVIEW

2.1 Problem Definition

As one of the major research domains of pattern analysis, cluster analysis is the organization of a collection of patterns into clusters based on similarity. Intuitively, patterns within a meaningful cluster are more similar to each other than they are to patterns belonging to a different cluster, in terms of the quantitative similarity measure adopted by the system. Clustering may be found under different names in different contexts, such as numerical taxonomy (in biology and ecology), partition (in graph theory) and typology (in social sciences) [TK99].

Cluster analysis is a useful approach in data mining processes for identifying hidden patterns and revealing underlying knowledge from large data collections.

The application areas of clustering, to name a few, include image segmentation, information retrieval, document classification, associate rule mining, web usage tracking and transaction analysis [HTTS03]. Some representative application directions of cluster analysis are summarized below [TK99]:

- **Data Reduction.** Cluster analysis can contribute to compression of information included in data. In several cases, the amount of available data is very large and its processing becomes very demanding. Clustering can be used to partition data set into a number of "interesting" groups. Then, instead of processing the data set as an entity, the process can obtain the representatives of the generated clusters for effective data compression.

- **Hypothesis Generation and Hypothesis Testing.** Cluster analysis can be used to infer some hypotheses concerning the data. For instance, a clustering system may find several significant groups of customers in a supermarket transaction database, based on their races and shopping behaviors. Then the system may infer some hypotheses for the data, such as "*Chinese customers like pork more than beef*" and "*Indian customers buy curry frequently*". One may further apply cluster analysis to another representative supermarket transaction database and verify whether the hypotheses are supported by the analysis results.

- **Prediction Based on Groups.** Cluster analysis is applied here to the data set and the resulting clusters are characterized by the features of patterns that belong to these clusters. Then unknown patterns can be classified into specified clusters based on their similarity to the clusters' features. For example, cluster analysis can be applied to a group of patients infected by the

same disease. Useful knowledge concerning "*what treatment combination is better for patients in a specific age and gender group*" can be extracted from the data. Such knowledge can further assist the doctor to find the optimal treatment for a new patient with considerations on his/her age and gender.

Unlike the other major category of pattern analysis research domain, i.e. classification or the so-called *discriminant analysis* in a more general form, which usually involves supervised learning, cluster analysis typically works in an unsupervised manner. To formalize the comparison between these two categories of analysis tasks, we model the problem domain as a mixture probability $M(K, W, C, Y)$, where the data points are approximated with K sub-groupings (patterns) $C_i, i = 1, \cdots, K$ given by

$$P(X) = \sum_{i=1}^{K} \mathbf{w}_i \cdot P(X|C_i, Y_i(X)), \tag{2.1}$$

where X is the input in the problem domain, $\mathbf{w}_i$ is the *mixture weight* and $Y_i(X) \equiv X \rightarrow C_i$ is a mapping from the input X to the sub-grouping C_i. Essentially, both classification and clustering involve the estimation of the model's parameters. In a classification task,

- K is pre-defined and fixed.
- Instances of X (marked as $\mathbf{x}$) are given with corresponding mapping labels $\mathbf{y}(\mathbf{x})$.
- Learning of the system involves estimating W and the distribution of C.
- The objective of the learning is to minimize the mismatch in predicting $\mathbf{y}(\mathbf{x})$ for a given $\mathbf{x}$.

On the other hand, in a clustering task,

- All the parameters of the model, namely K, W, C, and Y, are not known.
- The objectives of the learning are to:
  1. Minimize the summed intra-grouping variance (error) of C, and
  2. Maximize the summed inter-grouping distance of C.

A common formalization of the above two objectives, but not limited to, is

$$\begin{aligned} &\text{minimizing} \quad E = \textstyle\sum_{i=1}^{K} \int y_i(\mathbf{x}) e(\mathbf{x}, \mathbf{c}_i) p(\mathbf{x}) d\mathbf{x}, \text{ and/or} \\ &\text{maximizing} \quad D = \textstyle\sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} e(\mathbf{c}_i, \mathbf{c}_j) \end{aligned} \tag{2.2}$$

respectively, where c_i is the descriptive pattern (e.g. cluster centroid) of the sub-grouping C_i , $y_i(\mathbf{x})$ is the cluster membership assignment value on C_i, $e(\mathbf{x}, \mathbf{c}_i)$ is the variance between $\mathbf{x}$ and $\mathbf{c}_i$. Since a cluster analyzing system commonly deals with a finite set of training instances, the first objective of Equation 2.2 is practically implemented through

$$\text{minimizing } E = \sum_{i=1}^{K} \sum_{j=1}^{N} y_i(\mathbf{x}_j) e(\mathbf{x}_j, \mathbf{c}_i), \tag{2.3}$$

where N is the number of training instances.

In a clustering task, there is relatively fewer prior information (e.g. statistical models) available about the data if compared with the classification task. The decision-maker must take as few assumptions about the data as possible. In this point of view, clustering always remains a challenging task as the output quality of a clustering algorithm may vary depending on various factors, such as the features of the data set and parameter values of the algorithm [HBV01], which are unlikely available in advance.

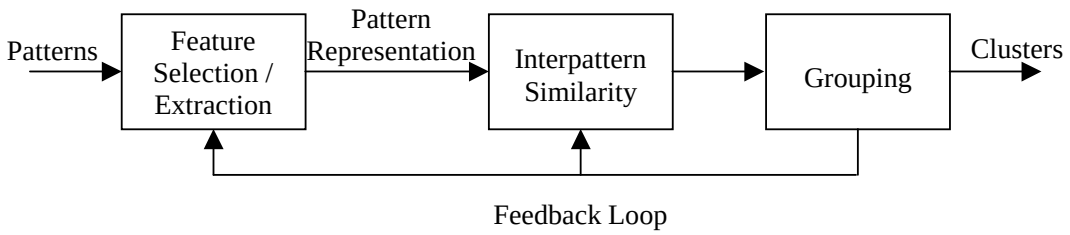


Figure 2.1: A typical sequencing of clustering activity.

2.2 The Prerequisites of Cluster Analysis

A.K. Jain et al. [JD88, JMF99] summarized several steps that a clustering activity typically involves. They are listed as follows:

1. Pattern representation, optionally including feature extraction and/or selection.
2. Definition of a pattern proximity measure for the data domain.
3. Clustering or grouping of data points according to the chosen pattern representation and the proximity measure.
4. Data abstraction (if needed).
5. Assessment of output (if needed).

The first three steps are depicted in Figure 2.1, which includes a feedback path where the grouping process output could affect subsequent feature extraction and similarity computations.

We consider the first two steps as the pre-processing of cluster analysis. The background knowledge on these two steps are briefly reviewed in the following sub-sections.

2.2.1 Pattern Representation, Feature Selection and Feature Extraction

Pattern representation refers to the paradigm for observation and the abstraction of the learning problem, including the type, the number and the scale of the features, the number of the patterns and the format of the feature representation. Feature selection, as a pre-processing for pattern representation, is defined as the task of identifying a set of most representative subset of the natural features (or transformations of the natural features) to be used by the machine. As another important step of feature representation, feature extraction refers to the paradigm for converting the observations of the natural features into a machine understandable format.

Pattern representation is considered as the basis of machine learning. For the ease of machine processing, the patterns are usually represented as vectors of measurements or points in a multidimensional space. It is very common that such an abstraction will incur discrepancy between the human's observation and the machine's input. In turn, as human accessibility of the patterns is highly dependent on their representation format, an unsuitable pattern representation may result in a failure to produce meaningful clusters as the user desires.

Given the synthetic data set in Figure 2.2a, using a cartesian coordinate representation, a clustering method would have no problem in identifying the five compact groups of data points (Figure 2.3a). However, when the same representation is applied to the data set in Figure 2.2b, the four string-shape clusters probably would not be discovered as they are not easily separable in terms of Euclidean

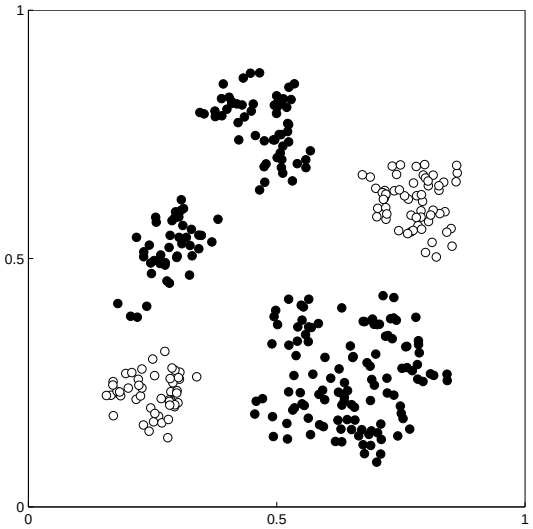
distance (Figure 2.4a). Instead, a polar coordinate representation could lead to better result as the data points in each string-shape cluster are close to each other in terms of polar angle (Figure 2.4b).

Feature selection and extraction play an important role for abstracting complex patterns into a machine understandable representation. The selected feature set used by a clustering system regularizes the area that the system gives attention to. Referring to the data set in Figure 2.2a, if a *coordinate position* is selected as the feature set, many clustering algorithms would be capable of identifying the five compact clusters (Figure 2.3b). However, if only the *color* of the data points is selected as the feature, a clustering system would probably output only two clusters, containing *white points* and *black points* respectively (Figure 2.3b).

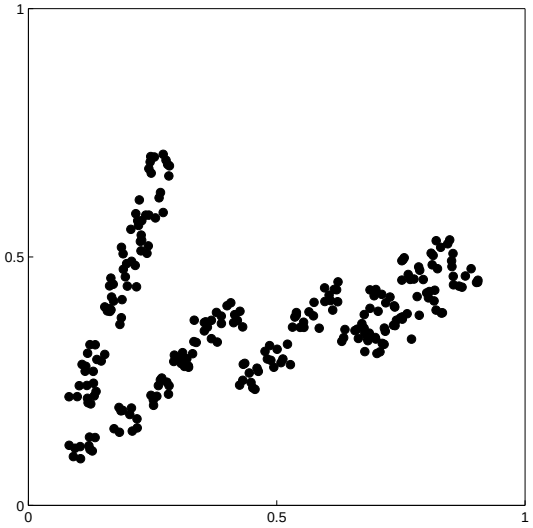
The feature set also affects the quality as well as the efficiency of a clustering system. A large feature set containing numerous irrelevant features does not improve clustering quality but incurs a higher computational cost to the system. On the other hand, an insufficient feature set may decrease the accuracy of the representation and therefore cause potential loss of important patterns in the clustering output.

2.2.2 Pattern Proximity Measure

Pattern proximity refers to the metric that evaluates the similarity (or in contrast, the dissimilarity) between two patterns. While a number of clustering methods (such as [RS98]) disclaim the use of specific "distance" (dissimilarity) measures, they use alternative pattern proximity measures to evaluate the so-called *relation-*

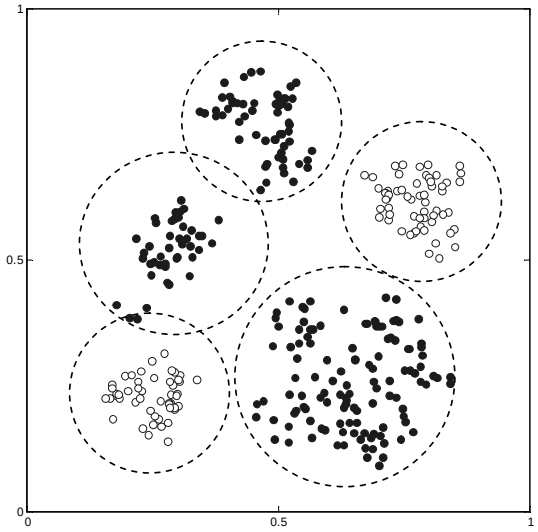


(a)

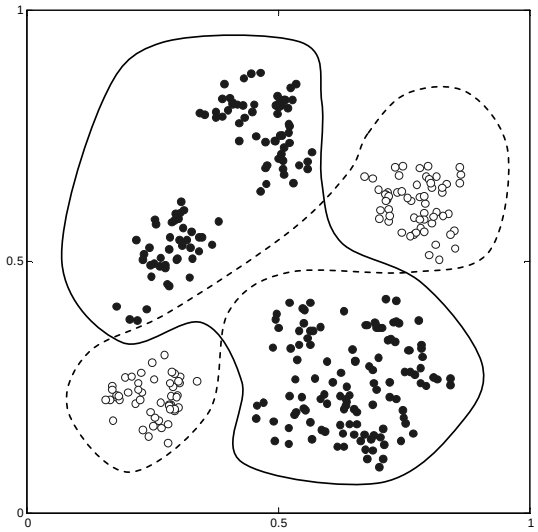


(b)

Figure 2.2: To identify compact clusters in terms of distance, a cartesian coordinate representation is more suitable for case (a), while a polar coordinate representation is more suitable for case (b), with reference to the "natural" groupings in Figure 2.3a and Figure 2.5 respectively.

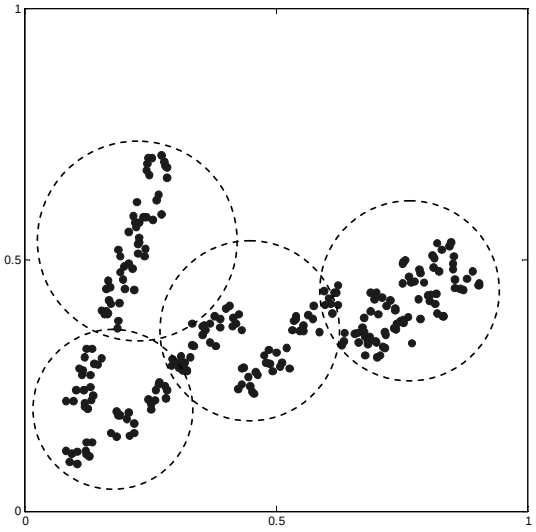


(a)

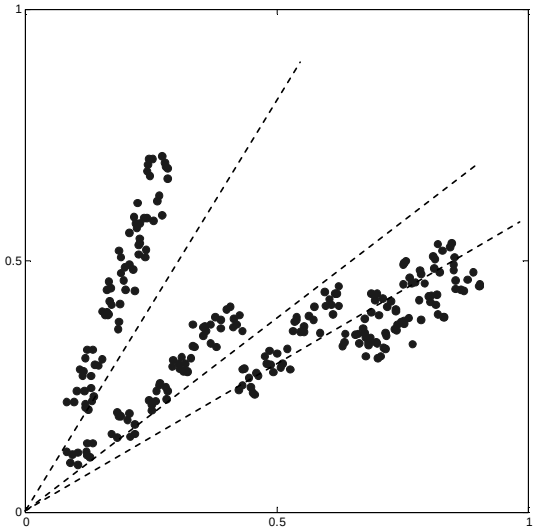


(b)

Figure 2.3: Two different, while sound clustering results on the data set in Figure 2.2a, using coordinate position (a) and color (b) as the feature respectively. Clusters in (a) are separated with dashed lines (not necessarily the decision boundaries of the machine). The two clusters in (b) are identified with dashed line and solid line respectively.



(a)



(b)

Figure 2.4: Two different clustering results on the data set in Figure 2.2b, using cartesian coordinate (a) and polar coordinate (b) for pattern representation respectively. Clusters are separated with dashed lines (not necessarily the decision boundaries of the machine). Result (b) is closer to the "natural" grouping of the data set in the user's view as illustrated in Figure 2.5.

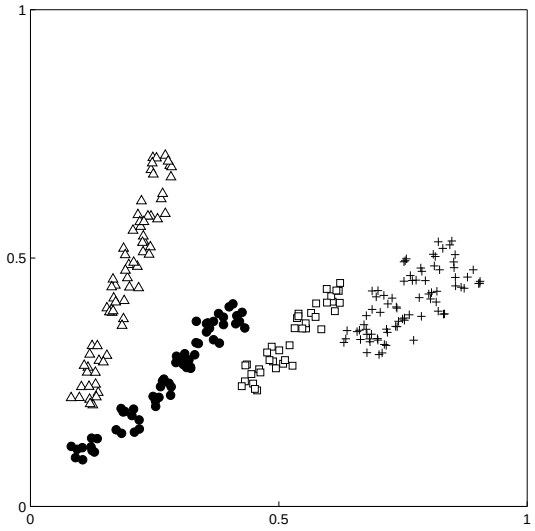
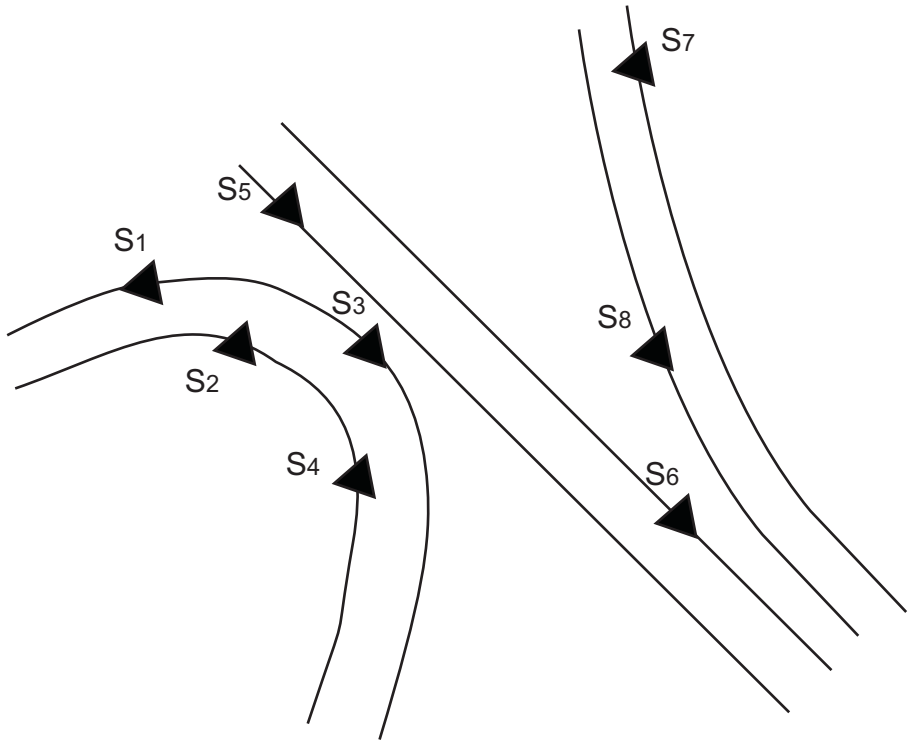


Figure 2.5: The "natural" grouping of the data in Figure 2.2b in a user's view. Data points in the same cluster are identified with the same marker.

ship between two patterns. A pattern proximity measure serves as the basis for cluster generation as it indicates how two patterns "look alike".

Since the type, the range and the format of the input features are defined during the pattern representation stage, a pattern proximity measure should correspond to the pattern representation. In addition, a good proximity measure should be capable of utilizing only the key features of the data domain. Referring to Figure 2.2 again, with a cartesian representation, Euclidean distance is suitable for identifying the geometric differences among the five clusters in data set (a) but may not be capable enough of recognizing the clusters in data set (b). Instead, cosine distance is more suitable for data set (b) as it gives no weight to a vector's radius and focuses on the differences of the vectors' projections on the polar angle only. Generally, a careful review on the existing correlations among patterns helps to determine a suitable pattern similarity measure.



Criteria	Clustering Result	Interpretation
Geometric distance	$C_1 = \{S_1, S_2, S_3, S_4, S_5\}$ $C_2 = \{S_6, S_8\}$ $C_3 = \{S_7\}$	Cameras in each cluster are geometrically closer to each other than to those in other clusters.
Connectivity	$C_1 = \{S_1, S_2, S_3, S_4\}$ $C_2 = \{S_5, S_6\}$ $C_3 = \{S_7, S_8\}$	Each cluster contains the cameras in the same road.
Density	$C_1 = \{S_1, S_2, S_3, S_4, S_5\}$ $C_2 = \{S_6, S_7, S_8\}$	C_1 identifies the zone intensively equipped with cameras, in contrast to the rest of the area.

Figure 2.6: Using various pattern proximity measures, the eight speed cameras on the three roads may be clustered into different cluster groupings, each solution with an acceptable interpretation.

Given an existing pattern representation paradigm, a data set may be separable in various ways. The use of different pattern proximity measures may result in very different clustering outputs. Figure 2.6 depicts an example that includes eight speed cameras on three roads as the system's input. Based on different criteria that measure the proximity between two cameras, there are different clustering solutions, each with an acceptable interpretation. In most cases, the clustering system is desired to output only one (or a few number of) optimal grouping solution that best matches the user intention on the data set, although that may be partial and subjective. Hence it is important to identify the pattern proximity measure that effectively and precisely formulates the user's intention on the patterns.

2.3 Clustering Algorithms: A Typology Review

A clustering algorithm groups the input data according to a set of predefined criteria. The clustering algorithm used by a system can be either statistical or heuristic. In essence, the objective of clustering is to maximize the intra-cluster similarity and minimize the inter-cluster similarity [ZFLW02]. The similarity measure is chosen subjectively based on the system's ability to create "interesting" clusters, as reviewed in Section 2.2.2. A large variety of clustering algorithms have been extensively studied in the literature. While a comprehensive survey of clustering algorithms is not the focus of this chapter, we give a bird's eye review of various types of available algorithms, with reference to some existing reviews and surveys [JMF99, SCZ00, SKK00, ZFLW02].

Clustering algorithms can be classified according to:

- the type of data input to the algorithm,
- the clustering criterion defining the similarity between data points and
- the theory and fundamental concepts on which clustering analysis techniques are based (e.g. fuzzy theory, statistics).

Readers should note that the pre-processing stages of a clustering activity (i.e. the pattern representation and pattern approximation stages) are considered as factors to classify the clustering algorithms. This is due to the inherently close interactions among every stage of a clustering activity. Given a machine learning paradigm, it usually can cope with a (or a few number of) specific pattern representation and is capable of grouping the input data based on a specific predefined pattern proximity measure.

There are a large number of categorizations of clustering algorithms, Table 2.1 names a few of them, based on different criteria related to the fundamental concepts.

For a better understanding of this thesis, the last typology study, i.e. the categorization based on system architecture, deserves an introduction with more details. A number of state-of-the-art algorithms with different system architectures are reviewed in the following sections.

2.3.1 Partitioning Algorithms

A partitioning algorithm obtains a single partition of the data points with a set of so-called *decision boundaries* [Bol98]. The studies of partitioning algorithms are

Table 2.1: Various types of clustering methods, based on learning paradigm, codebook size, cluster assignment and system architecture respectively.

Criteria	Categories
Learning paradigm	*Off-line:* Iteratively batch learning on the whole input set. *On-line:* Incremental learning that does not remember the specific input history.
Codebook size (number of output clusters)	*Static-sizing:* The codebook size is fixed. *Dynamic-sizing:* The codebook size is adaptive to the distribution of input data.
Cluster assignment	*Hard:* Each input is assigned with one class label. *Fuzzy:* Each input is given a degree of membership with every output cluster.
System architecture	*Partitioning:* The input space is naively separated into disjoint output clusters. *Hierarchical:* The output tree shows the relations among clusters. *Density-based:* The input data are grouped based on density conditions. *Grid-based:* The spacial input space is quantized into finite sub-spaces (grids) before clustering of each sub-space.

closely related to the *Vector Quantization* studies [FKKN96, FKN98, HH93]. The minor difference between these two tasks may be that, a partitioning algorithm involves the identification of the eventual partition inside the multidimensional data set to be analyzed, whereas a vector quantization method focuses more on representing the data by a reduced number of elements that approximate the original data set as closely as possible [Fle97]. Despite the difference outlined here, many researchers consider these studies practically equivalent.

The partitioning algorithms usually produce clusters by optimizing a criterion function defined either locally (on a subset of the patterns) or globally (defined over all of the patterns) [JMF99, SKK00]. It however has been shown that obtaining the global optimization a predefined criterion function is usually NP-Hard [GJW80]. Normally a partitioning algorithm involves an iterative search for the local-minimal or local-maximal solution and stops when such local optimization is reached.

In this category, K-Means [TG74, SKK00] probably is the most commonly used algorithm. The criterion used by the K-Means is the *Summed Square Error*, which intuitively reflects the distance of each point from the center of the cluster to which the point belongs. The summed square error can be formalized as

$$E = \sum_{i=1}^{K} \sum_{j=1}^{N} y_i(\mathbf{x}_j) ||\mathbf{x}_j - \mathbf{c}_i||^2, \tag{2.4}$$

where K is the number of clusters (partitions), N is the number of training instances, $y_i(\mathbf{x}_j)$ is a "hard" cluster assignment of the pattern $\mathbf{x}_j$ to the cluster C_i, defined as

$$y_i(\mathbf{x}_j) = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ is assigned to } C_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

$\mathbf{c}_i$ is the representative vector of cluster C_i, usually its centroid given by

$$\mathbf{c}_i = \frac{\sum_{j=1}^{N} y_i(\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{N} y_i(\mathbf{x}_j)} \tag{2.6}$$

and $||.||$ is the Euclidean distance function, defined as

$$||\mathbf{x}_j - \mathbf{c}_i|| = \sqrt{\sum_{k=1}^{M}(x_{jk} - c_{ik})^2} \tag{2.7}$$

where M is the dimension of the vectors $\mathbf{x}_j$ and $\mathbf{c}_i$.

Depending on the method used for re-computing the cluster centers, there are Batch K-Means and Online (incremental) K-Means. Typically both Batch K-Means and Online K-Means start with a set of K arbitrarily selected training instances (named *seeds* in some studies) as the representatives of the clusters (named *cluster prototypes*). Each learning iteration of Batch K-Means first assigns all training instances of the data set to the correspondingly winner clusters, each defined as the prototype that is nearest to the training instance. Then the cluster prototypes are updated with the mean of the training instances associated with the cluster. On the other hand, Online K-Means updates the winner cluster's prototype incrementally, i.e., right after each training instance is given. The identification of the so-called winner cluster is based only on the points processed so far, without considering the whole cluster or the whole database. In addition, Online K-Means does not calculate the actual centroid of each cluster, but estimate the prototype of the winner cluster though a learning function. The learning function normally adjusts the cluster prototype slightly closer to the presented training instance, such that the error between the training instance and the new cluster prototype is gradually decreased. A commonly used learning function is

defined as

$$\begin{aligned} \mathbf{c}^{t+1} &= \alpha \cdot \mathbf{x}_j + (1-\alpha) \cdot \mathbf{c}^t \\ &= \mathbf{c}^t + \alpha \cdot (\mathbf{x}_j - \mathbf{c}^t), \end{aligned} \tag{2.8}$$

where $\alpha \in [0, 1]$ is the learning rate, $\mathbf{c}^t$ and the $\mathbf{c}^{t+1}$ are the prototype of cluster C_i before and after learning.

A problem raised in the above incremental learning process is the concern on the convergence of the algorithm. Grossberg [Gro82] pointed out that such a learning activity may result in oscillation if the input data are too densely distributed. Bottou and Bengio [BB95] however proved its convergence, at least with a decreasing learning rate. In addition, the output of both batch K-Means and online K-Means are sensitive to the seed cluster prototypes. That is, the output solution is typically locally optimal and is deterministic by the initialization of the algorithm.

To tackle the above deficiency, a large number of variations and hybrid models over K-Means have been studied in the literature. Some studies seek for good initial prototypes so that the output solution tends to reach sub-optimal, i.e. optimal in a few number of local circumstances [ADR94, And73, BF98, KKZ94, LVV01]; some adjust the partition through merging and/or splitting existing partitions [BH65, VR92]; whereas the others combine extra criterion function during searching [DS73, MJ92, Sym81].

EM (Expectation Maximization) [DLR77] can be considered as the generalized model of K-Means. Unlike K-Means which assigns the input to one cluster to maximize the variance in means, EM computes probabilities of cluster memberships based on one or more probability distributions. The goal of the clustering

algorithm is to maximize the overall probability or likelihood of the data, given the (final) clusters. The links between the inputs and the output clusters are fuzzy, yielding partially overlapped partitions in the output space. It is then not difficult to apply a binary threshold in order to obtain a *hard* partitioning. Further more, the well-known MRF (Markov Random Fields) modeling [JS96] combines this probability model with Hidden Markov theory and has been widely applied to image analysis applications.

Another algorithm of this category is PAM (Partitioning Around Medoids) [KR87]. PAM determines the most centrally located instance within each cluster (entitled *medoid*) to representative that cluster. Similar to K-Means, the algorithm begins by selecting an object as the medoid of each cluster. Then, each of the non-selected objects is grouped with the medoid to which it is the most similar. Different from K-Means learning, PAM swaps medoids with other non-selected objects until all objects qualify as medoid. It is clear that PAM is a computational expensive algorithm, as it compares an object with the entire data set [NH94] to calculate the medoids. To tackle this deficiency, CLARA (Clustering Large Applications) [KR90] implements PAM in a subset of the inputs. It draws multiple samples of the data set, applies PAM on samples, and then outputs the best clustering out of these samples [NH94]. CLARANS (Clustering Large Applications based on Randomized Search) [NH02] combines the random sampling techniques with PAM for handling large scale spatial data. The clustering process can be presented as searching a graph where every node may belongs to the set of K medoids. CLARANS selects a node and compares it to a user-defined number of their neighbors searching for a local minimum. If a better neighbor is found (i.e., having a lower square error), CLARANS moves to the neighbor's node and the

process starts again; otherwise the current clustering is a local optimum. If the local optimum is found, CLARANS starts with a new randomly selected node in search for a new local optimum. This algorithm has been shown to have a higher efficiency over PAM.

2.3.2 Hierarchical Algorithms

Hierarchical algorithms creates a hierarchical decomposition of the inputs [SKK00]. Unlike partitioning algorithms that output a flat partition represented with the boundaries among the output clusters as well as the cluster centers (if needed), hierarchical algorithms output a so-called *dendrogram* [Hor88] that iteratively splits the input set into smaller subsets until each subset consists of only one object. In such a hierarchy, each level of the tree represents a clustering of the input. In addition, the nested relations between the clusters and similarity levels at which the groupings change are reflected in the hierarchy.

According to the methods that produce clusters, hierarchical clustering algorithms can further be divided into [TK99]:

- *Agglomerative algorithms.* They produce a sequence of clustering schemes of decreasing number of clusters at east step. Through each step of the clustering scheme, two closest clusters are merged.

- *Divisive algorithms.* These algorithms produce a sequence of clustering schemes of increasing number of clusters at each step. Contrary to the agglomerative algorithms, through each step, a selected cluster is split into two smaller clusters.

In the following, some representative hierarchical clustering algorithms are reviewed.

BIRCH [ZRL96] uses a hierarchical data structure called CF-tree for incrementally and dynamically clustering the incoming data points. CF-tree is a height-balanced tree that stores the clustering parameters. BIRCH can typically find a good clustering with a single scan of the data and improve the quality further with a few additional scans [HBV01]. It is also claimed as the first clustering algorithm to handle noise effectively [ZRL96]. However, the size of each node in the CF-tree is limited, and hence can hold a limited number of inputs only. The output of BIRCH does not always correspond to a natural cluster. Moreover, BIRCH is order-sensitive, which means it will generate different clusters for different orders of the same input data.

CURE [GRS98] represents each cluster by a certain number of points that are generated by selecting *well scattered* points and then shrinking them toward the cluster centroid by a specified fraction. Since each cluster is represented with more than one points, the algorithm is capable of identifying clusters with non-spherical shapes and wide variances in size. It uses a combination of random sampling and partition clustering to handle large databases. It is also claimed to be capable of handling noise effectively.

ROCK [GRS99] is a robust clustering algorithm for boolean and categorical data. Unlike a conventional agglomerative algorithm which merges two clusters based on their distance, ROCK introduces the *links* concept to measure the proximity between a pair of data points. Prior study showed that ROCK not only generates better quality clusters than transitional algorithms, but also exhibits

good scalability properties.

Generally, in a hierarchical clustering system, the number of output clusters is not necessarily predefined. The notable disadvantage of hierarchical algorithms lies in the difficulty in determining the termination condition, that is, at which point the merging or splitting process stops.

2.3.3 Density-based Algorithms

Based on the fact of clustering essentially being a *density estimation* problem [BF98], density based algorithms typically treat clusters as dense regions of objects in the data space that are separated by regions of low density.

A widely known algorithm of this category is DBSCAN (Density Based Spatial Clustering of Applications with Noise) [EKSX96]. In DBSCAN, for each point in a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN requires only one parameter and supports the user in determining an appropriate value for it. It is capable of handling noises and discovering clusters of arbitrary shape. Comparison study has show that DBSCAN is notably more efficient than CLARANS. Moreover, DBSCAN is used as the basis for an incremental clustering algorithm proposed in [EKS+98].

Hinneburg and Keim introduced the DENCLUE (DENsity based CLUstEring) algorithm [HK98] to cluster large multimedia databases. The basic idea of this approach is to model the overall point density analytically as the sum of influence functions of the data points. Then clusters can be identified by determining density-attractors, which are local maximum of the overall density function. Clus-

ters of arbitrary shape can be easily described by a simple equation based on overall density function. The main advantages of DENCLUE include its good clustering properties in data sets with large amounts of noise, its compact mathematical description of arbitrary shaped clusters, as well as its significantly higher efficiency over DBSCAN. However, DENCLUE is sensitive to its parameter settings. This prevents it from producing high quality results without human experiences.

One of the notable characteristics of density based algorithms is that, the output cluster may not include all input data points as data points in the low density regions are excluded. This ensures their scalability to large data base as well as their robustness over noises. This however prevents their application in problem domains where every input data are equally important.

2.3.4 Grid-based Algorithms

Recently a number of clustering algorithms have been presented for spatial data, known as grid-based algorithms. These algorithms quantize the space into a finite number of cells and then do all operations on the quantized space.

STING (STatistical INformation Grid-based method) [WYM97] divides the spatial area into rectangular cells using a hierarchical structure. It scans over the data set and computes the statistical parameters (such as mean, variance, minimum, maximum and type of distribution) of each numerical feature of the objects within cells. Then it generates a hierarchical structure of the grid cells so as to represent the clustering information at different levels. STING obtains the statistical parameters by scanning the data set once only. Hence it is of high efficiency

in handling spatial data. However, it has been pointed out that STING lowers the quality and accuracy of clusters, despite the fast processing time [SCZ00].

WaveCluster [SCZ00] is a novel grid-based clustering algorithm that incorporates a signal processing technique named wavelet transformation. It first summarizes the data by imposing a multidimensional grid structure onto the data space. Each grid cell summarizes the information of a group of points that map into the cell. The wavelet transformation then is applied over the original feature space. The multi-resolution property of wavelet transformation enables human inspection to identify the dense regions on the converted frequency domain. Arbitrary clusters can be inspected at different degrees of detail. In addition, prior knowledge about the exact number of clusters is not required in WaveCluster. The drawback of the algorithm is that it requires human inspection on the transformed domain. In addition, the wavelet transformation and the human's incapability in inspecting complex data limits the algorithms's scalability for handling high dimensional data.

The main characteristics of grid-based approaches is their high efficiency, as the processing time is dependent on the number of cells in each dimension in the quantized space [SCZ00], which is typically much less than the number of data objects. The difficulty raised in the application of these algorithms is how to determine the appropriate number of grids. In addition, the scalability of these algorithms to a high dimensional domain is yet to be further improved.

Despite the numerous clustering algorithms reviewed above, there is no single method that can cope with all clustering problems. The choice of the clustering algorithm for a specific task affects the clustering result in a fundamental way. In

addition, the learning activity of a large number of clustering algorithms is controlled and hence affected by a set of internal parameters. The optimal parameter set is usually decided through empirical experiments on the specific data set.

CHAPTER 3

ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

An *artificial neural network*, usually referred to as *neural network*, is a machine that models the way in which the biological brain performs a particular task or function of interests. Neural networks typically employ a massive interconnection of simple computing cells referred to as "neurons" or "processing units" [Hay99]. Aleksander and Morton [AM90] offered the following definition of a neural network viewed as an adaptive machine:

> A neural network is a massive parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use.

The infrastructure of the neural network commonly resembles the brain in two respects:

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The network is usually implemented by using electronic components or its simulated in software on a digital computer [Hay99], of which the latter is the focus of this thesis. When viewed in the software simulation aspect, a neural network features its capability of learning from examples or interactions, as opposed to conventional programming tasks.

3.2 Learning in Neural Networks

Following the definition in Section 1.3, a neural network's learning is closely related to its activity that improves its performance from the experience with respect to a task. The objective of a neural network's learning activity could be self-organizing information through correlations of the data, minimizing an error metrics or maximizing rewards in a trial-and-error system over time [PU99]. Haykin [Hay99]

summarizes the learning process in the context of neural networks as the following sequence of events:

1. The neural network is *stimulated* by an environment.
2. The neural networks *undergoes changes* in its free parameters as a result of this stimulation.
3. The neural network *responds in a new way* to the environment because of the changes that have occurred in its internal structure.

An important prerequisite of the neural network's learning is the appropriate modeling of the learning environment, only with which the corresponding knowledge generated by a neural network could be meaningful. Such a modeling is referred to as the *learning paradigm* in the literature. There are three major learning paradigms being studied in the neural network context, namely supervised learning, unsupervised learning and reinforcement learning. As already reviewed in Section 1.4, supervised learning refers to the scenarios that the training inputs are provided together with the desired output; whereas for unsupervised learning, there is no such desired output information given. Reinforcement learning is a learning environment with interaction, i.e. trial and error. The neural network first "tries" to make some predictions, then receives the scalar evaluation (reward) of this predictions and selectively *reinforces* its learning towards the maximal received rewards. Different from supervised learning, in a reinforcement learning paradigm, not all input-output pairs are given at the same time.

The changing of a neural network's free parameters includes the adjustment of the interconnection weights between the neurons and sometimes the network's

topology adoption, for instance, the number of layers, the number of neurons and the pattern of connections. These changes are guided by a set of well-defined rules, commonly referred to as a *learning algorithm*. Depending on the problem domain, there exist a number of basic learning rules in the literature, including memory-based learning, Hebbian learning, error-correction learning and competitive learning.

In memory-based learning, the neural network's past experiences are explicitly stored in a large memory of correctly classified input-output examples. Upon presentation of a testing instance (unknown data), the neural network responds by finding the *local neighborhood* of the testing instance in its memory and makes decision accordingly. Since the past experiences of the neural network are stored "as is", i.e. no new knowledge is generated, this category of learning rules are sometimes called "lazy learning". Memory-based learning essentially involves the studies in the following two aspects:

1. The criteria in locating the local neighborhood of the testing instance.

2. The decision rule applied on the local neighborhood.

The most representative memory-based algorithm in the literature probably is the k Nearest Neighbor (kNN) classifier [CH67, Das91], including its specialized case Nearest Neighbor (NN) classifier which corresponds to $k = 1$.

The Hebbian learning rule realizes the biological learning characteristics, which states that "when two interconnected neurons fire at the same time, and repeatedly, the synapse's strength is increased" [Heb49]. Hebb [Heb49] adopted this observation as the basis for pattern association. The Hebbian rule determines the change

in the weight connection between two neurons. If two neurons are both activated, then the weight of the connection between these two neurons are increased. It has been proved that if the set of input patterns used in training are mutually orthogonal, the association can be learned by a two-layer pattern associator using Hebbian learning. However, if the input patterns are not mutually orthogonal, interference may occur and the network may not be able to learn associations. The Error-correction, introduced as below, outcomes this limitation.

Error-correction learning is rooted in optimum filtering. During training, an input is presented to the neural network, which generates a set of values on the output units. The actual output is compared with the desired one. The mismatch between the actual output and the desired output then triggers the neural network's learning. The neuron interconnections are updated in the way that the mismatch (usually formalized as an error function) decreases. The Least Mean Square (LMS) rule, also called the Delta rule [WH60], is a widely applied learning rule in this category. LMS uses summed square error to measure the mismatch and best fits for neural architectures with no hidden-layer. The generalized LMS rule [RHW86], as an improvement over LMS, handles learning in multi-layer neural architectures well.

Like Hebbian learning, competitive learning is also biologically plausible [HKP91]. However, competitive learning distinguishes from the above three learning rules with the output neurons being activated in a competitive way. That is, at one time there is maximally only one output neuron being activated, whereas with in the other three learning rules, multiple output neurons may be activated simultaneously. Learning of the competitively activated neuron follows the way that the

local error decreases. With such feature, competitive learning is highly suitable for discovering statistically salient features, and hence is widely adopted by various clustering techniques.

The application domains of the learning algorithms are dominated by the modeling of learning environment, i.e. learning paradigms. Generally speaking, memory-based learning rules, albeit simple, may suit in both supervised and unsupervised learning paradigms; error-correction learning and Hebbian learning are normally applied for supervised and reinforcement learning paradigms; while competitive learning is widely applied in an unsupervised learning paradigms.

Since the focus of this thesis is on clustering (unsupervised learning), the review on the competitive learning process is extended in the following section. In addition, two families of competitive learning neural networks are reviewed.

3.3 The Competitive Learning Process

The competitive learning rule is closely related to the family of neural architecture named *competitive neural architecture* (including *self-organizing maps*). This architecture adopts self-excitatory and inter-inhibitory connections inside of the computational layer (Figure 3.1), which enables the neural network to adaptively self-organize and build the topology preserving feature maps in respect to the input [Hay99, PU99].

Rumelhart and Zipser [RZ85] summarized the three major elements of competitive learning, namely

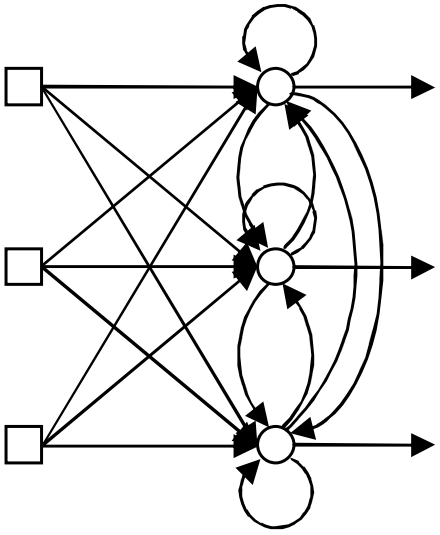


Figure 3.1: The competitive neural architecture.

1. A number of neurons initially with some randomly distributed interconnection weights, such that they have different responses to a given input pattern.

2. A limit imposed on the strength of the neurons.

3. A mechanism that permits the neurons to compete for the right to respond to a given input, such that at a time, only one output neuron (or only one neuron per group) is active.

Competitive learning algorithms may vary depending on the criteria used for the evaluation of the competition and the selection of the so-called "winner" of the competition. Generally, the competitive learning process falls into two steps, illustrated in Figure 3.2, and summarized as follows.

1. **Winner Search:** Given an input pattern $\mathbf{x}$ presented in the input layer, which causes a *local induction* $f(j)$ of each neuron j, where the inductive function $f(.)$ is predefined. The so-called winner of the competition is defined

as the neuron that receives either the minimal local induction or the maximal local induction as per definition. Given the instance of winner selection by maximization, the output signals y_j of the neurons are therefore set by

$$y_j = \begin{cases} 1 & \text{if } f(j) > f(k) \text{ for all } k, j \neq k \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

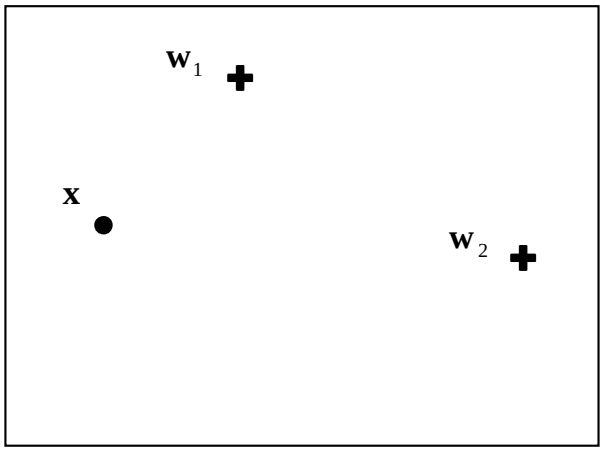
2. **Learning:** Let w_{ij} denote the interconnection weight from input node i to neuron j. The competitive learning rule updates the interconnection weights such that the weight vector $\mathbf{w}$ of the winner neuron moves toward the input pattern $\mathbf{x}$, formalized as

$$\Delta w_{ij} = \begin{cases} \eta(x_i - w_{ij}) & \text{if neuron} j \text{ wins the competition} \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

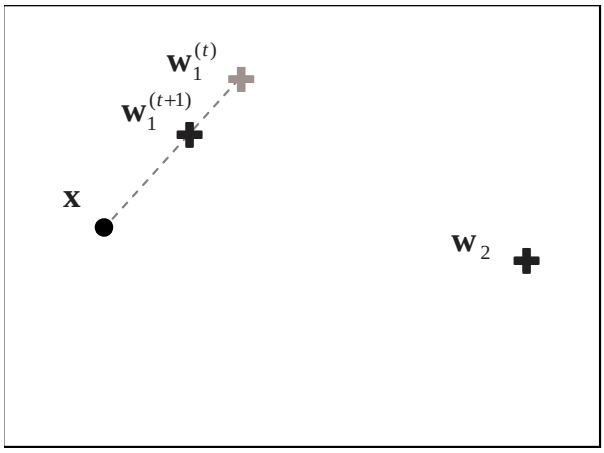
where $\eta \in [0, 1]$ is the learning-rate parameter.

With the above learning activity, it is understandable that, when the grouping of the input patterns is sufficiently distinct, with a proper scaling and upon reaching a stable state, each interconnection weight of the output neurons will thus reflect the centroid of a discovered grouping (Figure 3.3). Clustering is thus achieved.

Readers should note the above prerequisite of applying competitive learning for clustering: there exist distinct natural groupings of the input data. Previous study by Grossberg [Gro82] already pointed out that competitive learning may result in oscillation if the input data are too densely distributed, because the network may not respond to a given input pattern with the same output neuron. Figure 3.4 gives an example on which competitive learning fails to produce a stable output. A common practice to tackle this deficiency is to use a decreasing learning rate. The

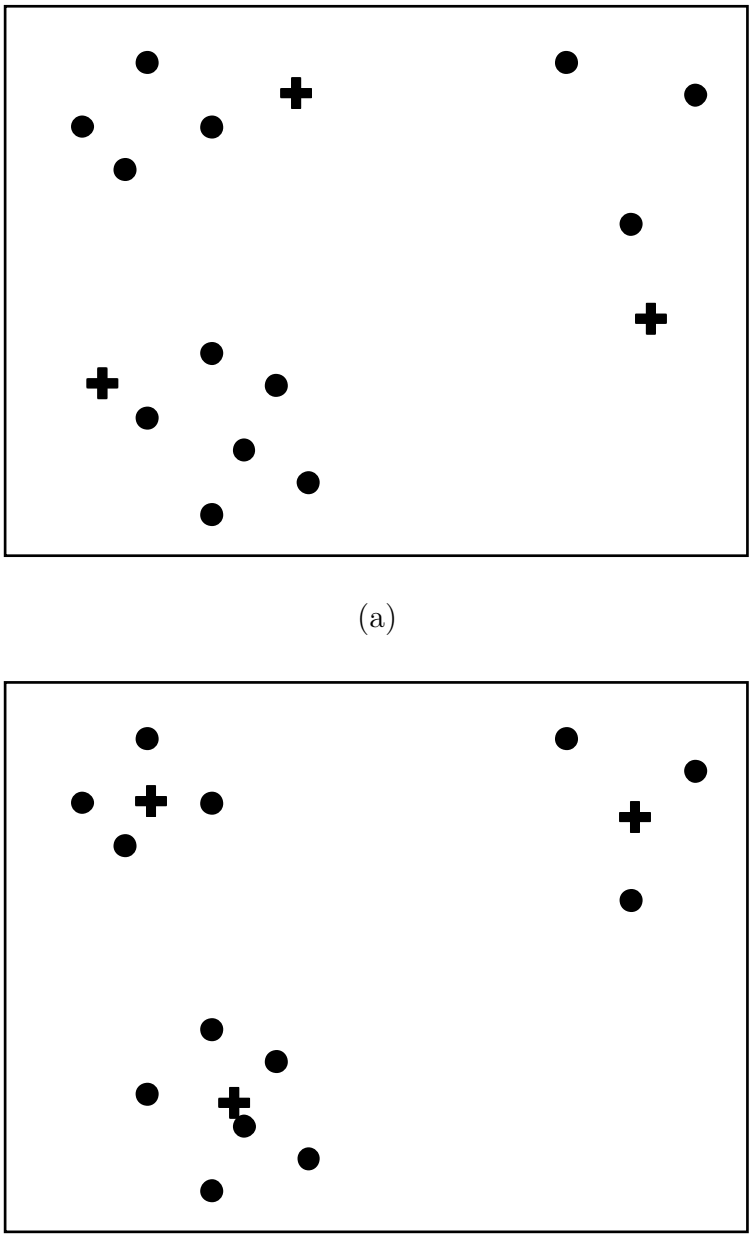


(a)



(b)

Figure 3.2: The competitive learning process. The dot represents the input point. The crosses represents the interconnection weights of the two output neurons. (a) The state of the network before learning. Neuron 1 is selected as the winner because it is closer to $\mathbf{x}$ than 2. (b) The state of the network after learning. The weights $\mathbf{w}_1$ is hence adjusted so that the data point corresponding to $\mathbf{w}_1^{new}$ moves towards to $\mathbf{x}$. The weights of neuron 2 remains unchanged.



(a)

(b)

Figure 3.3: Competitive learning applied to clustering. The dots represent the input points. The crosses represent the interconnection weights of the output neurons. (a) Initial state of the network. (b) Stable state of the network.

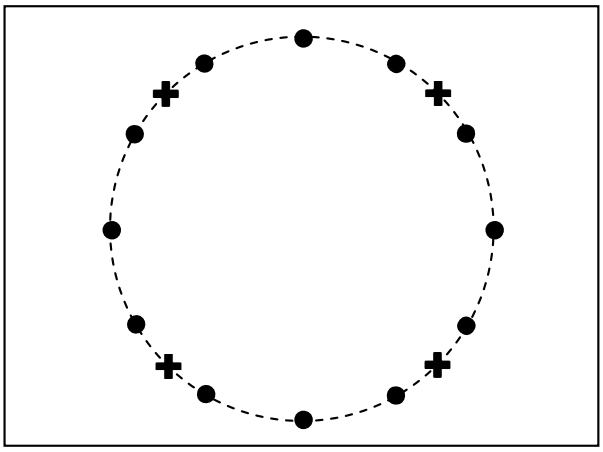


Figure 3.4: A task on which competitive learning will cause oscillation. When the input points are presented in a clockwise order and the learning rate is fixed, the weights of the output neurons will perturb but the network won't reach a stable state.

convergence of such practice was proved by Bottou and Bengio [BB95]. Figure 3.5 summarizes a few common practices for the learning rate decrease.

Another deficiency of the competitive learning process is its dependency on the presentation order of the input data, as the learning process is triggered by individual samples. That is, the output of the network usually falls into one of its local optima which is deterministic to the input order. Figure 3.6 illustrates this.

It is possible to build a simple, single layer feed forward neural network using the competitive learning rule. As a matter of fact, such a feed forward neural architecture is essentially equivalent to online K-Means, which is widely studied in the machine learning domain and reviewed in Section 2.3.1. Besides online K-Means, there are numerous algorithms (including neural networks) which adopt the competitive learning rule. Table 3.1 lists a few examples though a topology review.

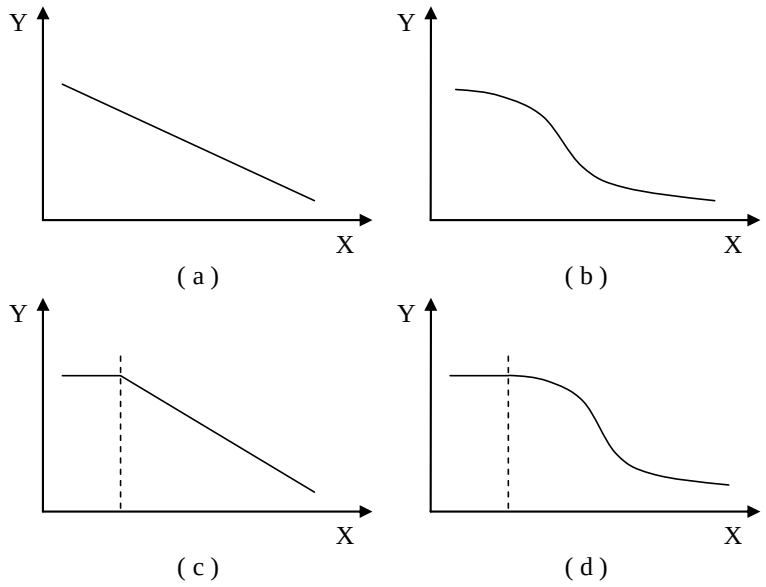
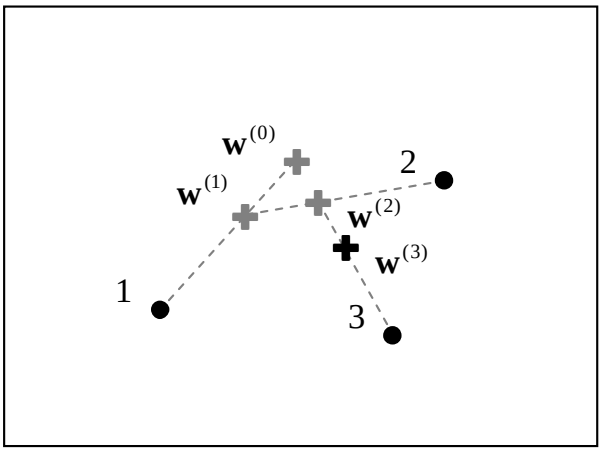
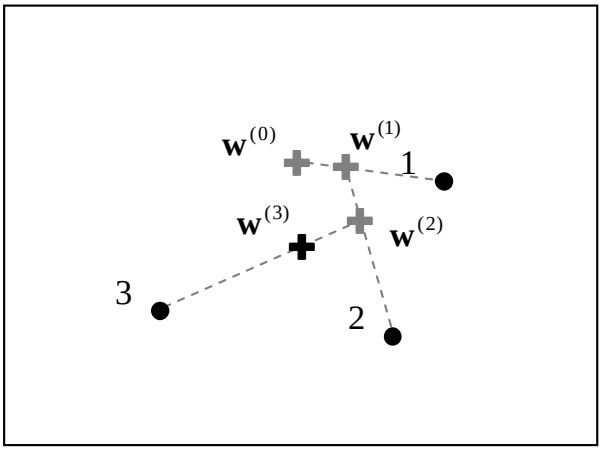


Figure 3.5: Examples of common practices for competitive learning rate decrease: (a) Linear function, (b) Gaussian function, (c) Linear function with threshold activator and (d) Linear function with threshold activator. The X-axis is the progress of the competitive learning, identified with either number of training samples, number of learning iterations or time stamps. The Y-axis is the learning rate.



(a)



(b)

Figure 3.6: The different input orders that affect the competitive learning process. The dots represent the input points, represented in consequence as marked by numbers 1, 2 and 3. Suppose upon presentation of each input point, the winner neuron is the same one, presented with the cross. (0), (1), (2) and (3) indicate the data points corresponding to the interconnection weights of the winner neuron before learning, and after learning of each input. (a) and (b) show that after the learning of three inputs in different orders, the weights of the output neuron as indicated with $\mathbf{w}^{(3)}$ are different.

Table 3.1: A topology review of clustering algorithms based on competitive learning. These algorithms are categorized based on the dimension of the recognition domain (fixed or varying) and the nature of applying the learning rule (hard learning or soft learning).

	Fixed Dimension	**Varying Dimension**
Hard Learning	Lloyd/LBG ([Llo57, LBG80]), K-Means ([Mac67])	ART ([CG87b]), Cluster Euclidean ([Moo89]), SART ([BA98])
Soft Learning	SOM ([KKL$^+$00])	Neuro Gas ([MS91]), Growing Cell ([Fri94]), GHSOM ([DMR00])

Based on the dimension of the recognition domain (number of output clusters), there are methods with a fixed dimension and methods with a varying dimension. Based on the nature of applying the learning rule, there are hard learning (winner-take-all) methods and soft learning (winner-take-part) methods. Among them, the two major families in the neural networks domain, namely the Self-organizing Map (SOM) and the Adaptive Resonance Theory (ART), are reviewed in the subsequent section.

3.4 A Brief Review of Two Families of Competitive Learning Neural Networks

3.4.1 Self-organizing Map (SOM)

Self-organizing Map (SOM), also known as Kohonen feature map, is originally proposed to project and visualize high-dimensional data spaces [Koh97]. Subsequent studies [Fle97, Fle99] pointed out its close relation with the online K-Means clustering algorithm, which is discussed above. It has been widely applied for clustering and data compression purposes.

In a self-organizing map, the output neurons are placed at the nodes of a multidimensional lattice. The lattice used in most studies is two-dimensional, for the convenience of graphical illustration and human inspection. In response to the input sequence, the neurons are selectively tuned in the course of a competitive learning process. The learning process builds a topological map on the lattice, essentially understood as a number of codewords in a higher-dimensional input space. In the topological map, the coordinates (spacial locations) of the neurons correspond to the particular features of the input patterns, through which the network achieves data compression and visualization (Figure 3.7).

A typical SOM algorithm using Euclidean distance is briefly summarized below.

1. Initialization: Given a training sequence $\mathbf{X} = \{\mathbf{x}_i : i = 1, \ldots, M\}$, randomly initialize the interconnection weights of the output neurons $\{\mathbf{w}_j : j = 1, \ldots, K\}$. Set the initial neighborhood of each node $N_j^{(0)}$ to be large.

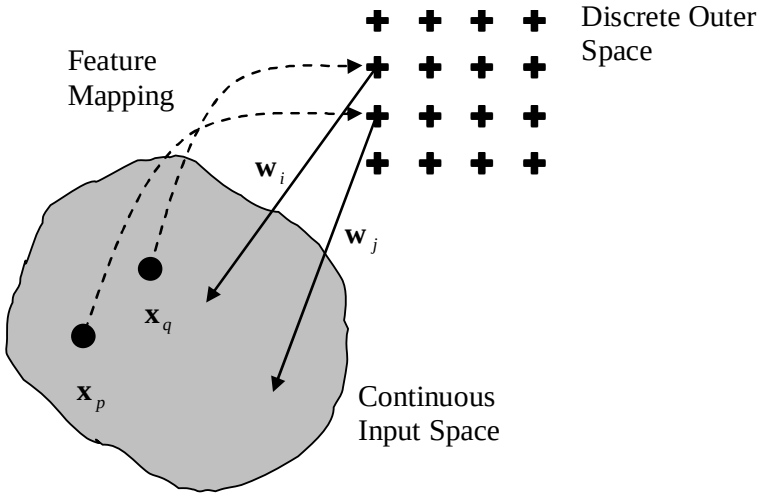


Figure 3.7: The feature map and the weight vectors of the output neurons in a self-organizing map neural architecture.

2. Winner search: Given an input $\mathbf{x}_i$, find the winner node J such that $||\mathbf{x}_i - \mathbf{c}_J|| \leq ||\mathbf{x}_i - \mathbf{c}_j||$ for $j = 1, \ldots, K$ and $j \neq J$.

3. Learning: Update the weights of the winner node and its neighbors, according to

$$\mathbf{w}_j^{(t+1)} = \mathbf{w}_j^{(t)} + h(j, J)\eta^{(t)}(\mathbf{x}_i - \mathbf{w}_j^{(t)}) \text{ for each } j \in N_J^{(t)}, \tag{3.3}$$

   where $h(j, J) \in [0, 1]$ is a scalar *kernel* function that gives a higher weight to a closer neighbor of the winner node J and $\eta^{(t)}$ is the learning rate.

4. At the end of each learning iteration, shrink the neighborhood so that $N_j^{(t+1)} \subset N_j^{(t)}$ for each j and decrease the learning rate so that $\eta^{(t+1)} \leq \eta^{(t)}$. Repeat from step 2 till convergence.

There are a large number of SOM variations depending on the dimension of the organization map, the definition of the pattern distance, the definition of the

neighborhood N_i, the kernel function $h()$ as well as the paradigm that iteratively re-adjusts N_i and η.

The feature of the SOM learning algorithm, which distinguishes it from the online K-Means algorithm, is the integral neighborhood function, which is centered around the neuron that wins the competition upon the presentation of an input. Tuning the weights of the neurons in the neighborhood helps in generating the topology in the way that the discovered patterns which are similar to each other are placed close to each other in the lattice. This feature is significantly useful for visualization purpose.

Albeit the above useful feature, no proof on the convergence of the network with non-zero neighborhood is given in the literature [Fle99]. A common practice to ensure the network's convergency is, after properly building the typological ordering of the output weights, to shrink the neighborhood till zero (i.e. the neighborhood contains the winner neuron only). With such, the self-organizing map works identically as an online K-Means clustering algorithm to reach a local optimally stable state. The convergence of online K-Means has been proved by Bottou and Bengio [BB95].

With such a training process, SOM's learning can be understood as two major stages, namely

1. **Lattice Construction:** The prototype of the output neurons are preliminarily estimated; the ordering of the output neurons on the lattice is built.

2. **Convergence:** The weights of the output neurons are further fine tuned such that the network reaches a stable state.

SOM is not a fast algorithm in this sense, due mainly to the extra computational cost in maintaining the neighborhood relationship for constructing the lattice and the decreasing learning rates adopted to reach convergence.

In addition to the above, since SOM adopts the competitive learning rules, the features of competitive learning are inherited to SOM, which include the dependence on the initialization of the output neurons and the dependence on the presentation order of the inputs.

3.4.2 Adaptive Resonance Theory (ART)

The Adaptive Resonance Theory (ART) architecture was first introduced by Stephen Grossberg in 1976 [Gro76a, Gro76b] to satisfy the *stability-plasticity dilemma*, which many of the networks at that time failed at. Subsequent work has led to a large number of ART modules. The representative architectures include ART 1 which handles arbitrary sequences of binary input patterns [CG87b], ART 2 which extends ART 1's capability to handle both binary and analog inputs [CG87a], ART 2A which provides an optimized searching paradigm over ART 2 [CGR91b], ART 3 which carries out parallel and hierarchical search [CG90], as well as Fuzzy ART which follows a similar paradigm of ART 2A and incorporates fuzzy set theory [CGR91c].

There are also a number of improved ART modules and ART-like architectures in the literature. Examples include Cluster Euclidean which can be understood as an ART architecture using Euclidean distance [Moo89], AHN which essentially is an ART 1 architecture with improved efficiency [HL95] and SART

which is a simplified class of ART that uses bidirectional (symmetric) category choice and match functions [BA98]. Extended work has also produced numerous modular ART architectures to handle more complicated tasks, such as Cascade ART [HLL+96], HART-J and HART-S [BW00] for hierarchical clustering; and ARTMAP [CGR91a], Cascade ARTMAP [Tan97] and ARAM [Tan95] for supervised learning.

A conventional ART architecture consists of three layers depicted in Figure 3.8: the input layer (F_0), the comparison layer (F_1) and the recognition layer (F_2). The input layer F_0 receives and stores the input patterns. Neurons in the input layer F_0 and comparison layer F_1 are one-to-one connected with hard-coded links, which corresponds to a normalization preprocess to prevent category proliferation. The comparison layer F_1 stores the short term memory for the current input pattern while the recognition layer F_2 stores the prototypes of recognition categories (clusters) as the long term memory. Interactions among these three layers in turn form two subsystems in the architecture, namely the attentional subsystem, which reacts with respect to the input, and the orienting subsystem, which guides the learning activities of the attentional subsystem.

In Carpenter et al's original prototype [CGR91b], the F_2 layer initially contains a number of so-called *uncommitted* nodes, which one by one will conditionally get *committed* upon input presentation. This however may give a wrong impression that ART uses "a fixed number of output nodes which limit the number of clusters that can be produced" [JMF99]. As an alternative interpretation, a number of subsequent studies (such as [BW00, Tan95]) refer the F_2 layer initially as a null set (i.e. contains no node) which dynamically grows by creating new recognition

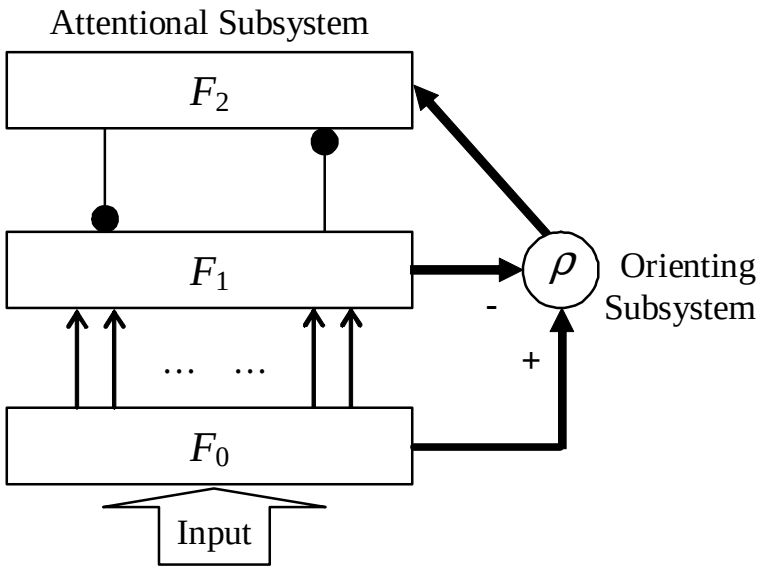


Figure 3.8: The ART Architecture.

categories (committed nodes) using distinct inputs. We follow the later interpretation in the rest of this thesis, as it highlights ART's capability of expanding the scale of its recognition field indefinitely.

The ART network follows a winner-take-all competitive learning process. Learning of the ART 1 and ART 2 networks involves modification of the weighted bottom-up (feed-forward) and top-down (feed-backward) connections between F_1 and the recognition layer F_2. These connections are further simplified with the bottom-up (feed-forward) connections only in subsequent studies, including ART 2A and Fuzzy ART. The interactions between F_1 and F_2 are controlled by the orienting subsystem using a vigilance threshold ρ.

Unlike SOM which initializes its fixed-dimensional recognition field using either random or heuristic criteria, ART initializes its recognition neurons using distinct inputs in a dynamic way. This initialization mechanism is primarily controlled by the vigilance parameter ρ. Briefly, when the similarity between the input and the

winner pattern reaches a certain degree, evaluated by ρ, ART follows the same learning rule of competitive learning, otherwise, a new recognition pattern will be created to directly encode the distinct input. The vigilance parameter guarantees a threshold on the variance of the recognition pattern caused by the learning function. This hence enables ART to carry out online competitive learning with relatively high learning speed. In our practise, ART has no difficulty in learning at an initial rate of 0.2 and above (Fuzzy ART's initial learning rate can be as high as 1.0), as compared with those of SOM and K-Means which are usually initialized below 0.1.

On the other hand, the dynamic initialization of recognition neurons in ART causes a number of shortages in this infrastructure. These include its dependency on the global vigilance threshold ρ, as well as its incapability of limiting the dimension of the recognition domain and hence the possibility in generating excessive recognition neurons (output clusters). This understanding motivates our exploration of a novel neural network architecture. The novel neural architecture, based on ART, retains the high efficiency and online initialization characteristics of ART and complements the above capability.

For a better understanding of the thesis, the characteristics of ART are analyzed in more details in the subsequent chapter, followed with the analysis of the novel neural architecture proposed in this thesis.

CHAPTER 4

ADAPTIVE RESONANCE THEORY UNDER CONSTRAINT

4.1 Introduction: The Motivation

As briefly reviewed in Section 3.4.2, the Adaptive Resonance Theory (ART) [CG87b] is a family of neural networks that develop stable recognition categories (clusters) by self-organization in response to arbitrary sequences of input patterns. Through dynamic creation of recognition categories for encoding distinct input samples, an ART module is capable of self-adjusting the scale of its recognition field, in terms of the number of committed nodes, with respect to the complexity of the problem domain. Its *fast commitment* mechanism and capability of learning at moderate speed guarantees a high efficiency.

While ART has its advantages in its adaptability and high learning efficiency, it is not recognized as the all-in-one solution to real-life applications. First, given a problem set, the dimension of ART recognition domain (i.e. the number of output clusters) depends on a global threshold parameter called *vigilance*, which essentially reflects the maximal variance of the data in each recognition neuron. Therefore, in order to control ART's recognition representation, the end-user is required to have prior knowledge in the distribution of the input samples and the estimation of the pattern similarity (or variance) within each desired cluster. Such knowledge however is unlikely to be available in the real life. Second, ART is designed with the capability of expanding its recognition domain indefinitely. If the vigilance value is not properly set, this in turn may generate excessively large number of output clusters which is beyond the end-user's control.

A representative example to the above problems is the online news topic detection task. In such application, the relation between the existing topics and emerging topics is difficult for the end-user to predict; the coverage of the new topics is unknown; and the total number of machine generated topics is required to be under control for human inspection. While an experienced user may fine tune the vigilance parameter of ART in order to obtain a satisfactory clustering result, it would be more convenient if the user could intuitively set how many topics he/she wants to generate out of each day's news articles based on his/her experience. So it is desirable to have a variance of ART that carries online learning, detects distinct inputs, while controls the scale of the recognition domain.

This chapter proposes a novel ART-based neural architecture named ART-C (Adaptive Resonance Theory under Constraint). It contains several improvements

over its predecessor which is introduced in 2002 [HTT02]. Our aim is to combine the neuron initialization and the online clustering capabilities of ART with the predictability in allowing a direct control on the number of the output clusters. This capability is achieved by a *constraint reset* mechanism in ART-C that adaptively adjusts the global vigilance threshold of the system and re-organizes the category representation in response to an intuitive constraint. Given a specific data set, the constraint reset mechanism guarantees the scale of the network's recognition field within a quantitative limit, and adaptively adjusts the network's vigilance value to satisfy the constraint. With such, ART-C complements ART's capacity in handling problems where the optimal number of clusters is more conceivable to the end-user than the intra-cluster similarity, or where a hard cap on the number of output clusters is required for human inspection, regardless of the optimal number of clusters over the input.

The remainder of this chapter extends the review on ART's learning paradigm, analyzes its characteristics, introduce the ART-C architecture in details, and briefly compares ART-C with ART in terms of the parameter settings, requirements on the user's prior knowledge and their common use.

4.2 The ART Learning Algorithm: An Extended Analysis

This section reviews the learning paradigm of ART in more details for a better understanding of the thesis. Under the common ART architecture as briefed in Section 3.4.2, there are a number of variations depending on the pattern proximity measures used by the network. The learning algorithm of the ART 2A variation

which is based on the inner dot similarity is given below, followed with the Fuzzy ART variation which is based on fuzzy logic.

4.2.1 The ART 2A Learning Algorithm

The ART 2A network introduced by Carpenter et al. [CGR91b] is a variation of the ART 2 network [CG87a]. Compared with ART 2, ART 2A adopts a simplified feed-forward architecture, with an optimized search strategy in identifying the winner node. The ART 2A learning algorithm uses dot product as the pattern proximity measure. The algorithm is summarized as below.

Parameters: The ART 2A dynamics are determined by the vigilance parameter $\rho \in [0, 1]$ and the learning rate $\eta \in [0, 1]$.

Network initialization: The category layer F_2 is initialized with the null set $\emptyset$ (i.e. contains no category).

Input normalization: The non-zero input vector $\mathbf{x}^0$ presented to F_0 is normalized according to

$$\mathbf{x} = \Re\mathbf{x}^0 \tag{4.1}$$

where the normalization function $\Re$ is given by

$$\Re\mathbf{x} \equiv \frac{\mathbf{x}}{||\mathbf{x}||} = \frac{\mathbf{x}}{\sqrt{\sum_i x_i^2}}. \tag{4.2}$$

Category choice: Given an F_1 input vector $\mathbf{x}$, for each F_2 node j, the *choice* function $T(\mathbf{x}, \mathbf{w}_j)$ is defined by

$$T(\mathbf{x}, \mathbf{w}_j) = \mathbf{x} \cdot \mathbf{w}_j \tag{4.3}$$

where $\mathbf{w}_j$ is the weight vector of node j. The system is said to make a choice when at most one F_2 node can become active. The choice is indexed at J where

$$T(\mathbf{x}, \mathbf{w}_J) = \max\{T(\mathbf{x}, \mathbf{w}_j) : \text{for all } F_2 \text{ node } j\}. \tag{4.4}$$

Resonance check: Resonance occurs if on the category selected above, the *match* function $M(\mathbf{x}, \mathbf{w}_J)$ in the orienting subsystem meets the vigilance criteria, i.e.

$$M(\mathbf{x}, \mathbf{w}_J) = \mathbf{x} \cdot \mathbf{w}_J \geq \rho, \tag{4.5}$$

during which learning ensues, as defined below. If the vigilance constraint is violated, *mismatch reset* occurs. In a mismatch reset, the above category is excluded from the search process (as in Equation 4.4) for the duration of the input presentation and the search process is repeated. In case the network fails to find an existing category who meets the vigilance criteria, a new category K is created by copying $\mathbf{x}$ as its weight vector

$$\mathbf{w}_K = \mathbf{x}. \tag{4.6}$$

Learning: Once the search ends and the resonance is achieved, the attentional subsystem updates the weight vector $\mathbf{w}_J$ according to

$$\mathbf{w}_J^{t+1} = \Re(\eta\mathbf{x} + (1-\eta)\mathbf{w}_J^t). \tag{4.7}$$

Some minor differences between the above review and the original ART 2A proposed in [CGR91b] are worth extended explanations. In Carpenter et al's version, a threshold θ is used to cut off the attribute value of the input vector such that if $x_i < \theta$, x_i is reset to be 0. Similar cut-off is applied on the weight vector as well during learning. This is claimed to "*effectively distinguish features*

that are irrelevant in given categories" [CGR91b]. As in the weight vector, once an attribute value drops below the threshold, the value will remain zero in the further learning. However, suggesting an appropriate "irrelevance" threshold value requires prior knowledge and may be quite subjective. In addition, most clustering system assume there is an effective feature selection preprocess and all features presented to the system are equally important. Therefore we follow a common setting of $\theta = 0$ for simplicity of analysis. Carpenter et al. also use a small constant α such that the so-called *uncommitted* nodes are forced to have a nominal, minor "similarity" of $\alpha \sum_i x_i$ with the input pattern. This follows that in some simulations, even when $\rho = 0$, some uncommitted nodes may be activated and the system may generate a few number of categories [CGR91b]. Readers should note this is practically equivalent to the result produced by the paradigm we summarized above, using a very small ρ value.

Optimizing the ART 2A Learning Algorithm

For formalization purpose as well as the consistency with the Fuzzy ART algorithm (summarized in the following sub-section), the ART 2A algorithm above is said to adopt a *choice* function and a *match* function. As a reality, since ART 2A adopts the same, symmetric inner dot function to evaluate the pattern proximity, it is understandable that both *choice* and *match* functions can be consolidated into one similarity measure.

In addition to the above, during resonance check, if the first selected winner $\mathbf{w}_J$ does not meet the vigilance criteria, it is not necessary to find another winner as $T_j < T_J$ for all $j \neq J$. As such, the algorithm can be further optimized in the

way that, once the first mismatch reset happens in respect to an input, then create a new category using the input.

4.2.2 The Fuzzy ART Learning Algorithm

The Fuzzy ART network, based on fuzzy logic theory, generally adopts the same architecture and follows the identical learning process as that of ART 2A. The major variations lie on the different set of input normalization, *choice*, *match* and *learning* functions for Fuzzy ART.

Fuzzy ART preprocesses the input vectors by first-level normalization and/or complement coding to avoid category proliferation. Specifically, Fuzzy ART works on non-zero input vectors and assumes the attribute values have been appropriately scaled to be non-negative. It is a common practice to limit the attribute value between 0 to 1 to present the "fuzzy possibility", 0 being the lowest possibility and 1 being the highest. The input vector $\mathbf{x}^0$ presented to F_0 is normalized to $\mathbf{x}$ according to

$$x_i = \frac{x_i^0}{\max\{x_i^0\}}. \tag{4.8}$$

Complement coding preserves the input vector's amplitude information and represents both the on-response and the off-response to the input vector by doubling the number of network connections. Given an properly normalized D-dimensional input $\mathbf{x}$, the complement coded F_1 input vector $\mathbf{x}'$ is a $2D$-dimensional vector

$$\mathbf{x}' = (\mathbf{x}, \mathbf{x}^c) \equiv (x_1, \ldots, x_D, x_1^c, \ldots, x_D^c) \tag{4.9}$$

where $x_i^c \equiv 1 - x_i$.

Fuzzy ART utilizes a set of fuzzy logic based pattern proximity measures and learning function, listed as below:

Choice function:

$$T(\mathbf{x}, \mathbf{w}_j) = \frac{|\mathbf{x} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{4.10}$$

where the fuzzy AND operation $\wedge$ is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i), \tag{4.11}$$

α is a predefined constant parameter and the norm $|.|$ is defined by

$$|\mathbf{p}| \equiv \sum_i p_i \tag{4.12}$$

for vectors $\mathbf{p}$ and $\mathbf{q}$.

Match function:

$$M(\mathbf{x}, \mathbf{w}_J) = \frac{|\mathbf{x} \wedge \mathbf{w}_J|}{|\mathbf{x}|}. \tag{4.13}$$

Learning function:

$$\mathbf{w}_J^{t+1} = \eta(\mathbf{x} \wedge \mathbf{w}_J^t) + (1 - \eta)\mathbf{w}_J^t. \tag{4.14}$$

4.2.3 Features of the ART Network

The features of the ART learning algorithm have been extensively studied in the literature. This section summarizes them as below:

1. **Sample-driven neuron initialization:** ART distinguishes itself from others in the way that it self-adjusts its architecture by creating new recognition (output) neurons using the distinct input. Compared with SOM, such

a sample-driven neuron initialization mechanism does not require a pre-scan of the input set. Hence, ART is more suitable for online clustering of incrementally presented data.

2. **High efficiency:** The resonance check limits the learning of the network in a constrained sub-space. Within the sub-space, the network is capable of learning at a relatively high learning rate. Particularly, upon a distinct input sample, the network essentially learns at a learning rate of 1.0 for fast commitment. Such a practice guarantees the high efficiency of the network. This characteristics of the network is analyzed in the following sub-section in more details.

3. **Outlier sensitivity:** As a side effect of the fast commitment of the network, the network is sensitive to outliers. The ART's output commonly contains a number of dead clusters or clusters each containing a few number of input samples. This deficiency can be tackled by adopting neuron pruning during learning.

4. **Input order dependence:** As a deficiency inherited from the competitive learning rule, ART's learning is highly affected by the presentation order of the input. Such a characteristics can be found in SOM as well, as previously reviewed in this thesis.

4.2.4 Analysis of the ART Learning Characteristics

This sub-section analyzes the characteristics of the ART learning algorithm in more details, for a better understanding of the thesis. Due to the high correlation

between ART 2A and Fuzzy ART's learning paradigm, the analysis focuses on ART 2A only. Here we introduce two definitions for the convenience of our further discussion.

Definition: (Committed Region) The *committed sub-region* S_j of each ART 2A's recognition category j is defined by the vector sub-space that satisfies

$$S_j = \{\mathbf{x} : M(\mathbf{x}, \mathbf{w}_j) \geq \rho\} \tag{4.15}$$

where $\mathbf{w}_j$ is the weight vector of category j, $M(\mathbf{x}, \mathbf{w}_j)$ is the *match* function adopted by the ART network. The *committed region* of ART's recognition space refers to the union of all its committed sub-regions.

$$S = S_1 \cup \ldots \cup S_N \tag{4.16}$$

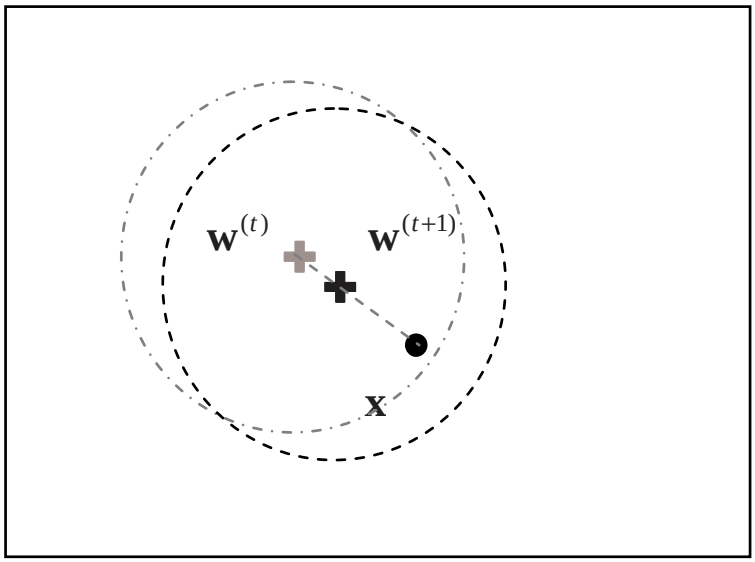
where N is the number of the recognition categories.

Definition: (Uncommitted Region) The *uncommitted region* of ART 2A's recognition space is defined by the complementary sub-space of the committed region

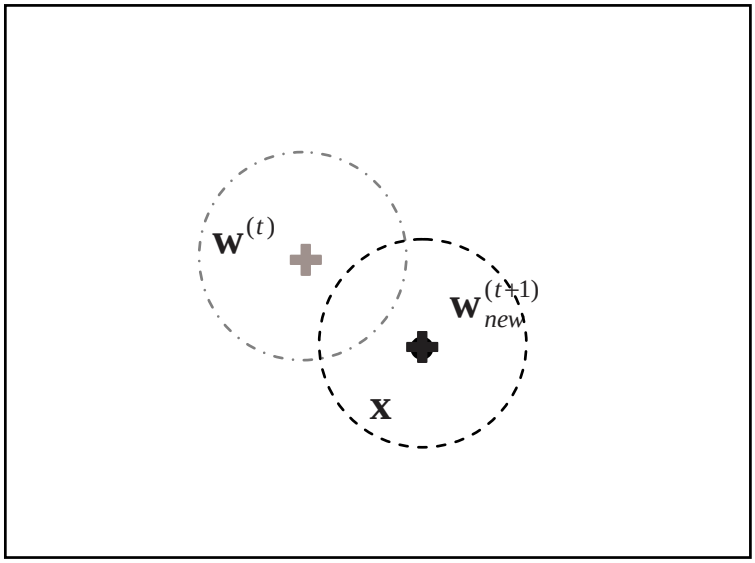
$$S = \{\mathbf{x} : M(\mathbf{x}, \mathbf{w}_j) < \rho \text{ for all } j = 1, \ldots, N\} \tag{4.17}$$

where N is the number of the recognition categories.

Note that the Euclidean normalization function $\Re$ in Equation 4.1 and Equation 4.7 limits ART 2A's learning on a unit hyper-sphere to avoid category proliferation. Under this condition, the choice function in Equation 4.3 is equivalent to the cosine similarity between two patterns.



(a)



(b)

Figure 4.1: The effect of the vigilance threshold on ART 2A's learning. Dashed circles identify the committed sub-region of the corresponding recognition category. (a) When the input falls into a committed sub-region, it is incorporated into the existing category. (b) When the input falls into the uncommitted region, it is used to create a new recognition category.

The network's mismatch reset mechanism brings interesting characteristics to the ART 2A learning paradigm. The vigilance threshold ρ guards whether an input will be incorporated into its most similar recognition category or will be used to generate a new category. Specifically, this threshold forms a circular decision boundary with a radius of $\sqrt{2(1-\rho)}$ around the weight vector of each category, in terms of the Euclidean distance (or ρ in terms of the arc on the unit hyper-sphere), as given $\mathbf{x} \cdot \mathbf{w} = \rho$ and $||\mathbf{x}|| = ||\mathbf{w}|| = 1$, $||\mathbf{x} - \mathbf{w}|| = \sqrt{(||\mathbf{x}||^2 + ||\mathbf{w}||^2 - 2\mathbf{x} \cdot \mathbf{w}} = \sqrt{2(1-\rho)}$. Upon the presentation of a new input, if it falls into one of the radius sub-space (committed region), it will be incorporated into an existing category (Figure 4.1(a)), otherwise the input sample is used to create a new recognition category (Figure 4.1(b)). It is understandable from the search function (Equation 4.4) that, when the Euclidean distance between two nearest categories is less than $2\sqrt{2(1-\rho)}$, the partitioning boundary between these two categories is given by their perpendicular bisector on the hyper-sphere. The joint of these decision boundaries essentially serves as the delimiter between the committed region and the uncommitted region (Figure 4.2).

As mentioned, mismatch reset of the network occurs if the input pattern falls into the uncommitted region. Learning of such an input is done by creating a new category with the input, which turns the sub-region around it given by Equation 4.15 into a committed sub-region. This fast commitment paradigm guarantees stable encoding of new distinct inputs. We would highlight that this characteristics particularly reflects the plasticity of the network.

Network resonance only happens if the input pattern falls into the committed region. Learning activity of the network in the committed region, is closely related

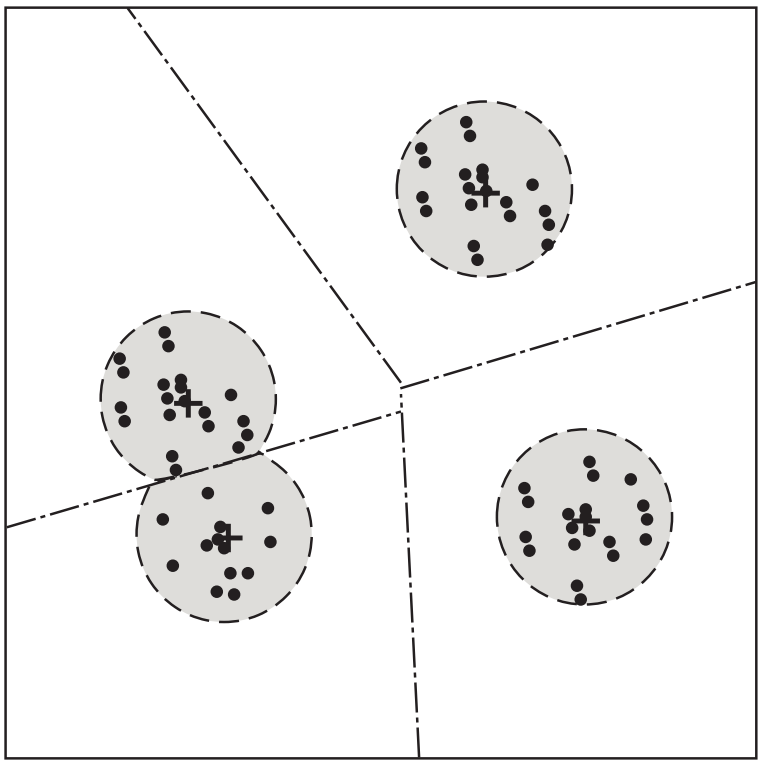


Figure 4.2: The decision boundaries (dashed lines), the committed region (gray) and the uncommitted region (white) of the ART 2A network being viewed on the unit hyper-sphere. The crosses identify the weights of the recognition categories. Vector quantization is done by relaxing the decision boundaries with $\rho = 0$ (dot-dashed lines) after the network convergence on the input sequence.

to the naive competitive learning paradigm as reviewed in Section 3.3.

With such a training process, we may understand the learning of ART as two parallel operations, namely

1. **Structure Adaption:** The network establishes an overall estimation of the data distribution by creating a number of recognition categories which correspond to the set of committed sub-regions that fully cover every input representations in the vector space.

2. **Input Encoding:** The weights of the recognition categories are fine tuned toward a local minimum of encoding error.

Network convergence corresponds to the stable establishment of the categories as well as a local minimum of encoding error. Upon network convergence, each input presentation will incur the resonance of the network. Hence the mapping of the input to the recognition category $q = \mathbf{x} \rightarrow \mathbf{w}$ is simplified as the network's category choice process, which essentially is a nearest-neighbor mapping. With such, the weight $\mathbf{w}_j$ can be used as the representation protocol of each encoding sub-region. Although the committed region of the network may not cover the whole input vector space upon network convergence, the simplified mapping essentially forms a set of relaxed decision boundaries by setting $\rho = 0$ for the specific input sequence (Figure 4.2). Clustering is thus done on the whole input space.

The restriction from the boundaries of each committed sub-region also explains the notably high efficiency of the ART network, compared with a conventional competitive learning paradigm. As mentioned above, network resonance

which triggers the network's learning activity is achieved only in an existing committed sub-region with a restricted coverage. This provides an upper bound for the variance of the newly learnt cluster prototype $\mathbf{w}^{(t+1)}$ from $\mathbf{w}^{(t)}$, specifically $||\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}|| \leq \sqrt{2(1-\rho)}$. Therefore the network is capable of learning at either slow or intermediate speed. As a comparison, a conventional competitive learning paradigm may cause cluster oscillation if the learning rate is too high, and hence is capable of learning at a slow speed only. Combination of this fast learning capability with the fast commitment mechanism ensures ART's capability of achieving stable encoding of input sequences with very few number of learning iterations in practice.

We however note a major deficiency of ART when it is used for clustering: the vigilance threshold ρ affects the number of ART 2A recognition categories generated on a specific input sequence in a major way. To explain in more details with ART 2A as the example, the coverage of a committed sub-region is a circular area with a maximal radius of $\sqrt{2(1-\rho)}$. The higher the ρ value, the lower is the coverage of each recognition region and the larger number of recognition categories required to encode a specific vector space. Specifically, $\rho = 1$ causes each unique input to be encoded as one separate category whereas $\rho = 0$ causes all inputs to be encoded into the same category.

Figure 4.3 illustrates the number of ART 2A's recognition categories (output clusters) with respect to different vigilance thresholds on several real-life data sets. In order to obtain a specific number of partitions over the input space, prior knowledge on the distribution of the data set is required to suggest a proper vigilance threshold. Without such knowledge, a user has to follow a trial-and-error

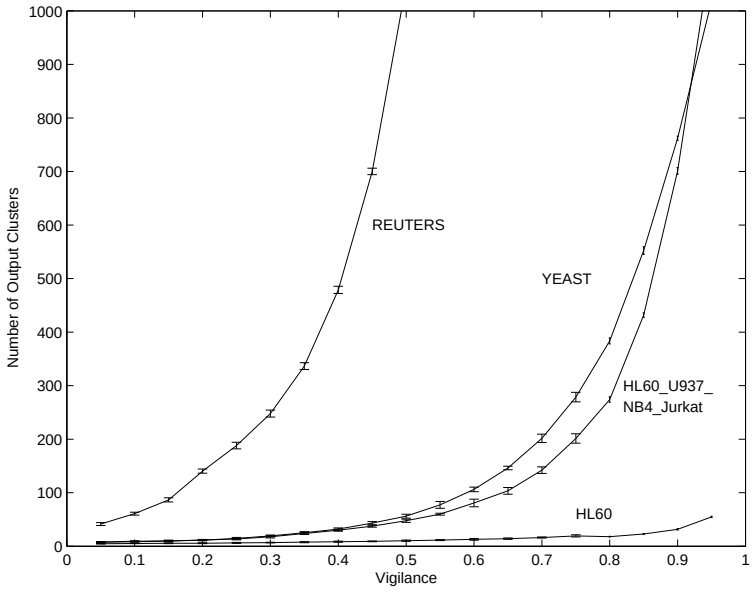


Figure 4.3: The number of ART 2A's output clusters with respect to different vigilance parameter values on different data sets, namely the Reuters-21578 top 10 category subset (REUTERS), the yeast cell cycle data set (YEAST) and the human hematopoietic data sets (HL60 and HL60_U937_NB4_Jurkat). Values are shown with means and standard deviations over ten runs using different input orders. Suggesting an appropriate vigilance value in order to get the specific number of output clusters requires prior knowledge on the distribution of the data set.

process to decide the appropriate vigilance threshold.

The deficiency of ART motivates our study of the ART-C learning paradigm, which removes the network's dependency on the vigilance parameter in its learning by incorporating a user-defined constraint on the category representation, in terms of the number of recognition categories. In the following section, we give details of the ART-C paradigm.

4.3 Adaptive Resonance Theory under Constraint (ART-C)

4.3.1 The ART-C Architecture

The Adaptive Resonance Theory under Constraint neural architecture, proposed in this PhD work first in 2002 [HTT02] and subsequently improved in 2004 [HTT04], is an ART-based architecture capable of performing online clustering of arbitrary input sequences while keeping the size of the recognition category field in a desired scale. The aim of the study is to combine the neuron initialization and the online clustering capabilities of ART with the predictability in allowing a direct control on the number of the output clusters.

The ART-C architecture is illustrated in Figure 4.4. Compared with the standard ART architecture, there is an additional constraining subsystem in the ART-C network. During learning, the constraining subsystem interacts with the attentional subsystem and the orienting subsystem. It adaptively estimates the distribution of the input data and self-adjusts the vigilance parameter for the orienting subsystem, which in turn governs the learning activities in the attentional subsystem.

Unlike a conventional ART network that mainly controls its learning activity with a vigilance threshold ρ, ART-C's learning is mainly guided by an intuitive constraint C on the maximal number of recognition categories in the F_2 layer. Such capability is achieved by introducing the *constraint reset* mechanism to the ART network. The constraint reset adaptively estimates the input distribution, and self-

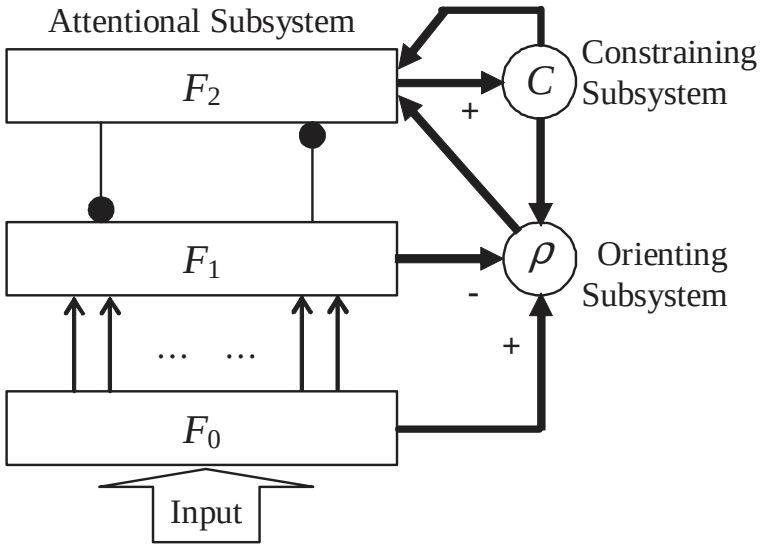


Figure 4.4: The ART-C Architecture.

adjusts the vigilance threshold in response to the constraint C. The dynamically adjusted vigilance threshold in turn drives the learning activities to satisfy the user-defined constraint. The ART-C 2A and Fuzzy ART-C learning algorithms respectively based on ART 2A and Fuzzy ART are introduced below.

4.3.2 The ART-C Learning Algorithm

Parameters: The ART-C dynamics are determined by the constraint C on the number of recognition categories and the learning rate $\eta \in [0, 1]$. For Fuzzy ART-C, there is an additional parameter α being used in the choice function (Equation 4.10).

Network initialization: The category layer F_2 is initialized with the null set $\emptyset$. The vigilance ρ for the orienting subsystem is initialized with a maximal value 1.0 (or a value optionally specified by the user).

Learning of each input presentation: Learning of each input presentation follows the ART 2A learning paradigm.

Constraint check: Constraint check is performed after the learning of each input presentation. The check compares the number of existing recognition categories N with the predefined constraint C

$$\zeta = \begin{cases} 1 & \text{if } N > C \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

The constraint is said to be satisfied with $\zeta = 0$, upon which the network carries on to learn the next input representation. Otherwise *constraint reset* occurs, as described below.

Constraint reset: Constraint reset re-organizes the recognition categories in the F_2 layer towards the satisfaction of the constraint and adjusts the ρ value based on the current category distribution. The process is introduced as follows.

1. *Search of the nearest category pair:* For each category pair $(i,\ j)$ in the F_2 layer, their similarity, noted as $H(i,j)$, is defined by the corresponding ART module's *choice* function of their corresponding weights $\mathbf{w}_i$ and $\mathbf{w}_j$, such that

$$H(i,j) \equiv T(\mathbf{w}_i, \mathbf{w}_j). \tag{4.19}$$

   To further explain, for ART-C 2A that is based on ART 2A,

$$H(i,j) \equiv T(\mathbf{w}_i, \mathbf{w}_j) = \mathbf{w}_i \cdot \mathbf{w}_j, \tag{4.20}$$

   whereas for Fuzzy ART-C that is based on Fuzzy ART,

$$H(i,j) \equiv T(\mathbf{w}_i, \mathbf{w}_j) = \frac{|\mathbf{w}_i \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}. \tag{4.21}$$

The *nearest neighbor* of each category i, indexed as $J(i)$, is the category that has the maximal similarity with i:

$$H(i, J(i)) = \max\{H(i, j) : j = 1, \ldots, N, j \neq i\}. \tag{4.22}$$

The *nearest neighbor similarity* of category i, marked as $\hbar(i)$ then refers to the similarity between category i and its nearest neighbor $J(i)$:

$$\hbar(i) \equiv H(i, J(i)). \tag{4.23}$$

The *nearest category pair*, indexed as $(I,\ J)$, is identified by the category I that has the maximal nearest neighbor similarity to its nearest neighbor J:

$$\hbar(I) = H(I, J(I)) = \max\{H(i, J(i)) : i = 1, \ldots, N\}. \tag{4.24}$$

2. *Adjustment of the vigilance:* The vigilance value $\rho^{(new)}$ for subsequent learning is decreased according to:

$$\rho^{(new)} = \max\{\hbar(i) : \text{all } i \text{ whose } \hbar(i) < \rho^{(old)}\}, \tag{4.25}$$

and thus $\rho^{(new)} < \rho^{(old)}$.

3. *Merging of the nearest category pair:* Merging of the nearest category pair $(I,\ J)$ is done by inserting a new category L, whose weight vector incorporates the weights of these two categories using the learning function with a learning rate $\eta = 0.5$. That is, for ART-C 2A,

$$\mathbf{w}_L = \Re(0.5\mathbf{w}_I + 0.5\mathbf{w}_J), \tag{4.26}$$

while for Fuzzy ART-C,

$$\mathbf{w}_L = 0.5(\mathbf{w}_I \wedge \mathbf{w}_J) + 0.5\mathbf{w}_J. \tag{4.27}$$

In addition, the categories I and J are deleted from the recognition layer F_2 after the creation of the new category. With such, the patterns of the categories I and J are estimated with the pattern of the new category L.

Each constraint reset cycle decreases the number of recognition categories in F_2 by one. Theoretically the constraint check and constraint reset processes should be repeated till the network satisfies the constraint (i.e. $\zeta = 0$). However, considering that the nature of the ART learning is to create at most one new recognition category for encoding of each input, constraint reset practically occurs only when $N = C + 1$. Therefore constraint reset happens at most once on each input representation, after which the number of recognition categories in the F_2 layer is decreased from $C+1$ to C. It is also understandable that constraint reset can only happen right after a mismatch reset, which is the direct cause of the increase of the number of recognition categories from C to $C + 1$.

4.3.3 Structure Adaptation of ART-C

We recall the two parallel operations in ART's learning process, namely structure adaption and input encoding. During input encoding, ART-C adopts the same learning rule as ART's. However ART-C distinguishes itself from ART in its novel way in structure adaptation.

It is understandable that the constraint C is imposed upon the category establishment of the network learning. Given the input sequence $X = \{\mathbf{x}_i : i = 1, \ldots, M\}$, as long as the constraint is satisfied, i.e. $N \leq C$, ART-C follows the identical learning paradigm as ART's. If the constraint is found broken, that is,

when the number of recognition categories reaches $C + 1$ on a sub-set of input $\hat{x} = \{\mathbf{x}_i : i = 1, \ldots, P\}, P \leq M$, the constraint reset is activated to re-estimate the new parameters of the network without losing the learning history.

We should point out that this online estimation of the data distribution is a difficult task, as the exact input history is not accessible during incremental learning. The network has to rely on the $C+1$ existing recognition patterns instead. In order to do such, the constraint reset process essentially makes two modifications on the network. Firstly it locates the most similar recognition categories and represent their patterns with a new category which encodes the learnt knowledge with less detail. Secondly it decreases the vigilance threshold so that the future learning of the network is less sensitive to distinct inputs.

Given a network with C existing recognition categories and a new category K created by the most recent mismatch reset, there are two possibilities for the constraint reset operation:

1. $K = I$ or $K = J$. In this case the learning of the network is practically equivalent to incorporating the most recent input into the most similar existing cluster with a compromised resonance criteria (Figure 4.5a) as well as a fixed learning rate $\eta = 0.5$.

2. $K \neq I$ and $K \neq J$. In this case the constraint reset process essentially moves one of the recognition category from a high density area (I and J) to the newly detected low density area (K), and re-represent I and J with a more general recognition category L, in the sense that committed sub-region of category L covers a larger area of space than those of I and J (Figure 4.5b).

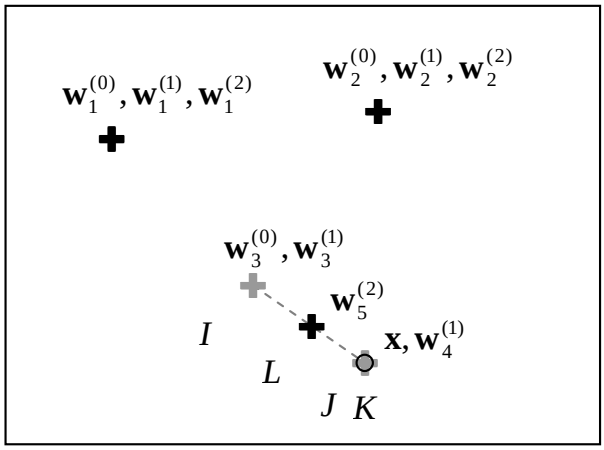
It is understandable that in both cases only the patterns of maximal two existing recognition categories are re-estimated. The patterns of the remaining $C - 1$ recognition categories are not affected.

The constraint reset mechanism identifies the essential difference between ART-C and ART. Specifically, to construct the committed recognition region with a full coverage of the input space, ART only increases the number of the committed sub-regions, each having a fixed maximal coverage area. In contrast, when the number of the committed sub-regions has reached a specific scale C, no more committed sub-region will be added into the ART-C recognition field. Instead, the ART-C network redistributes the committed sub-regions and expands the area of each committed sub-region by using a decreased vigilance threshold ρ.

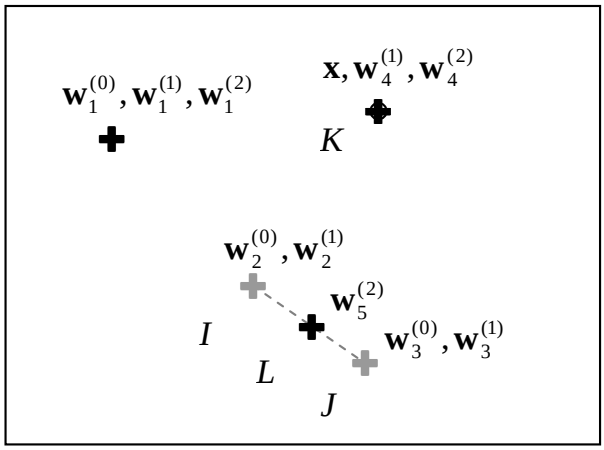
4.3.4 Variations of ART-C

Readers may note that discussions on the ART-C system, until this point, are based on the initial parameter setting of $\rho = 1.0$. With the initial vigilance value set to this highest value 1.0 and a constraint C, each of the first C unique input samples presented to ART-C will be used to create new recognition neurons. As long as the total number of unique input samples satisfies $M > C$, constraint reset happens for at least once on this input sequence. With constraint reset, the vigilance value ρ monotonically decreases to satisfy the constraint. This in turn yields exactly C recognition categories.

However, it is possible that the ART-C may be initialized with a vigilance value that is less than 1.0. The use of different parameter combinations of ART-C on



(a)



(b)

Figure 4.5: Changing of the ART-C 2A recognition categories being viewed on the unit hyper-sphere. $\mathbf{w}^{(0)}$, $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ respectively are the weights of the network's recognition categories: (0) before the presentation of the input $\mathbf{x}$, (1) after mismatch reset upon the recent representation of the input $\mathbf{x}$ and (2) after constraint reset right following the mismatch reset. (a) The network essentially incorporates the input into its most similar category, with a relaxed vigilance. (b) The network essentially moves one category from the highest density area to a low density area.

different problems thus yields to a number of interesting variations, summarized as following.

Suppose in a given input sequence, there are M unique input samples. Using a vigilance value $\rho = \rho^0$, the conventional ART generates C^0 recognition categories on this input sequence. By initializing ART-C with a vigilance value $\rho = \rho^1$ and setting the constraint to C^1, there are four typical possibilities depending on the parameters:

1. (Case 1) $\rho^1 \geq \rho^0$ and $C^1 = C^0$: Similar to the above discussion, constraint reset happens on ART-C; the vigilance value ρ of ART-C will be adaptively adjusted to the same level of ρ^0; and ART-C generates C^1 (i.e. C^0) recognition categories.

2. (Case 2) $\rho^1 \geq \rho^0$ and $C^1 < C^0$: Similar to the above discussion, constraint reset happens on ART-C; the vigilance value ρ of ART-C will be adaptively adjusted to a value that is less than ρ^0; and ART-C generates C^1 recognition categories.

3. (Case 3) $\rho^1 = \rho^0$ and $C^1 > C^0$: Constraint reset of ART-C will not happen; ART-C works identically to ART; and the actual recognition categories generated by ART-C is C^0.

4. (Case 4) $\rho^1 < \rho^0$ and $C^1 \geq C^0$: This is essentially same as Case 3 but is explicitly listed for a better understanding. Similarly, constraint reset of ART-C will not happen; ART-C works identically to ART; and the actual recognition categories generated by ART-C is less than C^1.

Cases with $\rho^1 > \rho^0$ and $C^1 > C^0$, as well as $\rho^1 < \rho^0$ and $C^1 < C^0$ are excluded from the above analysis as the comparison in these cases do not provide meaningful results. As shown above, ART-C extends ART's learning capacity to satisfy the user's constraint, while remains the potential of working identically to ART.

4.3.5 Related Work

The idea of controlling the category representation of an ART network using varying vigilance values has been investigated in the literature. The varying vigilance plays an essential role in the supervised ARTMAP networks [CGR91a]. Most closely related to the ART-C learning paradigm may be the HART modular designs (HART-J and HART-S) [BW00]. HART generates hierarchical representation of the input sequence. Learning activities in various layers of the hierarchy are guarded with different ρ values and produce category representation of the same input sequence with varying details, either from fine to coarse representation (HART-J) or from coarse to fine representation (HART-S). HART however lacks the capability of producing a predefined number of categories in any layer, as the modular design presets a vigilance value for each layer and limits the learning activities in each layer strictly like a conventional ART.

One key idea employed in ART-C is the redistribution of the representation categories during constraint reset through merging of categories. A number of hierarchical agglomerative clustering algorithms (Section 2.3.2) such as UPGMA [MS57] and neighbor-joining [SN87] apply a similar paradigm. Hierarchical agglomerative clustering algorithms typically represent M input samples as M reference clusters. Each clustering cycle identifies the most similar pair of clusters and merges them.

The process may repeat until there is only one cluster left. While they are able to generate a predefined number of C clusters over M input samples, they are notably computational intensive in maintaining the pair-wise similarity matrix as typically $M \gg C$ [Fri97, FK99, PR02]. The advantage of ART-C over this class of algorithms lies in its combination of competitive learning of individual inputs with the calculation of pair-wise category similarities. While online learning of individual inputs maintains a high efficiency, merging of the nearest category pair in ART-C enables a quick redistribution of recognition categories, with a notably lower computational cost.

4.3.6 Selection of ART and ART-C for a Specific Problem

The unique characteristics of ART is its capability in self-adapting its structure in response to the arbitrary input sequence through dynamic creation of recognition categories for encoding distinct input samples. Compared with SOM which requires a scan of at least a major portion of the input set in order to initialize its weight vectors, ART does not requires such a process and hence is more capable of carrying out online learning of incrementally presented inputs. However, the major deficiency of ART is its dependence on a proper estimated vigilance value.

ART-C's learning is not dependent to a fixed vigilance parameter. Instead, the constraint parameter C mainly affects ART-C's learning. When the constraint parameter C is set to a very large value, ART-C could learn identically as ART. On the other hand, when C is set to an appropriate small value, ART-C is capable of generate exactly C output clusters as desired.

In a preliminary study previously reported in [HTT02], Fuzzy ART-C and Fuzzy ART have shown reasonably similar output groupings on the same input sequence, working under different desired number of categories (Figure 4.6 and Figure 4.7). The case studies reported in a Chapter 6 will also show that the performance of ART-C 2A and ART 2A are very close, in terms of the output cluster validity as well as the efficiency.

These similarities and differences present an interesting consideration: given a specific clustering problem, which algorithm will be more suitable, ART or ART-C?

Readers may already note that in the machine learning domain, the clustering problem is ill-posed, as there does not exist "an absolute judgment as to the relative efficacy of all clustering techniques" [BA02a, BA02b]. As such, in our consideration, the usability of a clustering algorithm is more of importance than the efficacy of the algorithm.

From the end-user's point of view, like many other clustering algorithms, both ART and ART-C require some degree of prior knowledge in estimating the problem domain and setting the proper parameters. ART's learning is most dependent on the vigilance parameter. Suggesting such a parameter requires prior knowledge on the *relative pattern proximity* of the inputs. That is, how the input data are distributed? How similar would be two data points that falls into a natural cluster? While for ART-C, its learning is dependent on the constraint parameter. Suggesting such a parameter requires prior estimation on the *number of natural groupings* of the inputs. Hence, rather than stating either ART or ART-C "a better solution" than the other, we should instead consider ART and ART-C as the complementary solution to each other, depending on the availability of the prior knowledge on the

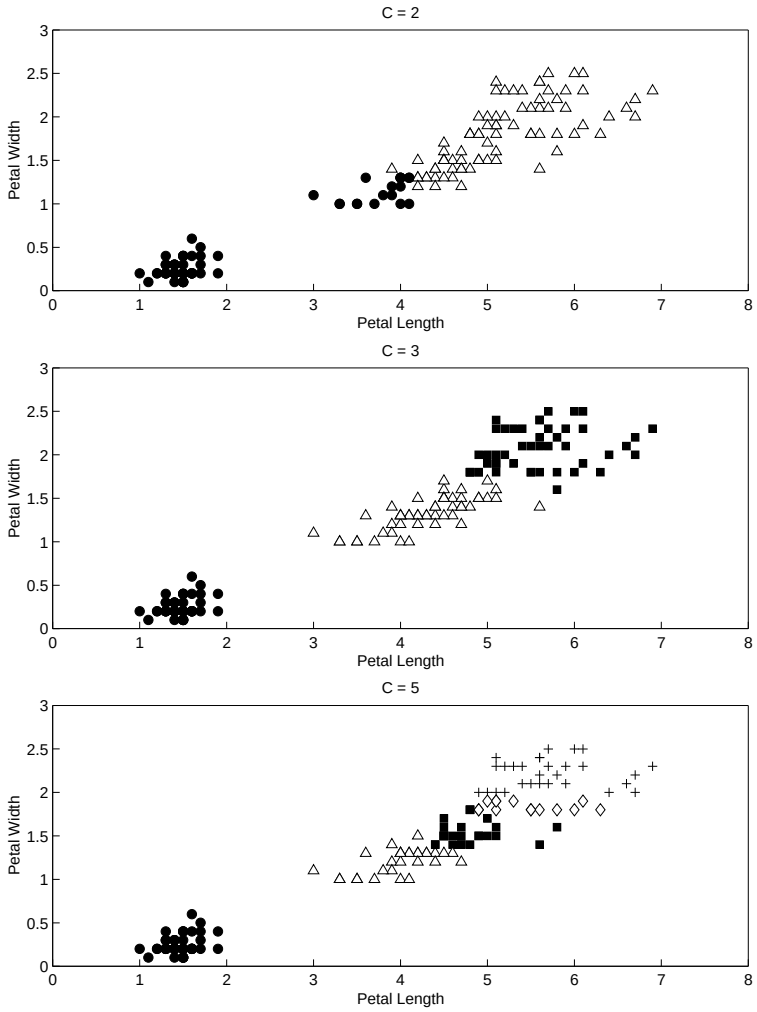


Figure 4.6: The outputs of Fuzzy ART-C on the Iris data set when we set $C = 2$, 3, and 5 respectively. The network converged at $\rho = 0.5958$, 0.7187, and 0.7542 correspondingly. Data points being grouped in the same cluster are identified with the same markers. The outputs are reasonably similar to those of Fuzzy ART illustrated in Figure 4.7.

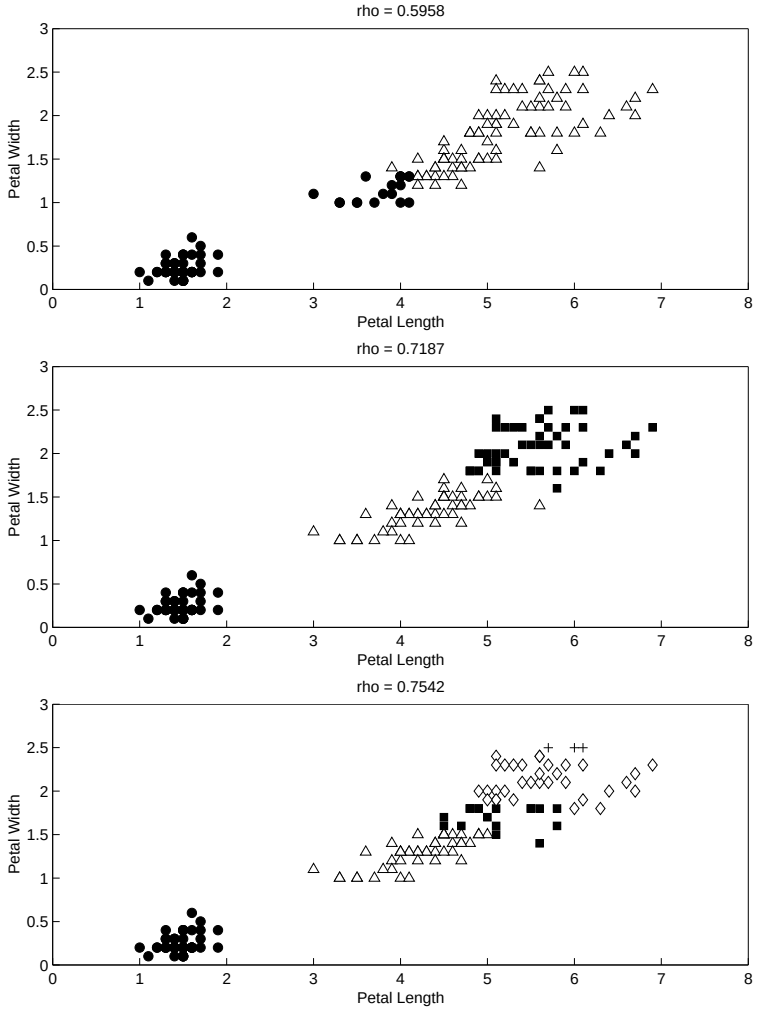


Figure 4.7: The outputs of Fuzzy ART on the Iris data set when we set $\rho =$ 0.5958, 0.7187, and 0.7542 respectively. The network outputted 2, 3, and 5 clusters correspondingly. Data points being grouped in the same cluster are identified with the same markers. The outputs are reasonably similar to those of Fuzzy ART-C illustrated in Figure 4.6.

input sequence. Table 4.1 summarizes a general guideline on the selection of ART and ART-C for a specific problem.

Table 4.1: A general guideline on the selection of ART and ART-C for a specific problem.

	ART	ART-C
Learning characteristics:	Control the intra-variance of each cluster. Increase the number of clusters to encode distinct input.	Control the number of clusters. Increase the intra-variance of each cluster to encode distinct input.
Learning most affected by:	Vigilance ρ.	Constraint C.
Optimal parameter setting requires prior knowledge on:	Local distribution (intra-variance) of at least one cluster.	Global estimation of optimal number of clusters.
When lacking of prior knowledge, typical paradigm to obtain the optimal parameter setting:	Trial-and-error on multiple vigilance values	Trial-and-error on multiple constraint values

CHAPTER 5
QUANTITATIVE EVALUATION OF CLUSTER VALIDITY

5.1 Problem Specification

One of the most important problems in cluster analysis is the evaluation of the clustering results. The major concern on the clustering results is how well the results fit the natural distribution of the data, i.e. the underlying grouping. This is the main subject of *cluster validity* [HBV01].

Due to the nature of the ill-posed clustering problem, a clustering algorithm may perform drastically differently depending on the following factors:

1. The feature of the input data set, e.g. dimension, scale, distribution, etc.

2. The representation of the pattern features.

3. The measure used to evaluate pattern proximity.

4. The parameter used by the algorithm.

As such, evaluation of the clustering results provides important guidance to studies on various studies (Figure 2.2), including selecting the appropriate pattern representation method and pattern proximity measure, choosing the optimal parameter set for an algorithm as well as choosing the optimal clustering algorithm that best fit the feature of the input data.

In a large number of previous studies, low-dimensional (usually 2D and 3D) synthetic and real-life data are commonly used as the inputs. This is for the convenience of human inspection on the clustering results via graphical visualization. However, the performance of a clustering algorithm may vary drastically depending on the feature of the input data. For instance, Grid-based clustering algorithms (Section 2.3.4) may perform well on 2D data sets but are generally not known as capable of handling high dimensional inputs. Visualization on 2D or 3D data sets only thus lacks the capability of inspecting the algorithms' general performance on high dimensional data. Moreover, most real-life data sets present a challenge for visualization due to the large scale. As such, quantitative assessment of the cluster validity is of great importance for cluster analysis.

Theodoridis and Koutroubas [TK99] summarized three major approaches for evaluation of clustering results, listed below:

1. **External criteria:** This assumes there is a known distribution of the input data that reflects the so-called optimal grouping or natural grouping over the

input. The clustering results are compared with this external information and the match between the two sets of results are evaluated.

2. **Relative criteria:** With the lack of a known input distribution, this compares the clustering result of an algorithm with those of other clustering algorithms or those of the same algorithm with different sets of parameters.

3. **Internal criteria:** With internal criteria, the evaluation is solely based on the statistics on the output result of a clustering algorithm, such as the proximity matrix and the output cluster centers etc.

Among the above, the thesis study focuses on the relatively more feasible criteria, i.e., external and relative criteria.

Since the nature of clustering is to group the input samples such that data points in the same cluster are more similar to each other than to points in a different cluster (Section 2.1), a large number of evaluation paradigms in the literature evaluate the cluster validity through two intuitive aspects, namely *intra-cluster compactness* and *inter-cluster separation*. Generally speaking, in an optimal clustering solution, the members in each cluster shall be as close as possible, while the clusters should be clearly separated to each other. Based on these two principles, the following sections introduce two sets of evaluation measures using internal criteria and external criteria respectively.

5.2 Cluster Validity Measures Based on Cluster Distribution

The set of measures introduced here evaluate the intra cluster compactness and inter cluster separation using intuitive statistics based on the internal distribution of the output clusters.

5.2.1 Cluster compactness

Suppose there exists a known *distance metric* that evaluates the *dissimilarity* of patterns, which also corresponds to the pattern proximity measure used by the clustering algorithm. The cluster compactness measure is based on the generalized definition of the *deviation* of a data set $\mathbf{X}$ given by

$$dev(\mathbf{X}) \equiv \sqrt{\frac{1}{M}\sum_{i=1}^{M} d^2(\mathbf{x}_i, \overline{\mathbf{x}})} \tag{5.1}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the distance metric between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, M is the number of vectors in $\mathbf{X}$ and $\overline{\mathbf{x}} = \frac{1}{M}\sum_i \mathbf{x}_i$ is the mean of $\mathbf{X}$. A smaller *deviation* indicates a higher homogeneity of the vectors in the data set, in terms of the distance measure $d()$. In particular, when $\mathbf{X}$ is one-dimensional and $d()$ is the Euclidean distance, $dev(\mathbf{X})$ becomes the statistical variance of the data set $\sigma(\mathbf{X})$. The cluster compactness for the output clusters $\mathbf{C} = \{\mathbf{C}_i, i = 1, \ldots, K\}$ generated by a clustering algorithm over the input set $\mathbf{X}$ is then defined as

$$Cmp = \frac{1}{K}\sum_i \frac{dev(\mathbf{C}_i)}{dev(\mathbf{X})} \tag{5.2}$$

where K is the number of clusters generated on the data set $\mathbf{X}$, $dev(\mathbf{C}_i)$ is the variance of the cluster $\mathbf{C}_i$ and $dev(\mathbf{X})$ is the variance of the data set $\mathbf{X}$.

The cluster compactness measure evaluates how well the subsets (output clusters) of the input is redistributed by the clustering system, compared with the whole input set, in terms of the data homogeneity. When the distance metric is the Euclidean distance, the cluster compactness measure is equivalent to the *average cluster scattering* index used in Halkidi et al.'s study [HVB00]. While a smaller *Cmp* value indicates a higher average compactness in the output clusters, this however does not necessarily mean a "better" clustering solution. Given a clustering system that encodes each and every unique input data into one separate cluster, the *cluster compactness* score of its output has a minimal value of 0. Such a clustering output is however not desirable. To tackle this, the *cluster separation* measure is used to complement the evaluation.

5.2.2 Cluster separation

In the literature, there exist a variety of measures that evaluate the separation of clusters through difference approaches. These are *single linkage* which evaluates through the closest members of different clusters, *complete linkage* which evaluates through the most distant members of different clusters and *centroid comparison* which evaluates through the centroids of different clusters. In our study, we adopt the centroid comparison approach due to the relatively low computational cost on search.

The cluster separation measure used here borrows the idea in [HBV01] and the cluster evaluation function introduced by [GP00]. The cluster separation of a

clustering system's output is defined by

$$Sep = \frac{1}{K(K-1)} \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} \exp(-\frac{d^2(\mathbf{c}_i, \mathbf{c}_j)}{2\sigma^2}) \tag{5.3}$$

where σ is a Gaussian constant, K is the number of clusters, $\mathbf{c}_i$ is the centroid of the cluster $\mathbf{C}_i$ and $d(\mathbf{c}_i, \mathbf{c}_j)$ is the distance between the centroid of $\mathbf{C}_i$ and the centroid of $\mathbf{C}_j$.

Compared with the method used in [HBV01], the evaluation measure above adopts Gaussian normalization to limit $Sep \in (0, 1]$. A smaller Sep score indicates a larger overall dissimilarity among the output clusters. However, given the particular case that a clustering system output the whole input set into one cluster, the Sep score reaches an indicative minimal value of 0, which is of no applicable value.

Noting the limitation of each evaluation measures above, it is necessary to combine the cluster compactness and cluster separation measures to evaluate the overall performance of a clustering system. An intuitive combination, named *overall cluster quality*, is defined as

$$Ocq(\beta) = \beta \cdot Cmp + (1 - \beta) \cdot Sep \tag{5.4}$$

where $\beta \in [0, 1]$ is the weight that balances cluster compactness and cluster separation. For example, $Ocq(0.5)$ gives equal weights to the two measures.

5.3 Cluster Validity Measures Based on Class Conformity

This category of validity measures assumes that there is a desirable distribution of the data set with which it is possible to perform a direct comparison of the

clustering output. Following the data distribution, one can assign a class label to each data point. The target of the clustering system can then be correspondingly interpreted as to replicate the underlying class structure through unsupervised learning. In an optimal clustering output, data points with the same class labels are clustered into the same cluster and data points with different class labels appear in different clusters.

To evaluate the degree of the conformity of the output clusters with respect to the predefined classes, we applied the *entropy* concept. In various science domains, entropy is widely used to evaluate the disorder or randomness in a closed system. The two validity measures based on class conformity are summarized below.

5.3.1 Cluster entropy

Boley [Bol98] introduced an information entropy approach to evaluate the quality of a set of clusters according to the original class labels of the data points . For each cluster $\mathbf{C}_i$, a cluster entropy Enc_i is computed by

$$Enc_i = -\sum_j \frac{n(l_j, c_i)}{n(c_i)} log \frac{n(l_j, c_i)}{n(c_i)} \tag{5.5}$$

where $n(l_j, c_i)$ is the number of the samples in cluster $\mathbf{C}_i$ with a predefined label l_j and $n(c_i) = \sum_j n(l_j, c_i)$ is the number of samples in cluster $\mathbf{C}_i$.

Understandably, if an optimal output cluster contains samples that come from the same class, the cluster entropy of this cluster has a minimal value of 0. In addition, since the external criteria, i.e. predefined classes, is assumed to be the optimal clustering solution, that is, to have optimal intra-class compactness and

inter-class separation, the cluster entropy measure thus reflects the intra-cluster compactness of the corresponding cluster.

To obtain an global evaluation of the whole clustering solution, the overall *cluster entropy Enc* is given by a weighted sum of individual cluster entropies by

$$Enc = \frac{1}{\sum_i n(c_i)} \sum_i n(c_i) Enc_i. \tag{5.6}$$

The cluster entropy reflects the quality of individual clusters in terms of homogeneity of the data points in a cluster. It however does not measure the compactness of a clustering solution in terms of the number of clusters generated. A clustering system that generates many clusters would tend to have very low cluster entropies but is not necessarily desirable. To counter this deficiency, we use another entropy measure below to measure how data points of the same class are represented by the various clusters created.

5.3.2 Class entropy

For each class $\mathbf{L}_j$, a class entropy Enl_j is computed by

$$Enl_j = -\sum_i \frac{n(l_j, c_i)}{n(l_j)} log \frac{n(l_j, c_i)}{n(l_j)} \tag{5.7}$$

where $n(l_j, c_i)$ is the number of samples in cluster $\mathbf{C}_i$ with a predefined label l_j and $n(l_j) = \sum_i n(l_j, c_i)$ is the number of the samples with class label l_j. The overall *class entropy Enl* is then given by a weighted sum of individual class entropies by

$$Enl = \frac{1}{\sum_j n(l_j)} \sum_j n(l_j) Enl_j. \tag{5.8}$$

Similar to above, it is understandable that if members in a predefined class are grouped in the same output cluster, the corresponding class entropy reaches a minimal value of 0. Correspondingly, class entropy reflects the separation of output clusters. In a particular case that a clustering algorithm generates one output cluster only, the class entropy has a indicative minimal value of 0, but is not desirable. Hence, we follow the identical paradigm for the combination of cluster compactness and cluster separation and define a combined *overall entropy* measure to facilitate our comparison:

$$Ens(\beta) = \beta \cdot Enc + (1 - \beta) \cdot Enl \tag{5.9}$$

where $\beta \in [0, 1]$ is the weight that balances the two measures.

5.4 Efficacy of the Cluster Validity Measures

To study the efficacy of the cluster validity measures, we carried out a number of controlled experiments on several synthetic data sets [HTTS03]. Tasks on these data sets include identifying the optimal number of clusters in the data set and choosing of a pattern proximity measure suitable for the specific data distribution. Batch K-Means (Section 2.3.1) is used as the clustering algorithm in the experiments.

Since the clustering output of K-Means can be affected by the initialization of cluster prototypes, we adopted the commonly used statistical validation paradigm in our experiments. Given a clustering task, we repeat the experiments for ten times. In each experiment, the presenting sequence of the input data is reshuffled and K-Means is trained to convergence. Based on the observation values from the

ten runs, the *means* and the *standard deviations* are reported. *t-test* is used to validate the significance of the comparative observations across the ten runs.

5.4.1 Identification of the Optimal Number of Clusters

Our experiments on the synthetic data set as shown in Figure 5.1 evaluated the K-Means clustering method using the Euclidean distance. The synthetic data set contains 334 data points, each is a two-dimensional vector of values between 0 and 1. Our task is to identify the optimal number of clusters on the data set in an unsupervised way. Here the *optimal* solution refers to the result that best reflects the data distribution and matches the user's post-validation on the data set, in terms of intra-cluster compactness and inter-cluster separation of the output clusters.

The paradigm of the experiment is summarized below. We apply the K-Means method on the data set using K values ranging from 2 to 8. For each K value, we evaluate the quality of the output using the measures based on *cluster compactness* (Cmp) and *cluster separation* (Sep). This enables us to observe the change of the score values according to the change of K. Intuitively the most satisfactory quality score indicates the best partition of the data set, while the corresponding K value suggests the optimal number of clusters on the data set.

Figure 5.2 depicts the change of *cluster compactness*, *cluster separation* as well as *overall cluster quality* by varying K from 2 to 8. $2\sigma^2 = 0.25$ is used for the ease of evaluation in Equation 5.3 and $\beta = 0.5$ for $Ocq()$ is used to give equal weights to *cluster compactness* and *cluster separation*. On each parameter setting,

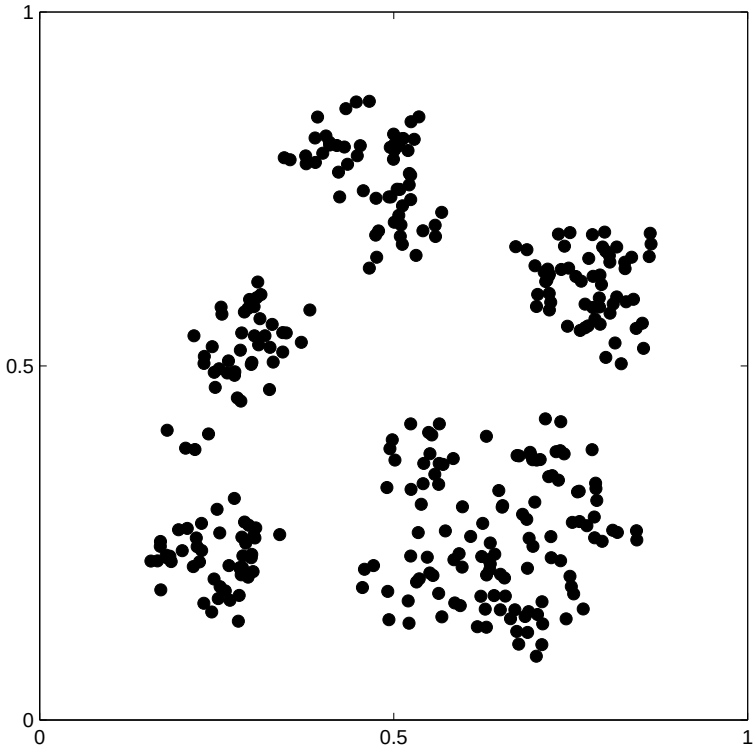


Figure 5.1: A synthetic data set used in the experiments.

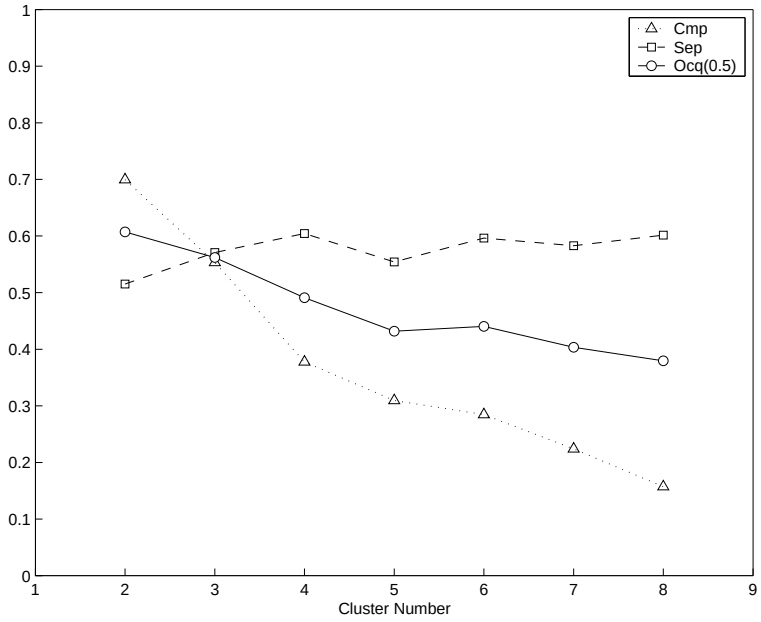


Figure 5.2: *Cluster compactness*, *cluster separation*, and *overall cluster quality* of K-Means on the synthetic data set in Figure 5.1. The locally minimal value of *overall cluster quality* $Ocq(0.5)$ at $K = 5$ suggests the optimal number of clusters on the data set.

the experiments are repeated for ten times, each using a different randomization of initial cluster seeds. To obtain a clear illustration, only the mean values of the ten observations over each measure are plotted in the figure while the standard deviations are not reported.

It is noted that, when K increases, the *cluster compactness* score gradually decreases and the *cluster separation* score generally increases. This is due to the nature that a larger number of partitions on the same data space generally tends to decrease the size of each partition (which causes higher compactness in each partition) as well as the distances among the partition centroids (which causes lower separation of partitions). However, as an notable exception, the *cluster*

separation shows a locally minimal value at $K = 5$, while the decreasing trend of *cluster separation* at $K = 5$ is significantly different from those at different K values. The *overall cluster quality* also shows a locally minimal value at $K = 5$. This suggests that the optimal number of clusters (in terms of Euclidean similarity) is five. The result is supported by human inspection of the data in Figure 5.1.

The drawbacks of this experimental paradigm however are notable. First, it is not easy to suggest a proper range of K values for the iterative testing if the user lacks a prior estimation of the data distribution. In addition, both the σ value for the calculation of *cluster separation* and the weight β for the calculation of *overall cluster quality* are subjectively determined. This shows that human interaction and prior knowledge on the problem domain is still required for the use of these evaluation measures.

5.4.2 Selection of Pattern Proximity Measure

Our experiments on the synthetic data set as shown in Figure 2.5 utilize the quality measures based on class conformation to evaluate two variations of ART-C networks, namely fuzzy ART-C (based on Fuzzy ART) and ART-C 2A (based on ART-2). While fuzzy ART-C groups input data according to *nearest hyper-rectangles*, ART-C 2A groups input data according to *nearest neighbors* in terms of cosine similarity. Our comparative experiments attempt to discover which of these two pattern proximity measures is more capable of identifying the data distribution on the problem domain and therefore produces clustering output with a higher match with the user intuition.

In order to evaluate the clustering quality using *cluster entropy*, *class entropy*, and *overall entropy*, we pre-assigned each data point with a class label based on our observation. There are four different class labels assigned to the 250 data points in the collection, each label corresponding to a string-shaped class in Figure 2.5.

Our experiments compared fuzzy ART-C and ART-C 2A with a preset constraint (C) of 4, each using a standard set of parameter values. Table 5.1 summarizes the statistics of the comparison results. While ART-C 2A is capable of producing a better balanced set of *cluster entropy* and *class entropy* scores (which indicates a better balance of cluster homogeneity and class compactness), the *cluster entropy* score of fuzzy ART-C is three times higher than that of ART-C 2A in our experiment. Although the *class entropy* score of fuzzy ART-C is slightly lower than that of ART-C 2A, the weighted *overall entropy* $Ens(0.5)$ of fuzzy ART-C is significantly higher than that of ART-C 2A due to the high *cluster entropy* value. This indicates that the cosine similarity based paradigm is more suitable than the nearest hyper-rectangle based paradigm on the tested problem domain. This result is not surprising to us, as prior comparison studies on hyper-rectangle methods also showed that they perform well only when the data boundaries are roughly parallel to the coordinates axes [WD95].

These preliminary experiments show that both the cluster distribution and class conformity based cluster validity measures proposed in this chapter are capable of detecting the differences between two sets of clustering results. The comparison results with these evaluation measures are valuable for the studies of various cluster analysis approaches.

Table 5.1: *Cluster entropy*, *class entropy*, and *overall entropy* of ART-C 2A and fuzzy ART-C on the synthetic data set in Figure 2.5. Both methods work on $C = 4$. All values are shown with the means and the standard deviations over ten runs, each using a different randomization of input order.

Method	Enc	Enl	$Ens(0.5)$
ART-C 2A	0.1483 ± 0.0173	0.1142 ± 0.0225	0.1312 ± 0.0092
fuzzy ART-C	0.5800 ± 0.0037	0.0337 ± 0.0064	0.3068 ± 0.0020

CHAPTER 6

CASE STUDIES ON REAL-LIFE PROBLEMS

6.1 The Gene Expressions

The gene expression is one of the major research topics in the bioinformatics domain. Gene expressions quantify the expression levels of individual genes, reflected by the amount of mRNA or protein products produced by the cell. Traditional methods in molecular biology generally work on a "one gene in one experiment" basis. In the recent years, a new technology named DNA microarray (also termed biochip, DNA chip and gene array in the literature) raised great interests among biologists. This technology promises to monitor the whole genome on a single chip so that researchers can have a better picture of the interactions among thousands of genes simultaneously.

An array is an orderly arrangement of samples to minimize the risk of errors during experiments. DNA microarray are fabricated by high-speed robotics that allow massively parallel gene expression and gene discovery studies. An experiment with a single DNA chip can provide researchers information on thousands of genes simultaneously (Figure 6.1). The primary goal of microarray experiments is to generate expression information for every gene in the array, under some set of conditions. Figure 6.2 illustrates the work flow of a typical microarray experiment [Kel99]. Expression may be studied in

- Different tissues;
- Different developmental stages;
- Different genotypes;
- Different treatments;
- Different times after a treatment.

The main technologies for producing DNA microarrays include the the cDNA microarray [EB99], Affymatrix GeneChip [LGL99] and SAGE methods [VZVK95]. The first method measures relative levels of mRNA abundance between different samples, while the last two measure absolute levels [LGG01]. Regardless of the experimental method, the microarray data obtained in an experiment is a set of multidimensional vectors (arrays), each vector corresponding to a gene, whereas each element of the vector corresponding to a different condition.

There are two major application forms for the DNA microarray technology:

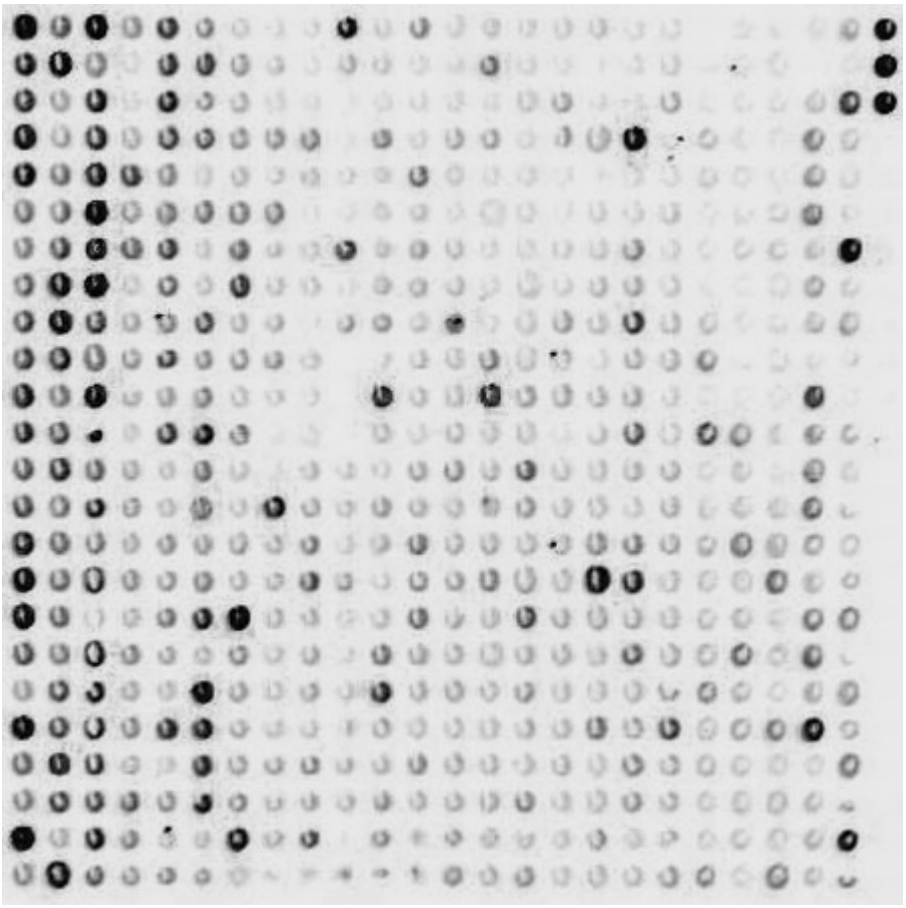


Figure 6.1: The image of a DNA chip as published at http://www.gene-chips.com/sample1.html. The intensity and color of each spot encode information on a specific gene from the tested sample. In the original image, the background is dark and the spots are in different light colors. The image colors are reversed here for a better print effect on gray-scale printers.

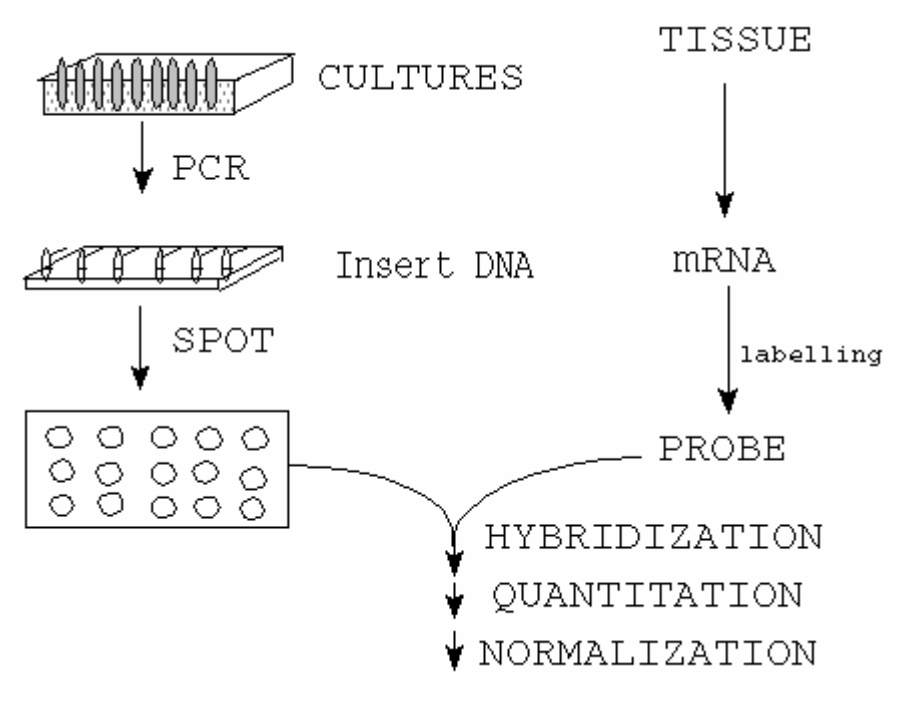


Figure 6.2: The work flow of a typical microarray experiment.

1. Identification of sequence (gene / gene mutation).

2. Determination of expression level (abundance) of genes.

The principle for sequence identification is base-pairing (i.e., A-T and G-C for DNA; A-U and G-C for RNA) or hybridization. The microarray provides a medium for matching known and unknown DNA samples based on base-pairing rules and automating the process of identifying the unknowns. More studies on the gene expressions focus on the clustering of genes similarities in terms of their expression profiles. This is used to determine the proteins that are expressed together under different cellular conditions [BK00, SEWB00, THE$^+$99]. The knowledge discovered by the clustering process is of great value for various molecular biological processes, such as correlating expression patterns as well as mapping expressions data to se-

quence, structural and biochemical data. The applications of these studies include disease diagnosis, drug discovery and toxicological research, etc.

The clustering algorithms being applied to analyze gene expression data include K-Means [MCA+98], hierarchical clustering [ESBB98], graph theory based clustering [BDSY99], naive Bayesian clustering [BF01] and Gaussian mixture model based clustering [YFRR01]. However, there are rare studies of neural networks on gene clustering besides a few applications of the self-organizing map (SOM) [HVD01, TSM+99].

6.1.1 The Rat CNS Data Set

The rat CNS data is generated by Wen et al. [WFM+98] for the study of the nature of the complex self-organizing processing underlying mammalian central nervous system (CNS) development. It contains the expressions of 112 rat genes during CNS development using the RT-PCR protocol [sWMB95]. Each array contains expressions on nine time points, covering the embryonic development phase (days 11 through 21, namely E11, E13, E15, E18 and E21), the postnatal development phase (days 0 through 14, namely P0, P7 and P14) and the adult phase (P90, or A). Prior study by Wen et al. on the data set using the FITCH software, which is based on K-Means, summarized five major waves of the gene expression patterns (Figure 6.3). With the exception of the *Constant* wave, they have shown high correlation with the four major functional categories identified using biological domain knowledge, namely *Neuroglial Markers*, *Neurotransmitter Receptors*, *Peptide Signaling* and *Diverse*.

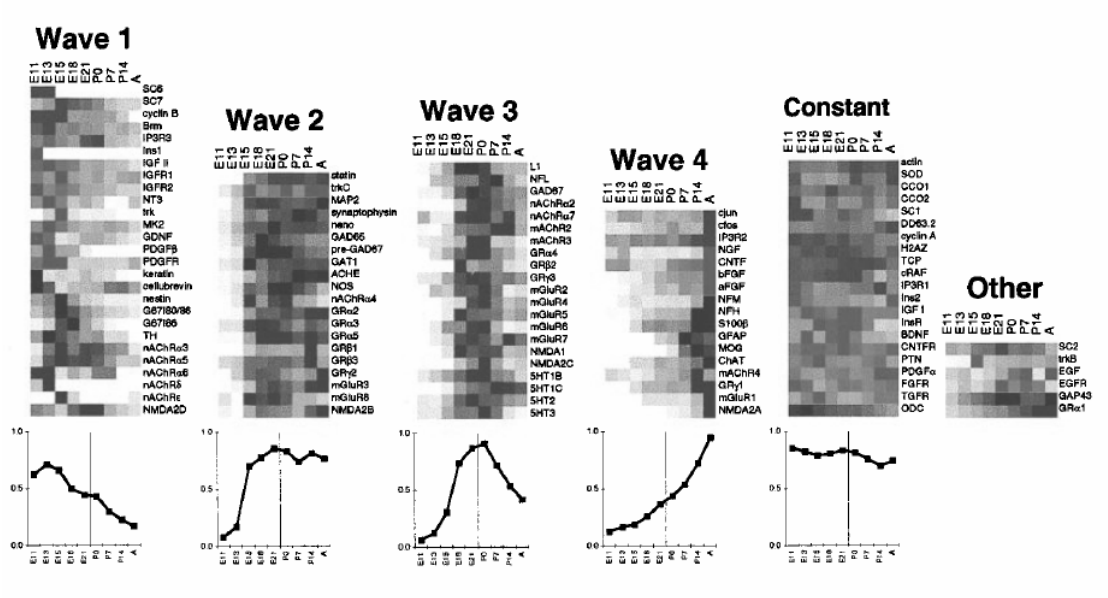


Figure 6.3: The gene expression patterns of the rat CNS data set discovered by Wen et al. The x-axis marks the different time points. The y-axis indicates the gene expression levels.

The objective of our experiments on the rat CNS data set is to evaluate the clustering efficacy of ART-C algorithm. We applied ART-C 2A to the clustering of the microarrays and compared the output of ART-C 2A with that of the FITCH software as reported in Wen's study. ART-C 2A is adopted due to the cosine similarity it used for evaluation of pattern proximity. The cosine similarity is most closely related to the K-Means algorithm adopted by the FITCH software. The data set on the public domain have been well selected and well normalized and hence is directly used without any preprocessing. The small size and the relatively distinct expression patterns of this data set enabled us to validate ART-C 2A's clustering results via visual inspection.

We adopted a standard set of parameters for ART-C 2A. The learning rate was

initialized with 0.05. We applied a simple threshold-linear function for the learning rate fading such that the learning rate $\eta^{(t+1)} = 0.9\eta^{(t)}$ if the network's recognition accuracy reached a threshold of 0.8. The algorithm was said to reach convergence if the cluster assignment for the input samples does not show a relative change of 0.5%. While Wen et al.'s study discovered five major "waves", five was not necessarily considered as the "optimal number of clusters" in the previous study [WFM$^+$98]. We tried different constraint (C) settings for ART-C 2A, and $C = 12$ was found to generate the most satisfactory results.

Figure 6.4 depicts the mean expression pattern of each cluster generated by ART-C 2A. Error bars corresponding to the deviations to the mean expressions are not plotted for a clearer illustration. The gene expressions grouped in each cluster are observed to have close similarity to each other. Each pattern has showed a distinct group of gene expressions, identified by the variances of the expressions across all time points and the time point corresponding to the peak level. The ART-C 2A output is observed to have close relevancy with the output of the FITCH software as reported by Wen et al. The mapping of the clusters generated by ART-C 2A to the five major waves discovered by FITCH is summarized in Table 6.1.

To further validate the ART-C 2A clusters, we investigated the correlation of genes in each cluster (Table 6.2) with the major gene functional categories previously identified through human inspection (Table 6.3) [WFM$^+$98]. With the exception of cluster 1, which encodes a max number of genes in *Peptide Signaling* and *Diverse* categories with relatively constant expressions (constant expressions normally are not of interests to biologists), the majority of the most clusters are

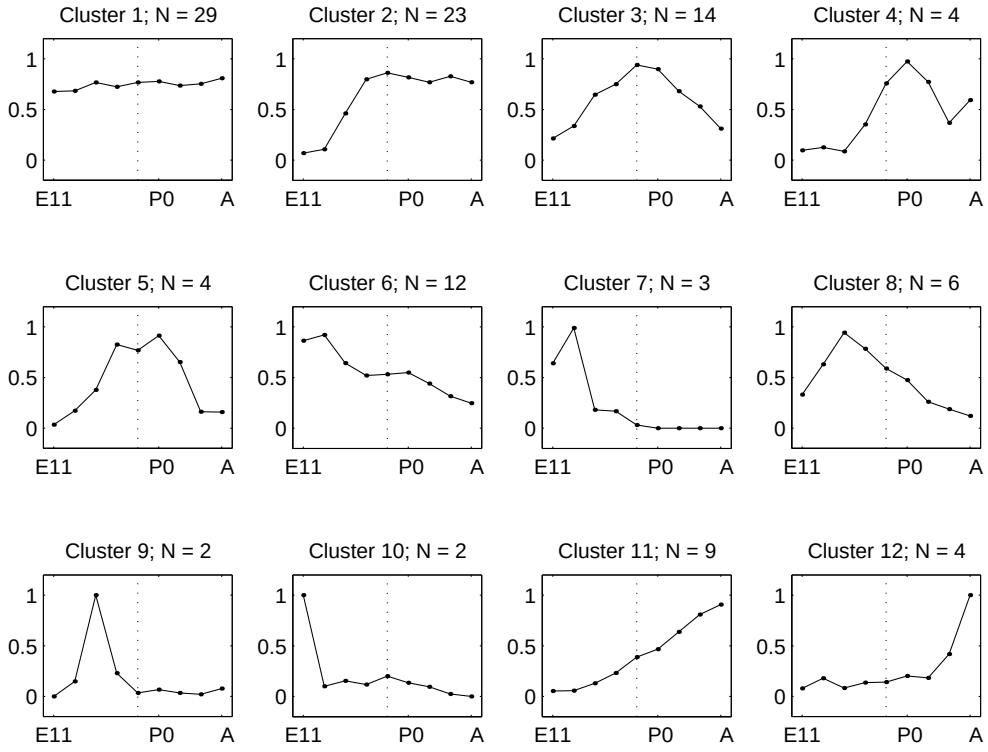


Figure 6.4: The gene expression patterns of the rat CNS data set generated by ART-C 2A. The X-axis indicates different time points with different labels. N indicates the number of genes being clustered in each cluster.

Table 6.1: Mapping of the gene patterns generated by ART-C 2A to the patterns discovered by FITCH. NA and NF indicate the number of gene expressions being clustered in ART-C 2A's and FITCH's grouping respectively. NC indicates the number of common gene expressions that appear in both ART-C 2A's and FITCH's grouping.

Cluster Pattern					
ART-C 2A	NA	FITCH	NF	NC	Interpretation
Clusters 6 - 10	25	Wave 1	27	19	Peak level during early E phase.
Cluster 2	23	Wave 2	20	14	Ascending levels during E phase and relatively constant levels during P and A phases.
Clusters 3 - 5	22	Wave 3	21	13	Peak level during late E and early P phases.
Clusters 11 - 12	13	Wave 4	17	12	Ascending levels across all time points.
Cluster 1	29	Constant Wave	21	18	Relatively constant levels across all time points.

dominated by genes of a single functional category. The best result is given by clusters 2 - 5, which clearly recognize the functional group *Neurotransmitter Receptors*. It is noted that, although cluster 6 and cluster 8 show relatively similar patterns (grouped into Wave 1 in Wen et al's study), the majority of the genes in these clusters actually corresponded to two different gene functional groups *Peptide Signaling* and *Neuroglial Markers*. This shows that ART-C 2A is capable of identifying subtle differences between the sets of two patterns.

It is interesting that although clusters 8 and 11 present very different patterns, they actually corresponded to the same functional group *Neuroglial Markers*. In addition, several small clusters, especially clusters 7, 9 and 10 clearly identify a number of noises who did not follow the correlation between their functions and expressions in the main stream. This reflected the underlying complexity of the gene expression data.

Table 6.2: The list of genes grouped in the clusters generated by ART-C 2A.

Cluster	Genes in the cluster
1	GAP43, GAT1, ODC, GRa1, GRb3, BDNF, CNTF, trkB, trkC, CNTFR, PTN, PDGFa, FGFR, TGFR, Ins2, IGF I, IGFR1, CRAF, IP3R1, IP3R2, cyclin A, H2AZ, cjun, TCP, actin, DD63.2, SOD, CCO1, CCO2
2	MAP2, synaptophysin, neno, S100, pre-GAD67, GAD67, ACHE, GRa2, GRa3, GRa5, GRb1, GRg2, GRg3, mGluR3, mGluR5, mGluR7, NMDA1, NMDA2B, nAChRa7, mAChR2, 5HT1c, 5HT2, statin
3	L1, NFL, GAD65, NOS, GRa4, mGluR8, NMDA2D, nAChRa3, nAChRa4, mAChR3, 5HT1b, EGFR, InsR, SC2
4	mGluR4, NMDA2C, nAChRa2, EGF
5	GRb2, mGluR2, mGluR6, 5HT3
6	cellubrevin, nAChRa6, NT3, MK2, GDNF, PDGFR, IGF II, IGFR2, IP3R3, cyclin B, Brm, SC1
7	nAChRd, PDGFb, SC6
8	nestin, G67I80/86, G67I86, TH, nAChRa5, SC7
9	nAChRe, trk
10	keratin, Ins1
11	NFH, GFAP, MOG, ChAT, GRg1, mAChR4, bFGF, aFGF, cfos
12	NFM, mGluR1, NMDA2A, NGF

Table 6.3: The correlation between the gene clusters discovered by ART-C 2A and the functional gene categories identified through human inspection. N is the number of genes in each cluster. NM, NR, PS and DV are the numbers of genes in functional category *Neuroglial Markers*, *Neurotransmitter Receptors*, *Peptide Signaling* and *Diverse* respectively. Boldface numbers identify the dominant functional category in each cluster.

Cluster	N	NM	NR	PS	DV
1	29	3	2	**12**	**12**
2	23	7	**15**	0	1
3	14	4	**7**	2	1
4	4	0	**3**	1	0
5	4	0	**4**	0	0
6	12	1	1	**6**	4
7	3	0	1	1	1
8	6	**4**	1	0	1
9	2	0	1	1	0
10	2	1	0	1	0
11	9	**4**	2	2	1
12	4	1	**2**	1	0
Total	112	25	39	27	21

6.1.2 The Yeast Cell Cycle Data Set and The Human Hematopoietic Data Set

To understand the characteristics of different learning algorithms in a quantitative way, our experiments compared the performance of ART-C 2A, ART, SOM, online K-Means and batch K-Means on two gene expression data sets, namely the yeast cell cycle data set and the human hematopoietic differentiation data set. ART-C 2A, ART 2A, SOM and online K-Means are compared side by side as all they are based on incrementally competitive learning and could generate a single hard partition of the input space for the quantitative comparison. Batch K-Means is also considered in the comparable experiments are it is closely related to online K-Means.

Data Pre-processing

The yeast cell cycle data set (YEAST)[1] of Cho et al. [CCW+98] consists of 6,601 genes expressions, each with 17 conditions. The 17 conditions are evenly divided into two panels, which correspond to two cell cycles, with the 9th condition as the intermediate condition between the two cell cycles.

The human hematopoietic data set (HL60_U937_NB4_Jurkat)[2] of Tamayo et al. [TSM+99] combines expression data from four different cell lines: HL-60 and U937, two *myeloid cell* lines which undergo macrophage differentiation in response to TPA; NB4, an *acute promyelocytic leukemia* cell line that undergoes neutrophilic

[1]The YEAST data set is available via http://sgdlite.princeton.edu/download/yeast_datasets/.

[2]The HL60_U937_NB4_Jurkat data set is available via http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi.

differentiation in response to all-trans retinoic acid (ATRA) and *Jurkat*, a T-cell line that acquires many hallmarks of T-cell activation in response to TPA. The data set contains a total of 17 conditions: 4 time points for HL60 (0, 0.5, 4 and 24 hours), 4 time points for U937 (0, 0.5, 4 and 24 hours), 5 time points for NB4 (0, 5.5, 24, 48 and 72 hours) and 4 time points for Jurkat (0, 0.5, 4 and 24 hours). There are a total of 6,416 gene expressions in the data set.

Both YEAST and HL60_U937_NB4_Jurkat data sets come in raw format and hence require appropriate preprocessing before they are used for the clustering study.

It is a common understanding by the biologists that the gene expressions with large variance across different conditions are of more study interests than constant ones. On the other hand, observations with too large variance may be due to experimental error and are not necessarily desirable. As such, the first preprocessing step of our experiments is to eliminate the relatively constant gene expressions from the database and regulate the variance of each gene expression. We follow a common preprocessing procedure [TSM+99] to apply a variance filter on each gene expressions. The variance filter adopts several parameters including min, max, α and γ. It first regulates the M-dimensional gene expression $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_M)$ according to

$$x_i^{new} = \begin{cases} max & \text{if } x_i \geq max, \\ min & \text{if } x_i \leq min, \\ x_i & \text{otherwise.} \end{cases} \tag{6.1}$$

Then the filter eliminates gene expressions which do not show a relative change of γ times and an absolute change of α units across all conditions. That is, a gene

expression will pass the filter only if

$$\begin{aligned} &\max\{x_i\} - \min\{x_i\} \geq \alpha, \quad \text{and} \\ &\max\{x_i\}/\min\{x_i\} \geq \gamma. \end{aligned} \tag{6.2}$$

With the parameter set $min = 20$, $max = 20,000$, $\alpha = 150$ and $\gamma = 3$ as suggested by Tamayo [TSM+99], 1,109 expressions from the YEAST data set and 1,423 expressions from the from HL60_U937_NB4_Jurkat data set passed the filter.

The second step of the preprocessing is to normalize the gene expressions to avoid the output clusters being dominated by vectors with significant length. For this purpose, we follow the practice of [YHR00] and apply the normalization with standard normal distribution within each observation panel of each gene expression. That is, for each elements $x_1, \ldots, x_i, \ldots, x_N$ in an observation panel (corresponding to each yeast cell's life circle, or each cell line of the human hematopoietic data),

$$x_i^{new} = \frac{x_i - \overline{\mathbf{x}}}{\sigma(\mathbf{x})}, \tag{6.3}$$

where $\overline{\mathbf{x}} = \frac{1}{N}\sum_i x_i$ is the mean and $\sigma(\mathbf{x}) = \sqrt{\frac{\sum_i (x_i - \overline{\mathbf{x}})^2}{N-1}}$ is the standard deviation. For the ease of the normalization, the intermediate 9th condition of the YEAST data set was excluded. This makes the actual dimension of the being used data to be 16.

Parameter Settings and Evaluation Methodology

All five clustering algorithms, namely ART-C 2A, ART 2A, SOM, online K-Means and batch K-Means are implemented in-house with C++ and share a common set of functions for vector manipulation. K-Means (both online and batch) and SOM

utilized Euclidean distances. Their reference clusters were initialized with random vectors by slightly perturbing from the mean vector of the input set.

SOM used a two-dimensional square map with corresponding square topological neighborhood (resonance domain). Its neighborhood size was initialized with half the total number of nodes. Gaussian neighborhood function was used. Various neighborhood shrinking strategies were tried before hand and the Gaussian shrinking function, which produced slightly better overall performance than others, was used. In addition, in each set of experiments dealing with different data set and different output map size, the Gaussian constants for these functions were fine-tuned in order to obtain an locally optimal output.

The learning rates of ART-C 2A, ART 2A, online K-Means and SOM were initialized with 0.05. We applied a simple threshold-linear function for the learning rate fading such that the learning rate $\eta^{(t+1)} = 0.9\eta^{(t)}$ if the network's recognition accuracy reached a threshold of 0.8. All the five algorithms were said to reach convergence if the cluster assignment to the input samples did not show a relative change of 0.5%.

We utilized the Euclidean distance for the evaluation of cluster compactness (Cmp) and cluster separation (Sep). $2\sigma^2 = 1.0$ as in Equation 5.3 was used to simplify our evaluation. On each data set, we evaluated the five algorithms on a varying number of output clusters. To obtain a statistically valid comparison, a batch of ten experiments using the same parameters for each algorithm were conducted. Each experiment randomly reshuffled the sequence of the input and trained the system to converge. The mean and the standard deviation of each evaluation measure over the ten experiments are reported. t-test was used to evaluate

the statistical significance of our comparison observation when appropriate.

Besides the cluster validity, we are particularly interested in the learning efficiency of each algorithm. To compare their learning efficiency, the number of learning iterations and the total CPU time cost being used by each algorithm to reach convergence were benchmarked and reported.

We note that the number of output clusters affects the score of all the evaluation measures used in our experiments. Strictly, two systems are not comparable if they work on different number of output clusters. The difficulty in our experiments is to suggest an appropriate ρ value for ART 2A in order to obtain a fixed C (such as 9 or 25) number of output clusters over a specific input sequence. To simplify our experiments, we manually tried various ρ values on one random input sequence, then used the ρ value which produced C output clusters on this input sequence in all the ten experiments. While on different input sequences the actual number of ART 2A output clusters may slightly vary from C, we found the variance was within an acceptable level that does not affect the validity of our comparison.

Evaluation Results

Both the two gene expression data sets are small scale, have a small number of features, and are densely distributed. Prior studies are capable of identifying a few number of expression patterns on these data sets only. Therefore on each data set, we set the target number of the output clusters to be relatively small. Table 6.4 and Table 6.5 report the five algorithms' cluster validity measures based on cluster distribution, together with the number of iterations to reach convergence and the

CPU time costs, when $C = 9$ and $C = 25$ (i.e. $K = 9$ and $K = 25$ respectively for K-Means), which correspond to a 3x3 and a 5x5 map in SOM respectively.

In all four batches of experiments, the cluster validity measures produced by ART-C 2A, in terms of both cluster compactness and cluster separation, were very close to those of ART 2A. Specifically, t-test did not suggest any significant difference between our observations on each evaluation measure.

In terms of cluster compactness, the validity measures of these five algorithms did not show significant differences in one the two data sets with $C = 9$. However, with $C = 25$, both online K-Means and batch K-Means produced significantly lower scores than the rest trio. In general, across these four batches of observations, online K-Means and batch K-Means slightly outperforms ART-C 2A and ART 2A in terms of cluster compactness. SOM did not seem to produce outstanding performance compared to the other four algorithms. In terms of cluster separation, the validity measures of both ART-C 2A and ART 2A were significantly lower than those of the other three algorithms. The difference among the latter three algorithms were not significant in our experiments.

In terms of efficiency, ART-C 2A incurred slightly more computational cost than ART 2A. With $C = 9$, the numbers of iterations used by ART-C 2A and ART 2A were relatively close to those of SOM and Online K-Means, which in turn were significantly fewer than that of batch K-Means. With $C = 25$, both ART-C 2A and ART 2A showed a significantly higher efficiency than online K-Means, batch K-Means and SOM, in the number of iterations as well as the CPU time cost. It deserves to point out that, on both data sets, the average CPU time cost of ART-C 2A on each iteration is nearly linear to the number of output clusters C

Table 6.4: Experimental results for ART-C 2A, ART 2A, SOM, Online K-Means and Batch K-Means on the YEAST data set, when the number of clusters C were respectively set to 9 and 25. I, T, Cmp, Sep and Ocq indicate the number of learning iterations, the cost of training time (in ms), *cluster compactness*, *cluster separation* and *overall cluster quality* respectively. All values are shown with the mean and the standard deviation over ten runs.

$C = 9$	Method	I	T (ms)	Cmp	Sep	$Ocq(0.5)$
	ART-C 2A	7.5 ± 2.3	134 ± 43	**0.7562 ± 0.0326**	**0.1416 ± 0.0063**	**0.4489 ± 0.0183**
	ART 2A	6.8 ± 3.1	112 ± 52	0.7595 ± 0.0206	0.1422 ± 0.0071	0.4509 ± 0.0118
	SOM	8.9 ± 3.0	169 ± 44	0.7749 ± 0.0324	0.1619 ± 0.0103	0.4684 ± 0.0210
	Online K-Means	6.5 ± 1.3	95 ± 18	0.7780 ± 0.0035	0.1512 ± 0.0031	0.4646 ± 0.0006
	Batch K-Means	12.3 ± 3.1	144 ± 36	0.7665 ± 0.0073	0.1639 ± 0.0063	0.4652 ± 0.0026
$C = 25$	Method	I	T (ms)	Cmp	Sep	$Ocq(0.5)$
	ART-C 2A	7.3 ± 3.5	315 ± 176	0.7059 ± 0.0207	0.1658 ± 0.0045	0.4359 ± 0.0102
	ART 2A	6.2 ± 3.1	245 ± 117	0.7240 ± 0.0183	**0.1655 ± 0.0050**	0.4448 ± 0.0087
	SOM	11.1 ± 3.2	597 ± 173	0.6745 ± 0.0218	0.1983 ± 0.0138	0.4364 ± 0.0174
	Online K-Means	14.0 ± 1.9	535 ± 74	**0.5592 ± 0.0228**	0.1834 ± 0.0047	**0.3713 ± 0.0122**
	Batch K-Means	12.8 ± 1.8	387 ± 52	0.6496 ± 0.0198	0.1850 ± 0.0046	0.4173 ± 0.0113

Table 6.5: Experimental results for ART-C 2A, ART 2A, SOM, Online K-Means and Batch K-Means on the HL60_U937_NB4_Jurkat data set, when the number of clusters C were respectively set to 9 and 25. I, T, Cmp, Sep and Ocq indicate the number of learning iterations, the cost of training time (in ms), *cluster compactness*, *cluster separation* and *overall cluster quality* respectively. All values are shown with the mean and the standard deviation over ten runs.

$C = 9$	Method	I	T (ms)	Cmp	Sep	$Ocq(0.5)$
	ART-C 2A	8.4 ± 6.3	178 ± 135	**0.7062 ± 0.0311**	0.1627 ± 0.0147	0.4345 ± 0.0153
	ART 2A	6.8 ± 3.0	132 ± 57	0.7090 ± 0.0314	**0.1551 ± 0.0133**	**0.4321 ± 0.0163**
	SOM	7.0 ± 2.4	164 ± 37	0.7327 ± 0.0407	0.1917 ± 0.0233	0.4622 ± 0.0320
	Online K-Means	7.3 ± 1.9	129 ± 34	0.7188 ± 0.0064	0.1903 ± 0.0133	0.4546 ± 0.0077
	Batch K-Means	14.6 ± 5.1	204 ± 72	0.7168 ± 0.0083	0.1897 ± 0.0120	0.4532 ± 0.0079
$C = 25$	Method	I	T (ms)	Cmp	Sep	$Ocq(0.5)$
	ART-C 2A	8.3 ± 4.9	430 ± 258	0.6267 ± 0.0189	0.1742 ± 0.0071	0.4004 ± 0.0079
	ART 2A	6.6 ± 4.4	302 ± 216	0.6573 ± 0.0303	**0.1736 ± 0.0061**	0.4154 ± 0.0140
	SOM	10.0 ± 4.2	641 ± 213	0.6541 ± 0.0473	0.2299 ± 0.0307	0.4420 ± 0.0384
	Online K-Means	14.0 ± 2.9	635 ± 130	**0.5370 ± 0.0314**	0.2141 ± 0.0075	**0.3755 ± 0.0144**
	Batch K-Means	13.1 ± 2.6	482 ± 93	0.6069 ± 0.0121	0.2238 ± 0.0087	0.4154 ± 0.0083

Table 6.6: ART-C 2A's average CPU time cost on each learning iteration over the YEAST and HL60_U937_NB4_Jurkat data sets. TPI is the CPU time cost per iteration (in ms). TPI/C calculates the prorated time cost for each cluster. It shows that on each data set, the TPI/C values for $C = 9$ and $C = 25$ are very close. This indicates the time cost of ART-C 2A is nearly linear to C, which also suggests that in the controlled experiments, the extra time cost of ART-C 2A's constraint reset process is not a dominate portion of the total time cost of ART-C 2A.

	$C = 9$		$C = 25$	
	TPI (ms)	TPI/C (ms)	TPI (ms)	TPI/C (ms)
YEAST	17.34 ± 5.13	1.93 ± 0.57	43.26 ± 16.96	1.73 ± 0.68
HL60_U937_NB4_Jurkat	21.65 ± 11.32	2.41 ± 1.26	52.15 ± 19.78	2.09 ± 0.79

(Table 6.6). This in practice shows that the extra time cost of the constraint reset process is of minor impact to the ART-C 2A's learning on these real-life data sets.

6.2 The Text Documents

Text documents may be one of the most widely seen data in the everyday life. While text documents are common and natural to human, they feature in large scale, large diversity and high ambiguity. This presents a great challenge to machine processing. Thus the text document domain has been serving as a test bed for numerous multi-disciplinary research, including natural language processing,

information extraction, information retrieval, knowledge and information management, information visualization and data mining, particularly text mining.

Clustering techniques are of great value for various text analysis studies. Applications of clustering techniques have been seen in most of the above multi-disciplinary studies. Our work on the text document domain focuses on the methodology study. That is, we evaluate the efficacy as well as the efficiency of the various algorithms on the text clustering problem, based on a common set of text processing techniques in the literature.

6.2.1 The Reuters-21578 Text Document Collection

The Reuters-21578 (REUTERS) text document collection [3] was originally released for evaluation of text categorization methods. The whole collection contains 21,578 English newswire articles residing with Reuters Ltd, each with varying document length. The news articles are stored in ASCII text format and annotated with SGML tags. The documents are further tagged into 135 categories (classes) based on the economic content. A portion of the documents are labeled with multiple categories. For documents in each category, there are further divided into *training* and *testing* sets. Depending on the different splitting criteria (such as ModLewis, ModApte and ModHayes etc.), there may be varying portion of unused documents (i.e. neither in training set nor in testing set). The class labels available on each document enable keyword feature selection and quality evaluation using class conformity based measures in our experiments.

[3] The REUTERS corpus is available via http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

Data Pre-processing

Our experiments adopt the documents used by the ModApte split, which screens out 8,676 documents from the collection and makes use of the remaining 12,902 documents. For the ease of our experiments, we adopt a sub-set of the collection that contains the training and testing documents from the top ten categories, in terms of the number of documents in a category. Documents with multiple class labels were duplicated so that each copy was associated with one class label accordingly.

A pre-requisite of text processing is to extract a suitable feature representation of the documents. Typically, word stems are suggested as the representation units by information retrieval research [Cav94, vR79, TWC02]. We adopt the bag-of-words representation of document features, which projects the document feature into a discrete keyword space and hence converts the feature of the free text documents into vector format [ST00].

To select the keyword features, CHI (χ) statistics is adopted as the ranking metric in our experiments. A prior study on several well-known corpora including Reuters-21578 and OHSUMED has showed that CHI statistics generally outperforms other feature ranking measures, such as term strength (TS), document frequency (DF), mutual information (MI) and information gain (IG) [YP97].

The CHI statistics that works on the binary category splitting is described as below. For a word token t, its CHI measure is defined by

$$CHI(t) = \sqrt{\frac{(n_{pt+} + n_{nt+} + n_{pt-} + n_{nt-})(n_{pt+}n_{nt-} - n_{nt+}n_{pt-})^2}{(n_{pt+} + n_{pt-})(n_{nt+} + n_{nt-})(n_{pt+} + n_{nt+})(n_{pt-} + n_{nt-})}} \tag{6.4}$$

where n_{pt+} and n_{nt+} are the number of documents in the positive category and the negative category respectively in which the token t occurs at least once; and n_{pt-} and n_{nt-} are the number of documents in the positive category and the negative category respectively in which the token t doesn't occur. A higher CHI measure indicates a greater correlation between the word token and the positive/negative category. A token with higher CHI measure is hence more suitable for presenting the feature of the documents based on the current category splitting.

The CHI feature selection is carried out on the ten categories individually. For each category, we treat it as the positive category and the other nine categories as the negative category. Keywords based on this splitting are then selected through threshold cut-off. Keywords for all ten category splittings are finally combined and the duplications are removed. With a preset threshold $\chi \geq 15$, we obtained 365 keywords as the features.

During feature extraction, the document is first segmented and converted into a keyword frequency vector $(tf_1, tf_2, \ldots, tf_D)$, where tf_i is the in-document term frequency of keyword w_i and D is the number of keywords being selected (i.e. the dimension of the input vector). A term weighting method based on *inverse document frequency* (IDF) [SB88] and the Euclidean normalization is then applied on the frequency vector to produce the keyword feature vector

$$\mathbf{x}^1 = \Re(\mathbf{x}^0) = \Re((x_1^0, x_2^0, \ldots, x_D^0)), \tag{6.5}$$

where $\Re$ is defined by Equation 4.2 and x_i^0 is computed by

$$x_i^0 = (1 + \log_2 tf_i) \log_2 \frac{n}{n_i}, \tag{6.6}$$

where n is the number of documents in the whole document set and n_i is the number of documents in which the keyword w_i occurs at least once. During feature

Table 6.7: The statistics of the top-10-category subset of the Reuters-21578 text collection.

Category	No. of keywords	No. of docs (excl. null)
acq	43	2,319
corn	17	237
crude	64	577
earn	21	3,569
grain	35	582
interest	30	477
money-fx	65	715
ship	52	285
trade	79	486
wheat	17	283
Total no. of keywords (excl. dup.)		365
Total no. of docs (excl. null)		9,530

extraction it is also necessary to remove the null vectors (i.e. documents with $tf_i = 0$ for all i) from the collection as they present no useful information on the document content.

The final document collection used in our benchmark thus contains 9,530 vectors, each being 365-dimensional. The statistics of the data set is summarized in Table 6.7.

Evaluation Results

The five algorithms being studied in our experiments, namely ART-C 2A, ART 2A, SOM, online K-Means and batch K-Means, adopted the same parameter settings as those in Section 6.1.2.

To obtain a better understanding of each algorithm's learning efficiency, we tested them on ten subsets of the REUTERS data set constructed as follows. Documents from each class were evenly split into ten folds. The ith subset used in our experiments contained document folds $1, \ldots, i$ from each category. In this way, all the ten subsets used in our experiments had nearly identical document class distribution, while the number of data samples in each subset varied from 957 to 9,530.

The well annotated category (class) labels available on the REUTERS data set enable the evaluation of cluster validity using class conformity based measures. As such, the cluster validity measures were evaluated using both cluster distribution and class conformity. In addition, we benchmark the learning efficiency of each algorithm based on the number of iterations and CPU time cost used to reach convergence.

In contrast to the two gene expression data sets, the REUTERS data set is relatively high-dimensional, large-scale, noisy, and sparsely distributed. Therefore we did not expect a cluster algorithm to replicate the exact ten clusters corresponding to the labeled classes in our experiments. Instead, we tested the algorithms on each subset with $C = 25$, $C = 49$ and $C = 81$ for ART-C, corresponding to $K = 25$, $K = 49$ and $K = 81$ for K-Means and a 5x5, 7x7 and 9x9 map in SOM

respectively. The comparative experiments with different C values showed very similar results. Experimental results with $C = 49$ are reported in Figure 6.5.

It is interesting that all the five algorithms produced rather consistent cluster validity scores in response to the varying number of input samples. This is probably due to the similar data distribution in each subset used in the experiments. In terms of cluster compactness (Cmp), batch K-Means produced significantly better scores than online K-Means and SOM in all the experiments, while the latter two in turn performed slightly better than ART-C 2A and ART 2A in most experiments. In terms of cluster separation (Sep), all the five algorithms performed quite similarly in all experiments.

Using the set of validity measures based on class conformity, ART-C 2A and ART 2A produced significantly higher cluster entropy (Enc) scores than those of online K-Means, batch K-Means, and SOM, while the performance of the latter three methods were quite close. As for class entropies (Enl), all algorithms produced similar scores. Interestingly, these observations generally harmonize with the comparable results using the cluster distribution based measures.

In terms of efficiency, both ART-C 2A and ART 2A showed a significantly higher efficiency than online K-Means. The CPU time cost of ART-C 2A and ART 2A were about half of that of online K-Means. Online K-Means in turn was significantly faster than batch K-Means and SOM in all experiments. This is reflected by both the number of iterations and the CPU time cost.

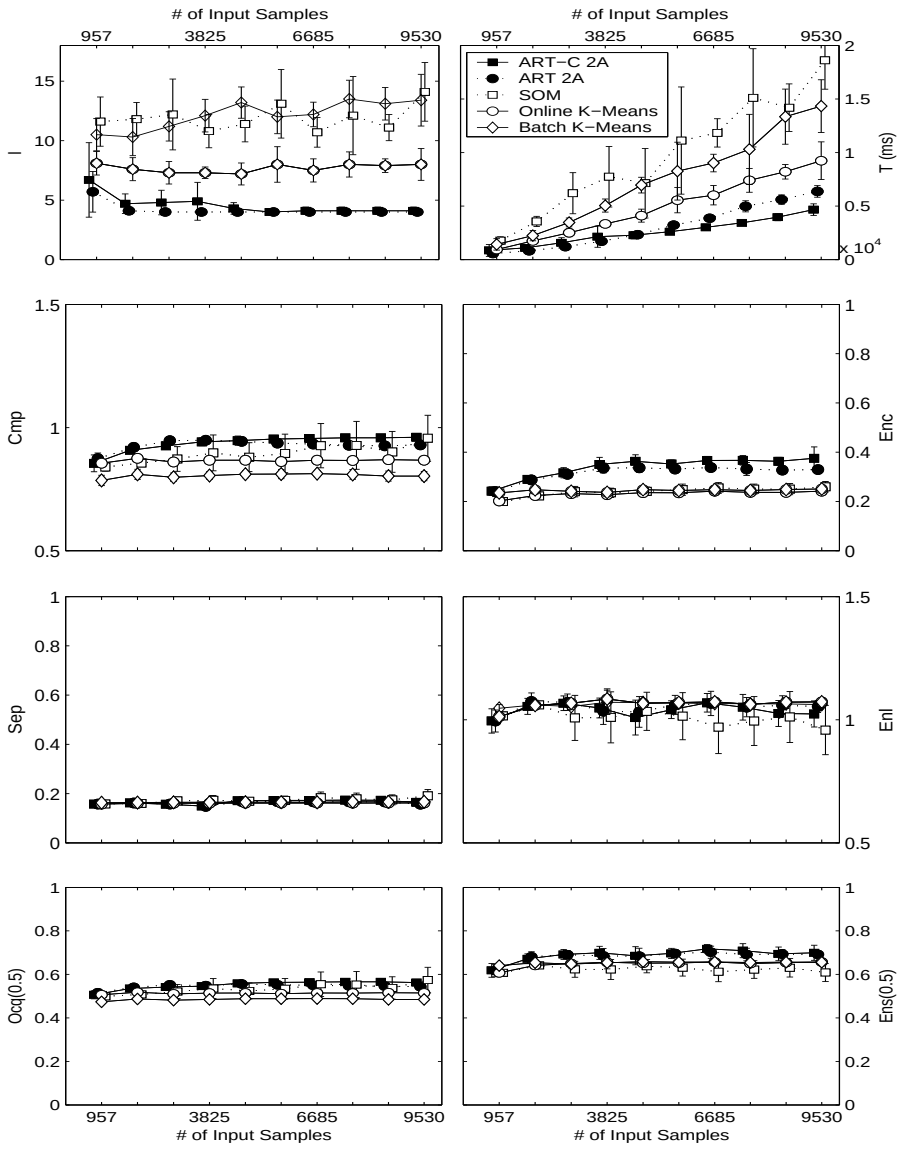


Figure 6.5: Experimental results for ART-C 2A, ART 2A, SOM, Online K-Means and Batch K-Means on the Reuters-21578 data set with 49 clusters. I and T indicate the number of learning iterations and the cost of training time (ms) respectively. *Sep*, *Cmp*, and *Ocq* indicate *cluster separation*, *cluster compactness*, and *overall cluster quality* respectively. *Enc*, *Enl*, and *Ens* indicate *cluster entropy*, *class entropy*, and *overall entropy* respectively. All values are shown with the mean and the standard deviation over ten runs.

6.3 Discussions and Concluding Remarks

In the reported case studies, the ART-C 2A network has been successfully applied to the clustering of the rat CNS gene expressions. The clustering results shows a strong correlation between the gene functional groupings and the discovered clusters. The results are reasonably comparable to those of FITCH, which is based on K-Means, also reviewed in this thesis. This suggests that the appropriate use of a clustering algorithm could greatly help in discovering the underlying knowledge from real-life problems. Interestingly, the comparison between the ART-C 2A output and the FITCH output shows different characteristics of these two learning algorithms. Generally speaking, the nature of ART-C 2A learning paradigm that generates new recognition categories using distinct input samples tends to be more capable of discovering differences among the sub-groupings. However, such a learning habit, compared with FITCH's, tends to be more sensible to noises and outliers and may generate a greater number of distinct clusters that contain small number of samples only. This in turn raises the risk of dead node (cluster that contains no sample) generation.

The benchmark on the cluster validity of the five clustering algorithms, namely ART-C 2A, ART 2A, SOM, online K-Means and batch K-Means, led to mixed results. Generally speaking, the performance of ART-C 2A is quite comparable to that of ART 2A. Compared with Online K-Means and Batch K-Means, on the gene expression data sets, ART-C 2A and ART 2A output with comparable intra-cluster compactness and better inter-cluster separation. While on the REUTERS data set, ART-C 2A and ART 2A output with worse intra-cluster compactness and comparable inter-cluster separation, both reflected by the cluster distribution

based measures and class conformity based measures.

As another large family of self-organizing neural networks, SOM did not show notably outstanding performance over others, even with the optimized parameter settings in our controlled experiments. However readers shall note the application domain of SOM is mainly on topology preserving mapping and visualization, rather than clustering.

Additionally, we must point out that the observations above reflect the nature of clustering. As a matter of fact, the ill-posed clustering problem "precludes an absolute judgement as to the relative efficacy of all clustering techniques" [BA02a, BA02b].

We are particularly interested in the relatively high efficiency of ART-C 2A and ART 2A reflected in our controlled experiments. This is due to their capabilities of dynamically initializing the reference clusters using distinct input samples through the network's mismatch reset cycle. Mismatch reset ensures stable encoding of new samples through one scan. The constraint reset process in ART-C 2A also serves to move cluster centroids quickly from a high density area to a low density area, with minor impact to the learning history. In contrast, SOM and K-Means slowly adjust at least one of their existing recognition patterns towards distinct inputs, and require typically more than one learning iterations to stably encode there. This is clearly reflected in our experiments as the numbers of iterations used by SOM and K-Means are generally larger than those of ART and ART-C. However, when working with too few number of output clusters, which corresponds to a very low vigilance threshold, such an advantage is not notable in our experiments, as both ART-C 2A and ART 2A work rather like the competitive learning in this scenario.

As a general guideline, the advantages of ART and ART-C will be mostly suitable for online learning of large scale, incremental input data.

Despite of the advantage above, readers shall note that both ART-C 2A and ART 2A require a Euclidean normalization on the input and category representation in order to avoid category proliferation. As such, the input vector length information is ignored by the networks. This limits the application of ART-C 2A and ART 2A to the problems where the input vector length information is not of critical importance.

As our concluding remarks, the ART-C learning paradigm retains the efficient cluster creation capability of ART, and allows a user to directly control the number of the output clusters by imposing a constraint on ART category learning. The constraint reset mechanism of ART-C adaptively adjusts the network's vigilance threshold which guides the network's learning and redistributes the recognition categories to satisfy the constraint. As such, unlike a conventional ART module which requires prior knowledge in estimating an appropriate *vigilance* parameter, the knowledge in estimating an optimal *number of clusters over the data set* is required by an ART-C module. We consider this to be a good alternative to the conventional ART module and is of great value for various real-life applications where the knowledge for the global estimation of the optimal number of clusters is more conceivable than that for the local estimation of intra-cluster variances.

CHAPTER 7
SUMMARY AND FUTURE WORK

As one of the primary research domains, pattern analysis covers a large variety of multi-disciplinary studies spanning in numerous application domains. The focus of this thesis is on the methodology study. Particularly, the purpose of this thesis is to explore efficient unsupervised learning algorithms for cluster analysis that require minimal prior knowledge on the problem domain and the system's parameter setting, in view of the large scale input data in real-life applications.

In this thesis, a novel neural network architecture based on competitive learning has been proposed and studied. The proposed network, named ART-C (for Adaptive Resonance Theory under Constraint), has the following improvements:

- It tackles ART's dependency on the user's prior knowledge in estimating the distribution of the input, thus provides a more intuitive application for real-life problems.

- It shows satisfactory performance for clustering of real-life data, including gene expressions and text documents. Its clustering efficacy is comparable to that of algorithms in the same family, including ART, SOM and K-Means.
- It shows distinguishably higher efficiency on large scale inputs, compared with algorithms in the same family.

One challenging task in cluster analysis is the quantitative assessment of the cluster validity. Previous studies in the literature are mostly focused on tuning the parameters of one algorithm in controlled experiments. In view of the existing validation measures, this thesis proposes two sets of evaluation measures, respectively based on cluster distribution and class conformity. Experiments have shown that these validity measures are capable of systematically indexing subtle differences between different clustering solutions, which in turn serve as valuable guideline for various studies in clustering process, including choosing optimal feature representation and pattern proximity measure, tuning parameters of a clustering algorithm, and cross-method comparison.

In view of the previous research and the advancement of the pattern analysis technologies, the following topics are suggested in the future work:

1. **Fully automatic clustering:** To simplify the problem, most existing clustering algorithms assume some parameters of the problem model (such as the number of clusters) are known. Designing a fully automatic clustering algorithm that requires no user knowledge still remains a challenge, yet it offers a great potential in various application domains. Fully automatic clustering essentially involves a search for optimal clustering solution. Prior studies

like [PR02] involve the evaluation of the codebook during each learning iteration, and determine whether to add elements to some clusters or remove elements from them. Greedy techniques are usually used to determine the semi-optimal number of clusters. However since a global search of the optimal solution is NP-hard [GJW80], how to design an appropriate heuristic for the search process is yet a challenging work.

2. **Noise-free pattern analysis:** Noisy data that contain outliers are very common in most real-life applications. In some circumstances they are of no contribution to problem solving; yet in other circumstances they may indicate emerging patterns and hence are of great value. Identifying these distinct emerging patterns is as important as identifying the major patterns for analysis purpose. How to exclude the "actual" noise without losing meaningful distinct patterns thus remains as an interesting topic. A well-designed information filtering algorithms in signal processing area could be a great solution for this purpose. For example, WaveCluster [SCZ00] applies wavelet transformation to preprocess the primary data, filters out the noises and traces the boundaries of high density data groupings using image-processing-based method. WaveCluster however is incapable of handling high-dimensional data due to the computational complexity. How to apply signal processing methods to high dimensional data yet remains an challenging topic.

BIBLIOGRAPHY

[ADR94] M. Al-Daoud and S. Roberts. New methods for the initialisation of clusters. Technical Report 94.34, School of Computer Studies, University of Leeds, 1994.

[AM90] I. Aleksander and H. Morton. *An Introduction to Neural Computing*. Chapman and Hall, London, 1990.

[And73] M.R. Anderberg. *Cluster Analysis for Applications*. Academic Press, Inc., New York, NY, 1973.

[BA98] A. Baraldi and E. Alpaydin. Simplified ART: A new class of ART algorithms. Technical Report TR-98-004, Berkeley, CA, 1998.

[BA02a] Andrea Baraldi and Ethem Alpaydm. Constructive feedforward ART clustering networks - part I. *IEEE Transactions on Neural Networks*, 13(3):645–661, 2002.

[BA02b] Andrea Baraldi and Ethem Alpaydm. Constructive feedforward ART clustering networks - part II. *IEEE Transactions on Neural Networks*, 13(3):662–677, 2002.

[BB95] L. Bottou and Y. Bengio. Convergence properties of the K-Means algorithms. In G. Tesauro, D. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing System 7*. MIT Press, 1995.

[BDSY99] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[BF98] Paul S. Bradley and Usama M. Fayyad. Refining initial points for K-Means clustering. In *Proceedings of 15th International Conference on Machine Learning*, pages 91–99. Morgan Kaufmann, San Francisco, CA, 1998.

[BF01] Y. Barash and N. Friedman. Context-specific bayesian clustering for gene expression data. In *The Fifth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 12–21, 2001.

[BH65] G.H. Ball and D.J. Hall. ISODATA, a novel method of data analysis and classification. Technical report, Stanford University, 1965.

[BK00] A.J. Butte and I.S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Proceedings of the Pacific Symposium on Biocomputing*, 2000.

[Bol98] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[Buc02] B.G. Buchanan. Poster of the AI TOPICS booth at AAAI conference, 2002.

[BW00] G. Bartfai and R. White. Incremental learning and optimazation of hierarchical clusterings with ART-based modular networks. In L.C. Jain, B. Lazzerini, and U. Halici, editors, *Innovations in ART Neural Networks*, pages 87–132. Physica-Verlag, 2000.

[Cav94] W. Cavnar. Using an N-gram-based document representation with a vector processing retrieval model. In *Proceedings of TREC 3*, pages 269–278, 1994.

[CCW+98] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. conway, L. Wodicka, T.G. Wolfsberg, A.E. Gaberielian, D. Landsman, D.J. Lockhart, and R.W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[CG87a] G.A. Carpenter and S. Grossberg. ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[CG87b] G.A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image processing*, 34:54–115, 1987.

[CG90] G.A. Carpenter and S. Grossberg. ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

[CGR91a] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by self-organizing neural network. *Neural Networks*, 4:565–588, 1991.

[CGR91b] G.A. Carpenter, S. Grossberg, and D.B. Rosen. ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4:493–504, 1991.

[CGR91c] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[CH67] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[Das91] B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Las Alamitos, California, 1991.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelyhood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B(39):1–38, 1977.

[DMR00] M. Dittenbach, D. Merkl, and A. Rauber. The growing hierarchical self-organizing map. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 15–19, Como, Italy, July 2000.

[DS73] E. Diday and Simon. The dynamic cluster method in non-hierarchical clustering. j. comput. inf. sci. 2, 61c88. diday, e. and simon,. *Journal of Computer Information Science*, 2:61–88, 1973.

[Dui04] Bob Duin. The pattern recognition files. Web Portal, 2004. Available via http://www.ph.tn.tudelft.nl/PRInfo/.

[EB99] M.B. Eisen and P.O. Brown. DNA arrays for analysis of gene expression. *Methods Enzymol*, 303:179–205, 1999.

[EKS+98] M. Ester, H.P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th VLDB Conference*, 1998.

[EKSX96] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference On Knowledge Discovery and Data Mining (KDD)*, 1996.

[ESBB98] M. Eisen, P.T. Spellman, D. Botstein, and P.O. Brown. Cluster analysis and display of genome-wide expression patterns. In *Proceedings of National Academy of Science USA*, volume 95, pages 14863–14867, 1998.

[FK99] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Neural Networks*, 21(5):450–465, 1999.

[FKKN96] Pasi Franti, Juha Kivijarvi, Timo Kaukoranta, and Olli Nevalainen. Genetic algorithms for codebook generation in vector quantization. Technical Report A-1996-5, University of Joensuu, Department of Computer Science, 1996.

[FKN98] P. Franti, J. Kivijarvi, and O. Nevalainen. Tabu search algorithm for codebook generation in vector quantization. *Pattern Recognition*, 31(8):1139–1148, 1998.

[Fle97] A. Flexer. Limitations of self-organizing maps for vector quantization and multidimensional scaling. *Advances in Neural Information Processing Systems 9.*, pages 445–451, 1997.

[Fle99] A. Flexer. On the use of self-organizing maps for clustering and visualization. In *Principles of Data Mining and Knowledge Discovery*, pages 80–88, 1999.

[Fri94] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441 – 1460, 1994.

[Fri97] B. Fritzke. Some competitive learning methods, 1997. Draft document.

[GJW80] M.R. Garey, D.S. Johnson, and H.S. Witsenhausen. The complexity of the generalized Lloyd-max problem. *IEEE Transactions on Information Theory*, 28(2):255–256, 1980.

[GP00] E. Gokcay and J.C. Principe. A new clustering evaluation function using Renyi's information potential. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

[Gro76a] S. Grossberg. Adaptive pattern classification and universal recoding. I. parallel development and coding of neural feature detector. *Biological Cybernetics*, 23:121–134, 1976.

[Gro76b] S. Grossberg. Adaptive pattern classification and universal recoding. II. feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, 23:187–202, 1976.

[Gro82] S. Grossberg. *Studies of Mind and Brain.* D. Reidel Publishing, 1982.

[GRS98] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference*, 1998.

[GRS99] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the IEEE Conference on Data Engineering*, 1999.

[Hay99] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition, 1999.

[HBV01] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Intelligent Information Systems*, 2001.

[Heb49] Hebb. *The Organization of Behavior*. 1949.

[HH93] C. Huang and R. Harris. A comparison of several vector quantization codebook generation approaches. *IEEE Transactions on Image Processing*, 2(1):108–112, 1993.

[HK98] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of Knowledge Discovery and Data Mining (KDD)*, pages 58–65, 1998.

[HKLK97] Timo Honkela, Samuel Kaski, Krista Lagus, and Teuvo Kohonen. WEBSOM — self-organizing maps of document collections. In *Workshop on Self-Organizing Maps*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[HKP91] K. Hertz, A. Krogh, and R. Palme. In *Introduction to the Theory of Neural Computation*, chapter 9, pages 217–243. Addison-Wesley, 1991.

[HL95] C. Hung and S. Lin. Adaptive Hamming Net: a fast-learning ART 1 model without search. *Neural Networks*, 8(4):605–618, 1995.

[HLL+96] H.L. Hung, H.Y. Mark Liao, S.J. Lin, W.C. Lin, and K.C. Fan. Cascade fuzzy ART: a new extensible database for model-based object recognition. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 187–198, 1996.

[Hor88] B.K.P. Horn. *Robot Vision*. The MIT Press, 4 edition, 1988.

[HTT02] Ji He, Ah-Hwee Tan, and Chew-Lim Tan. ART-C: A neural architecture for self-organization under constraints. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 2550–2555, 2002.

[HTT04] Ji He, Ah-Hwee Tan, and Chew-Lim Tan. Modified ART 2A growing network capable of generating a fixed number of nodes. *IEEE Transactions on Neural Networks*, 15(3):728–737, 2004.

[HTTS03] Ji He, Ah-Hwee Tan, Chew-Lim Tan, and Sam-Yuan Sung. On quantitative evaluation of clustering systems. In Weili Wu, Hui Xiong, and Shashi Shekhar, editors, *Information Retrieval and Clustering*, pages 105–133. Kluwer Academic Publishers, 2003.

[HVB00] M. Halkidi, M. Vazirgiannis, and I. Batistakis. Quality scheme assessment in the clustering process. In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 265–276, 2000.

[HVD01] J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–136, 2001.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Upper Saddle River, NJ, 1988.

[JDM00] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JS96] Y. Jhung and P.H. Swain. Bayesian contextual classification based on modified m-estimates and markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):67–7, 1996.

[Kel99] S. Kelly. Creation of a pea expressed sequence tag array. Undergraduate Thesis, University of Maniboba, 1999.

[KKL+00] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, 2000.

[KKZ94] I. Katsavounidis, C. Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, 1994.

[Koh97] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, second edition, 1997.

[KR87] L. Kaufmann and P.J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis based on the L1 Norm*, pages 405–416. 1987.

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

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communication*, 28:84 – 95, 1980.

[LGG01] N.M. Luscombe, D. Greenbaum, and M. Gerstein. What is bioinformatics? An introduction and overview. *Yearbook of Medical Informatics*, pages 83–100, 2001.

[LGL99] R.J. Lipshutz, T.R. Gingeras, and D.J. Lockhart. High density synthetic oligonucleotide arrays. *Nat Gen*, 21(1):20–24., 1999.

[Llo57] S.P. Lloyd. Least squares quantization in PCM. Technical report, Bell Labs, 1957. Published in 1982 in IEEE Transactions on Information Theory.

[LVV01] A. Likas, N. Vlassis, and J.J. Verbeek. The global k-means clustering algorithm. Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, IAS-UVA-01-02 2001.

[Mac67] J.B. MacQueen. Some methods for classification and analysis of multivariante observations. In *Proceedings of the 5th Berkely simposium in mathematics and probability*, pages 281–297, 1967.

[MCA+98] G. Michaels, D. Carr, M. Askenazi, S. Fuhrman, X. Wen, and R. Somogyi. Cluster analysis and data visualization of large-scale gene expression data. In *Pacific Symposium on Biocomputing*, 3, pages 42–53, 1998.

[Mit97] T.M. Mitchell. *Machine Learning*. The McGraw-Hill Companies Inc., 1997.

[MJ92] J. Mao and A.K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

[Moo89] B. Moore. ART 1 and pattern clustering. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 174–1985, 1989.

[Mor88] W. Morris. *Morris Dictionary of Word and Phrase Origins*. Harper-Resource, 2nd edition, 1988.

[MS57] C. D. Michener and R. R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.

[MS91] T.M. Martinetz and K.J. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397 – 402. 1991.

[NH94] R. Ng and J. Han. Effecient and effective clustering methods for spatial data mining. In *Proceedings of the 20nd VLDB Conference*, 1994.

[NH02] R.T. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[Nil96] N.J. Nilsson. Introduction to machine learning. An early draft of a proposed textbook, September 1996.

[PR02] Giuseppe Patane and Marco Russo. Fully automatic clustering system. *IEEE Transactions On Neural Networks*, 13(6):1285–1298, 2002.

[PU99] Andres Perez-Uribe. *Structure-Adaptable Digital Neural Networks*. PhD thesis, Logic Systems Laboratory, Computer Science Department, Swiss Federal Institute of Technology, 1999.

[Rau99] A. Rauber. LabelSOM: on the labeling of self-organizing maps. In *International Joint Conference on Neural Networks*, volume 5, pages 3527–32, Piscataway, NJ, 1999. IEEE Service Center.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533 – 536, 1986.

[RS98] G.D. Ramkumar and Arun N. Swami. Clustering data without distance functions. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1):9–14, 1998.

[RZ85] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, 9:75–112, 1985.

[SB88] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[SCZ00] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A wavelet-based clustering approach for spatial data in very large databases. *The International Journal on Very Large Databases*, 8(4):289–304, 2000.

[SEWB00] Hagit Shatkay, Stephen Edwards, W. John Wilbur, and Mark Boguski. Genes, themes and microarrays - using information retrieval for large-scale gene analysis. In *Proceedings of ISMB*, 2000.

[Sim86] H.A. Simon. Research briefings 1986: Report of the research briefing panel on decision making and problem solving. Technical report, National Academy of Sciences, 1986.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[ST00] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215, 2000.

[sWMB95] R. somogyi, X. Wen, W. Ma, and J.L. Barker. Developmental kinetics of GAD family mRNSs parallel neurogenesis in the rat spinal cord. *Journal of Neuroscience*, 15(4):2575–2591, 1995.

[Sym81] M.J. Symon. Clustering criterion and multivariate normal mixture. *Biometrics*, 37:35–43, 1981.

[Tan95] A.H. Tan. Adaptive Resonance Associative Map. *Neural Networks*, 8(3):437–446, 1995.

[Tan97] A.H. Tan. Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. *IEEE Transactions on Neural Networks*, 8(2):237–235, 1997.

[TG74] J.T. Tou and R.C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Massachusetts, 1974.

[THE+99] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. Technical report, Department of Health Research and Policy, Stanford University, 1999.

[TK99] S. Theodoridis and K. Koutroubas. *Pattern recognition*. Academic Press, 1999.

[TSM+99] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to homatopoietc differentiation. In *Proceedings of the National Academy of Science*, volume 96, pages 2907–2912, 1999.

[TWC02] C.M. Tan, Y.F. Wang, and C.D.Lee. The use of bigrams to enhance text categorization. *Information Processing and Management*, 30:529–546, 2002.

[vR79] C.J. van Rijsbergen. *Information Retieval*. Butterworths, London, 1979.

[VR92] N. Venkateswarlu and P. Raju. Fast ISODATA clustering algorithms. *Pattern Recognition*, 25(3):335–342, 1992.

[VZVK95] V.E. Velculescu, L. Zhang, B. Vogelstein, and K.W. Kinzler. Serial analysis of gene expression. *Science*, 270:484–487, 1995.

[Wat85] S. Watanabe. *Pattern Recognition: Human and Mechanical*. Wiley, New York, 1985.

[WD95] D. Wettschereck and T.G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19(1):5–27, 1995.

[WFM+98] X. Wen, S. Fuhrman, G.S. Michaels, D.B. Carr, S. Smith, G.L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. In *Proceedings of the National Academy of Science*, pages 334–339, 1998.

[WH60] B. Widrow and M. E. Ho. Adaptive switching circuits. *Western Electronic Show and Convention, Convention Record*, 4:96 – 104, 1960.

[WYM97] W. Wang, J. Yang, and R.R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 186–195, 1997.

[YFRR01] K.Y. Yeung, C. Fraley, A.E. Raftery, and W.L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.

[YHR00] K.Y. Yeung, D.R. Haynor, and W.L. Ruzzo. Validating clustering for gene expression data. Technical Report UW-CSE-00-01-01, Department of Computer Science and Engineering, University of Washington, January 2000.

[YP97] Y. Yang and J.P. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 412–420, 1997.

[ZFLW02] O. R. Zaiane, A. Foss, C.H. Lee, and W. Wang. On data clustering analysis: Scalability, constraints, and validation. In M.-S. Chen, P.S. Yu, and B. Liu, editors, *PAKDD 2002*, pages 28–39, Taipei, May 2002. Springer-Verlag Berlin Heidelberg.

[ZRL96] T. Zhang, R. Ramakrishnman, and M. Linvy. BIRCH: An efficient method for very large databases. In *Proceeding of ACM SIGMOD Conference*, 1996.

AUTHOR BIOGRAPHY

Ji He is a PhD candidate in the Department of Computer Science, School of Computing, National University of Singapore and Institute for Infocomm Research, Singapore. His research interests include text mining, knowledge discovery, machine learning and neural networks. He obtained a Bachelor of Science in Electronic Engineering in 1997 and a Master of Science in Information Science and Management in 2000 from Shanghai Jiao Tong University, China.

During his PhD candidature, his publications include:

- Ji He, Man Lan, Chew-Lim Tan, Sam-Yuan Sung, and Hwee-Boon Low. Initialization of Cluster Refinement Algorithms: A Review and Comparative Study. In *the Proceedings of International Joint Conference on Neural Networks (IJCNN)*. July 2004. Budapest, Hungary.
- Ji He, Ah-Hwee Tan, and Chew-Lim Tan. Modified ART 2A Growing Network Capable of Generating A Fixed Number of Nodes. *The IEEE Transactions on Neural Networks*. 15(3), p728-737, 2004.
- Ji He, Chew-Lim Tan, Hwee-Boon Low, and Dan Shen. Unsupervised Learning for Document Classification: Feasibility, Limitation, and the Bottom Line. In *the International Joint Conference on Natural Language Processing*. March 2004. Sanya, China.

- Ji He, Ah-Hwee Tan, Chew-Lim Tan, and Sam-Yuan Sung. On quantitative evaluation of clustering systems. In Weili Wu et al, editors, *Information Retrieval and Clustering*. Kluwer Academic Publishers, Boston Hardbound, ISBN 1-4020-7682-7. December 2003.

- Ji He, Ah-Hwee Tan and Chew-Lim Tan. Self-organizing Neural Networks for Efficient Clustering of Gene Expression Data. In *the Proceedings of International Joint Conference on Neural Networks (IJCNN)*. July 2003. Portland, OR, USA. p1684-1689.

- Ji He, Ah-Hwee Tan, and Chew-Lim Tan. On Machine Learning Methods for Chinese Documents Classification. *Applied Intelligence*. 18(3), p311-322, 2003.

- Ji He, Ah-Hwee Tan and Chew-Lim Tan. ART-C: A Neural Architecture for Self-Organization Under Constraints. In *the Proceedings of International Joint Conference on Neural Networks (IJCNN)*. May 2002. Hawaii, USA. p2550-2555.

- Ji He, Ah-Hwee Tan and Chew-Lim Tan. Machine Learning Methods for Chinese Web Page Categorization. In *the ACL'2000 2nd Workshop on Chinese Language Processing*. October 2000. Hongkong, China. p93-100.

- Ji He, Ah-Hwee Tan and Chew-Lim Tan. A Comparative Study on Chinese Text Categorization Methods. In *the PRICAI'2000 International Workshop on Text and Web Mining*. August 2000. Melbourne, Australia. p24-35.