NANYANG TECHNOLOGICAL UNIVERSITY

TOWARDS GENERAL SEMI-SUPERVISED CLUSTERING USING A COGNITIVE REINFORCEMENT K-ITERATION FAST LEARNING ARTIFICIAL NEURAL NETWORK (R-KFLANN)

RINA TSE

SCHOOL OF COMPUTER ENGINEERING

2010

TOWARDS GENERAL SEMI-SUPERVISED CLUSTERING USING A COGNITIVE REINFORCEMENT K-ITERATION FAST LEARNING ARTIFICIAL NEURAL NETWORK (R-KFLANN)

RINA TSE

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2010

ACKNOWLEDGEMENTS

The author would like to express her sincere gratitude to her supervisor, Assistant Professor Alex Tay, for his valuable advice, guidance and enthusiastic support throughout her work on the project. She would also like to thank him for giving her the opportunity to explore and research into this interesting area. It has been a rewarding experience working with him, not only in academic research and training, but also in broader philosophical aspects.

Also, the author would also like to express her appreciation to her fellow graduate students, Mr. Xuejie Zhang and Mr. Paul Nguyen, for their help, suggestions and insightful opinions through countless discussions in the lab.

In addition, the author would also like to take this opportunity to thank the Emerging Research Lab and its staff, Ms. Dewi Muharyani Cendrawasih and Ms. Thin Nandar Soe, for their helpful support and suggestions and for providing a pleasant research environment.

Last but not least, the author would like to thank her family and friends who patiently provide the moral support and encouragement through the course of her research study. To them she dedicates this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The study of a semi-supervised clustering has recently attracted great interest from the data clustering community. Work in semi-supervised clustering systems has been done focusing on specific types of auxiliary information, i.e. a partial labeling or pairwise constraints. However, in some applications the clustering characteristics desired may not be restricted to only these predefined types of constraints. Furthermore, the user may not always be able to formulate an explicit clustering specification. In such cases, only a weak good or bad evaluation feedback is obtainable as semi-supervision information. This research proposes a novel form of semi-supervised clustering using Reinforcement K-Iteration Fast Learning Artificial Neural Network (R-KFLANN) architecture that utilizes generic reward or punishment feedback, enabling it to address different types of high-level clustering requirements provided at run-time. To illustrate this concept, the R-KFLANN was tested in two different application domains of data classification and robot mapping. The results indicate that the system was able to adapt to the online reinforcement presented and eventually improve the output in serving the covert specifications of both tasks. It could significantly improve the clusters using only overall failure rate information loosely coupled with the hidden class labels in the classification problem. This was also evident in the navigation problem when the clustering specification could not even be explicitly formulated; R-KFLANN was still able to incorporate the high-level task's characteristics into the cluster representation, yielding a better map efficiency. Additionally, it could fulfill the navigation task requirements which would not be achievable unless the system was tuned manually using a small tolerance. These findings suggest the usefulness of R-KFLANN semi-supervised clustering in serving clustering objectives imposed online by the high-level tasks without being restricted to the traditional semi-supervised constraints.

CHAPTER ONE

INTRODUCTION

Data clustering is an important data mining tool widely used in applications such as pattern recognition, information retrieval, image processing, and robot navigation [1-6]. From the information of how the data are distributed in their feature space, similar entities could be grouped together while dissimilar ones are separated [7]. The clustering outputs are subsequently utilized by a high-level application, which typically possesses additional constraints on the characteristics of the clusters to work on [8]. The previous work focusing on adapting the clustering process according to these top-down requirements has become of a great interest in the recent data clustering research [9, 10]. Different types of auxiliary information have been shown to successfully guide clustering outputs towards the user's intended applications. Clustering algorithms which are able to incorporate auxiliary information into their cluster formation processes is known as semi-supervised clustering [9, 11]. In the previous work, clustering systems have been proposed to improve their output by directly working on partial labeling or pairwise constraints information. It was found that clustering results could be significantly improved, focusing on these specific types of auxiliary information available [10, 12, 13].

However, instead of restricting to one precise form of clustering constraints, this research focuses on constructing a generic framework which is able to function in the situation when the types of the possible clustering constraints are not predetermined and

not observable to the algorithm. In practice, the external clustering objectives and constraints may only be imposed when the clustering system interacts with the environment. In such cases, it is not possible for a clustering adaptation mechanism to be designed a priori. Also, the objective of desired clustering output characteristics may not be explicitly formulated. In some situations [14], it may not even be possible for the user to work out the desired cluster configurations; only an abstract feedback suggesting how well the current clustering output performs with respect to the current task at hand is available as the semi-supervision. As a result, a novel Reinforcement K-Iteration Fast Learning Artificial Neural Network (R-KFLANN) is proposed in order for a clustering algorithm to realize the external requirements by interacting with the environment via a reinforcement process. The R-KFLANN adaptive clustering is inspired by the neural network's cognitive attention and reinforcement learning models. By using only a weak coupling between the clustering adaptation mechanism and the auxiliary information, a broader range of external information formats could be incorporated. While starting off as initially unsupervised, the reinforcement learning paradigm allows R-KFLANN to automatically develop its own adaptive behavior in response to the semi-supervision imposed.

The remainder of this thesis is organized as follows. Chapter 2 provides a detailed review of the related work on different types of data clustering and the advantage of the R-KFLANN framework. After that, the concepts of KFLANN clustering and reinforcement learning are discussed with the formulation of the R-KFLANN system in Chapter 3-5. The experiments on the learning performance of R-KFLANN and the results are presented in Chapter 6. Finally, Chapter 7 concludes the thesis.

CHAPTER TWO

LITERATURE REVIEW

In this chapter, the survey of previous developments in semi-supervised clustering research field will be given. Various approaches taken and techniques proposed in the literature will be categorized according to the abstract information from external sources presented to the clustering algorithm. The concept and motivation of reinforcement clustering and advantages of K-Iteration Fast Learning Artificial Neural Network (R-KFLANN) will be elaborated.

2.1 Unsupervised and semi-supervised clustering

Originally, a clustering task is formulated based on a cluster definition concerning only an unsupervised measure, for example, the within-cluster or between-cluster variance that is frequently used in center-based clustering, the neighborhood density in density-based clustering, or a probability distribution in model-based clustering [7, 15-17]. Recently, attention was also given to a new form of clustering task known as semi-supervised clustering or constrained clustering [10]. In this type of clustering, the objective is no longer based merely on internal measure as in the classical clustering problems. An additional objective of satisfying external constraints given from the

high-level task is also considered. With this aim, additional constraint information is hence to be given in a form of semi-supervision.

As the name suggests, the information obtained from the semi-supervision is in a weaker form, as opposed to the normal supervision where a full labeling of the dataset is used. Figure 2.1 below summarizes different levels of supervisions according to the absence of knowledge.

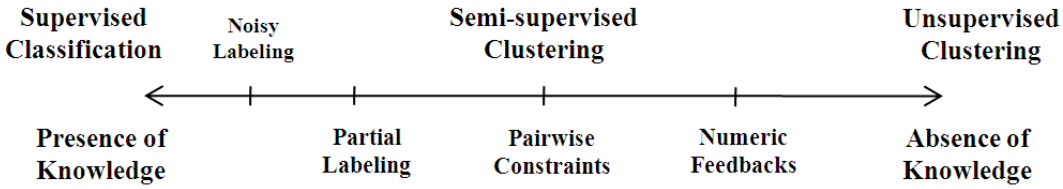


Figure 2.1 Different Levels of Supervisions (Modified from [18])

The degree of supervision includes full labeling, partial labeling, pairwise constraints, numeric feedbacks, spanning to pure unsupervised. From the figure, the supervision of the techniques shown on the right is weaker but more general than those on the left.

2.2 Types of semi-supervisions

2.2.1 Partial labeling

Previous work has proposed to improve clustering results by additional supervision in a form of partial labeling or class memberships. For example, Kamiya et al., 2007 [19], introduced a semi-supervised self-organizing incremental neural network (ssSOINN)

which further divides the subgraph should it contains multiple labels. The work by Bouchachia and Pedrycz, 2006 [18], Cebron and Berthold, 2005 & 2006 [20, 21], and Pedrycz and Waletzky, 1997 [22] provided a variety of extensions to the fuzzy C-Means (FCM) and Fuzzy ISODATA in order to incorporate available labeling information. Apart from these, distance function/metric learning based approaches have also been proposed. for instance, in the work by Eick et al., 2006 [23]; a model-based approach by Sinkkonen and Kaski, 2002 [24]; and a seeding approach such as the Seeded-KMeans by Basu et al., 2002 [13]. The results obtained from these studies have shown that, with an aid of additional labeling information, the clustering results could be guided towards a more appropriate representation of the data.

2.2.2 Pairwise constraint

Alternative to the partial labeling approach, researchers have also proposed frameworks for semi-supervised clustering employing pairwise must-link and cannot-link constraints as supervision. Examples of these include the matrix factorization based semi-supervised algorithm introduced by Wang et al., 2008 [25], BoostCluster algorithm by Liu et al., 2007 [10], Spherical K-Means via feature projection (SCREEN) method by Tang et al., 2007 [26], a locally linear metric adaptation (LLMA) method by Chang and Yeung, 2004 & 2006 [27, 28], a nonparametric Relaxational Metric Adaptation (RMA) method, 2006 [29], MPCK-Means and HMRF-KMeans by Bilenko et al. and Basu et al., 2004 [12, 30], and Constrained K-means (COP-KMEANS) by Wagstaff et al., 2001 [31]. These developments of semi-supervised clustering frameworks using pairwise constraint information provide a great advantage over those using the first approach: the clustering could still be effectively guided even in a situation where class information is not available, or in some cases, not applicable. Moreover, the semi-supervision defined in a pairwise

constraint format is considered to be more general, since any labeling information could still be incorporated in a straight forward manner.

However, as pointed out by Wagstaff, 2007 [32], Cohn et al., 2008 [14] and Basu, 2003 [11], the clustering characteristics desired by the user may not be restricted to only these predefined constraints, but could also involve other types of constraints, for example, cluster-level constraints, including existential and capacity constraints [33, 34], or attribute-level constraints [35]. In real clustering-based applications such as a robot navigation problem by Tse et al., 2008 [6], or direct marketing campaign, retail chain product grouping, and document clustering applications discussed by Banerjee and Ghosh, 2002 [36], the size and skewness of the clusters were considered among other cluster desirability factors, and a certain number of points per cluster were also given as one of the examples of application-dependent constraints. As extensively discussed in the applications as Yahoo! problem [14] where clustering was used to group documents or emails according to users' criteria, in such scenarios, each user could have his own kind of objective functions, and hence many different forms of possible abstract feedback: "This document does not belong here," or "This cluster looks good," and so on. It was also mentioned that the user may not be able to explicitly formulate what they think defines a good clustering, but instead "know it when they see it" [14]. In the solution provided, however, the type of user feedback supported was still limited to only the pairwise constraint format and a general framework for representing user feedback about clusters was yet to be developed [14].

2.2.3 Numeric reinforcement

From the situation of general supervision information discussed above, this research seeks a clustering algorithm which is able to operate on various types of possible

information given from the user. A novel Reinforcement K-Iteration Fast Learning Artificial Neural Network (R-KFLANN) framework proposed in this work is able to perform a semi-supervised learning on weak information of reward or punishment, and hence providing a general framework in accommodating different types of supervision. Another powerful attribute is that, by using an automatic learning of reinforcement, R-KFLANN determines the clustering adaptation strategy through online interactions. As a result, the clustering characteristics suitable for the specific application requirements need not be explicitly articulated a priori by the designer. Such framework is advantageous when the clustering requirements are hidden, not predefined but changing with the applications, or when the action-response is indirectly associated.

2.3 Reinforcement K-Iteration Fast Learning Artificial Neural Network (R-KFLANN)

The Reinforcement KFLANN proposed combines the concepts of reinforcement learning [37] and a K-Iterations Fast Learning Artificial Neural Network (KFLANN) [38-40].

KFLANN is an artificial neural network inspired by a cognitive view of how data is perceived as clusters, which depends on the attentive level of the network. The KFLANN attention model is derived from the vigilance and tolerance concept. In perceiving the grouping of a data sample, the network needs to discern if the sample is “similar” to the cluster representative by evaluating the similarity of the individual feature attributes with respect to the current vigilance level of the network. A feature of two patterns can be considered matched if there is no extensive deviation between them. The permissible extent

of deviation is determined by a tolerance criterion. This unique concept embodies a local clustering strategy that provides a variation of attention levels across different regions within the feature space. The structure of KFLANN is also special in that the second-layer neurons are capable of being grown or pruned dynamically as driven by the data patterns and the attention state of the network. As a result, KFLANN provides the flexibility in adapting the clustering results to non-uniform requirements across the data space.

The proposal of having a local adaptation strategy is similar to that of a semi-supervised clustering work using locally linear metric adaptation (LLMA) by Chang and Yeung, 2004 & 2006 [27, 28]. Metric adaptation methods for supervised and unsupervised clustering were categorized into global and local ones. Given a general clustering objective, the ability to perform a local adaptation is found beneficial as using only a global adaptation of strategy could lead to a suboptimal result, due to the fact that the preferred clustering strategy may not be uniform. On the other hand, the global consistency could still be achieved in the local adaptation by the interactions between clusters after each KFLANN reshuffling process which will smooth the adaptation effect.

The idea of using reinforcement learning technique to guide a clustering process towards the desired objective was studied in a few works previously [41-46]. Oh et al. 2000 [45] used reinforcement learning to modify the degree of fuzziness and the tradeoff between two fuzzy criteria in the fuzzy clustering's objective function. The objective function was then optimized by the clustering algorithm through the Picard iteration. In the work by Bagherjeiran et al., 2005 [41, 42], the data were adaptively projected into a new space by learning the relative weights among different features in the distance metric function, even though, the clustering behavior was not adaptive. However, none of these works allow multiple non-uniform clustering strategies to be applied within the same dataset.

Likas, 1999 [44] was the first to suggest that a general unsupervised clustering system can be built by reworking a competitive learning algorithm at its weight updating step. This updating process can be made generalized by using a reinforcement learning system to enforce different clustering strategies. Depending on the reward function used, the competitive learning reduced to different well-known unsupervised clustering algorithms such as FSCL, RPCL, Maximum-entropy clustering, and SOM. The reinforcement learning was accomplished using stochastic Bernoulli units which performed hill-climbing in solving an immediate reinforcement learning problem, i.e., not considering the future reward. Subsequently, Barbakh and Fyfe, 2007 & 2009 [43, 46], further explored additional types of internal reward functions that could be used for the reinforcement clustering proposed. However, the semi-supervised clustering problem of addressing the high-level user's constraints and objectives has not been investigated.

This work weaves the elements and concepts of R-KFLANN into performing semi-supervised clustering. The R-KFLANN learning theory and its interaction process with the external feedback will be discussed in the chapters that follow.

CHAPTER THREE

K-ITERATION FAST LEARNING ARTIFICIAL NEURAL NETWORK (KFLANN)

In this chapter, the fundamental concept and properties of the K-Iterations Fast Learning Artificial Neural Network will be discussed. KFLANN is an artificial neural network consisting of two layers of neurons with full connection between the two layers. The structure of KFLANN is shown in Figure 3.1.

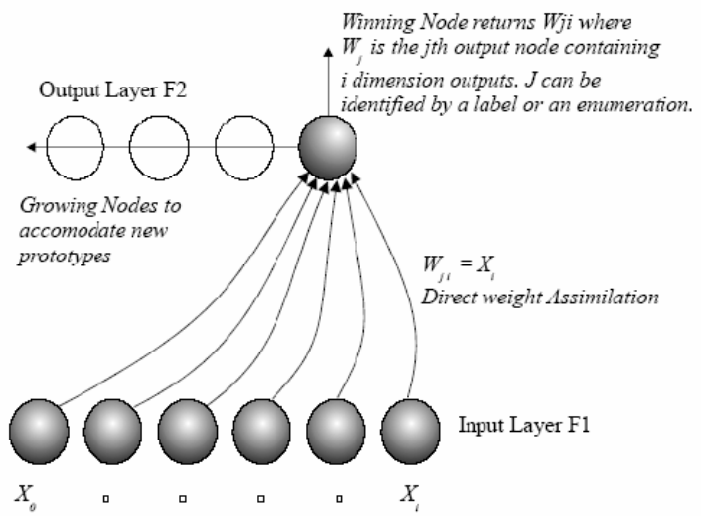


Figure 3.1 KFLANN conceptual structure (adapted from [40])

The KFLANN model evolved from the concept of Fast Learning Artificial Neural Network (FLANN) [47] which was inspired by Adaptive Resonance Theory (ART) [48] and Kohonen's self-organizing map. However, FLANN encountered a problem of data

presentation sequence sensitivity (DPSS) [40] in which the cluster formation is highly dependent on the order of data sequence in the dataset. As a result, KFLANN was proposed, enforcing FLANN with a K-Means-based strategy such that better consistency and stability of the resulting clusters could be achieved.

From a data mining perspective, KFLANN is used for prototype-based (also called center-based) clustering applications. In prototype-based clustering, the clusters are formed in a way that the prevailing input sample ressembles the cluster whose prototype/ cluster representative/examplar matches the sample better than any other existing prototypes. A cluster prototype is generally the centroid i.e. the mean, or the medoid of the cluster. The signature of each cluster's prototype is learned and stored by the network in the synaptic weight vectors connecting to the second-layer neuron representing each cluster.

3.1 KFLANN unsupervised clustering process

The learning of KFLANN is unsupervised. The formation and updating of the cluster prototypes are merely guided by the distribution of the data patterns and the attention state of the network. Since the KFLANN clustering process holds a tight relationship with the human attention model, it has been found useful in various cognitive applications, e.g., cognitive robot navigation and cognitive visual understanding [6].

In a perception process, an object is considered as having a chance of belonging to a cluster only if enough features resemble those of the cluster prototype. For instance, if an object sample has too many features deviating from the features of the cluster C_i, the cluster C_i is eliminated from the possible matching candidates. The generalization (dichotomized as strictness) of this matching process is determined by the network's attentiveness level

when perceiving the object. Following this matching process, the perceived object sample is grouped into one of the matching cluster prototypes which display the strongest similarity to the object sample in the data space.

The matching process provides the network a means to determine if a new object category is encountered. In a situation when there is no suitable matching prototype within the existing repository, the network expands its current knowledge pool by growing its second-layer neurons in order to accommodate the newly discovered object type. This dynamic network structure is one advantage of KFLANN over static-structured neural network models such as Kohonen self-organizing map (SOM) since the number of types or clusters of the objects that the network will learn is not restricted to a predetermined setting, but instead, flexes according to the data pattern presented to the network. In order to mimic the human cognitive attentive model, the KFLANN can be set to exhibit low attentiveness (dichotomically higher generalization) when the training samples are perceived. This results in fewer object types learnt as generalization causes objects to be perceived as similar.

The KFLANN attentiveness level is a function of two network parameters: *tolerance* and *vigilance*. The tolerance vector $\bar{\delta}$ defines the acceptable amount of deviation allowed in each input feature from the prototype feature stored in the memory. Options are provided for the tolerance values to be manually or heuristically adjusted through the statistical property of the data. For example, in a robot mapping application [6], the tolerances can be defined as a function of the standard deviation of the environmental expanse. Thus when operating in a confined space of a laboratory, the cognitive map tolerance can be in small units in the order of meters. However if the space is extended to city-wide navigational scenarios, these statistical quantities automatically extend to kilometers, scaling the clustering tolerance of the system and pegging it to the scenario.

This tolerance adjustment, therefore, provides a unified means to achieve the appropriate level of stringency for a robust cognitive perception process.

The vigilance $\rho \in [0,1]$ specifies the minimum number of matched features in order for a pattern to be considered as similar to the prototype stored in the memory:

$$\rho = \frac{f_{match}}{d} \tag{3.1}$$

The function of the vigilance value is similar to the tolerance in that it provides a control over generalization of the matching process. The higher vigilance translates to the higher attentiveness and hence a more stringent scope before the output neurons to be activated. When the network vigilance approaches one, only a few noisy features will be perceived as a significant disparity during prototype comparisons, hence providing a higher probability for the network to generate new clusters. However if the vigilance is set at lower levels, generalizations dominate the network and noise is tolerated, making it easier for small disparate features of an object to be recognized as a previously learnt pattern.

The KFLANN algorithm and its behavior with regard to these two parameters will be discussed next.

3.2 KFLANN algorithm

Given an input dataset $\mathfrak{D} \equiv \left\{ \bar{x}_l \in \mathbb{R}^d \right\}$, where $n \equiv \left| \mathfrak{D} \right|$ denotes the number of samples and d is the dimension of the data space, the KFLANN produces the output clusters $C_j \subseteq \mathbb{R}^d$ under the conditions: $C_i \cap C_{j \neq i} = \varnothing$ and $\mathfrak{D} = \bigcup_{j=1}^{k} C_j$, such that $\left(C_1, C_2, ..., C_k \right)$ is a

mutually exclusive and exhaustive partition of $\mathfrak{D}$. During the learning stage of the network, the training samples are presented through the input-layer neurons, each corresponding to each feature of the input vector. Starting with no prior knowledge, a set of prototypes $W \equiv \left\{ \overline{w}_j \in \mathbb{R}^d \right\}$ stored in the memory is empty, with the number of clusters $k \equiv |W| = 0$. When each training sample l is presented, the network determines whether a new cluster formation is needed, according to the matching criterion:

$$\frac{\sum_{i=1}^{d} D[\delta_i^2 - (w_{ji} - x_{li})^2]}{d} \geq \rho, \text{ where} \tag{3.2}$$

$$D[a] = \begin{cases} 1; & a > 0 \\ 0; & a \leq 0 \end{cases}, \tag{3.3}$$

for all $\overline{w}_j$ attained previously. If no matching is found, the network grows its second layer neurons to accommodate the new knowledge creating a new cluster prototype $\overline{w}_j = \overline{x}_l$ .Using the Euclidean distance as a based metric, the input $\overline{x}_l$ is then assigned to the nearest matching cluster candidate from:

$$winner_j = \arg\min_j [\sum_{i=1}^{d} (w_{ji} - x_{li})^2]. \tag{3.4}$$

In a similar way to human attention, the KFLANN can reduce its processing time in recognizing an input pattern by screening out the cluster prototype that is obviously not likely to win in the nearest neighbor competition. Instead of calculating the exact distance and a nearest winner comparison from all existing prototypes, it is possible to terminate the calculation $(w_{ji} - x_{li})^2$ for all dimensions i prematurely when the unmatched feature count exceeds $\lfloor (1-\rho) \cdot d \rfloor$. Hence, $\overline{w}_j$ is then removed from the candidate list. This is most

useful when the number of prototypes or the number of features is large but in fact there are only a few potential candidates for the winner calculation.

Once all data $\bar{x}_l \in \mathfrak{D}$ are assigned, the set of cluster representatives W is updated by recalculating the centroids of all clusters: $\bar{w}_j = \mu(C_j)$. In order to obtain a stable representation of the dataset, the data is replayed for many epochs reorganizing the previously learned prototype to the top of the data sequence. This reorganization of the data is called reshuffling, inspired by the rapid eye movement stage of the dreaming process evident the human brain. It is believed that this process sorts out the learned information before storing it into the long-term memory [49]. This reshuffling process was designed with an objective of providing a means for consistent clustering regardless of the data presentation sequence (DPS) changes [38, 39].

When KFLANN has finished its learning process, the KFLANN algorithm $\mathscr{A}$ completely maps the original dataset onto the set of prototypes $\mathscr{A}: \mathfrak{D} \to W$. The subsequent input data presented to the network will be recognized by the second layer neuron storing the corresponding prototypes in the same way as in the learning period discussed above. The KFLANN algorithm is summarized here in Figure. 3.2.

The KFLANN Algorithm

Step 1 *NETWORK INITIALIZATION*
Given a dataset of n samples and d features, Set the vigilance. Initialize l = 1; calculate σ_i, *for all d features in the dataset, assign* $\delta_i = \sigma_i$, *and initialize* $W = \varnothing$.

Step 2 *Present the* l^{th} *pattern* $\bar{x}_l$ *to the input layer neurons. If the network does not have any output node yet, GOTO Step 8.*

Step 3 *Determine all possible output node matches using*

$$\frac{\sum_{i=1}^{d} D[\delta_i^2 - (w_{ji} - x_{li})^2]}{d} \geq \rho,$$

$$D[m] = \begin{cases} 1; & m > 0 \\ 0; & m \leq 0 \end{cases},$$

D is the discriminant function imposed on the difference

between the tolerance and the Euclidean distance.

Step 4 Maintain a list of all matches that fulfilled Step 3. Using the winner take all criteria below, determine a single winner cluster from the matched list.

$$winner_j = \arg\min_j [\sum_{i=1}^{d} (w_{ji} - x_{li})^2]$$

Step 5 If winner is found, Assign the current pattern to the winning cluster. Else, Goto Step 8.

Step 6 If $l = n$, compute the mean values for each cluster and identify the cluster members closest to the means as the cluster centroids.

If the collection of centroids is consistent through the epochs, terminate.

Else, GOTO Step 7.

Else, increment l by 1 and GOTO Step 2.

Step 7 Reshuffle all centroids to the top of the data list for the next epoch. The data sample size remains constant. Set $l = 0$. Reset all output nodes and GOTO Step 2.

Step 8 Create a new output neuron j. Attach the prototype $\bar{w}_j = \bar{x}_l$ to the new neuron, assign the current pattern $\bar{x}_l$ to the cluster C_j created and GOTO Step 6.

ρ is the vigilance, σ_i and δ_i are the standard deviation and the tolerance for i^{th} feature of the input space, $\bar{w}_j$ is the stored weight vector and $\bar{x}_l$ is the current pattern presented.

Figure 3.2 KFLANN Algorithm (Adapted from [40])

3.3 KFLANN attention state and clustering behavior

Since the KFLANN perception process relies on its attention state, an appropriate adjustment in the attention parameters could help control the clustering behavior of the network to suit the requirements of the application.

The proper range of the vigilance is $\rho \in (0,1]$ for the learning phase of the network, and $\rho \in [0,1]$ for usage phase. When $\rho = 0$, KFLANN becomes most relaxed and will not learn any new class of objects. It idly assigns the object presented to the nearest centroid without creating any new clusters in the second network layer. The data space partitioning of KFLANN hence reduces to the Voronoi tessellation. Under normal circumstances, the

vigilance $\rho \in [0.5,1]$ is used. The effect of this setting on the tolerance will be explained.

From the concept of the tolerance, KFLANN uses statistical insights in estimating the tolerance setting. Firstly, the deviation in an input pattern from the cluster representative is a result of properties such as noise or a natural diversity among the objects within the class which can be predicted from the class distribution statistics. If the distributions of all underlying classes in the dataset are known, the tolerance vector $\bar{\delta}$ can be readily specified based on the standard deviation of each distribution as follows:

Consider the distribution of only one feature *i* from samples within a class C_j for the case of a Gaussian distribution, as shown in Figure 3.3. The empirical rule [50] states that at least 99.7 percent of the data should fall within three standard deviations away from the mean. This implies that any sample lying outside $3\sigma_{C_j,i}$ away from w_{ji} is not likely to belong to this Gaussian distribution underlying C_j.

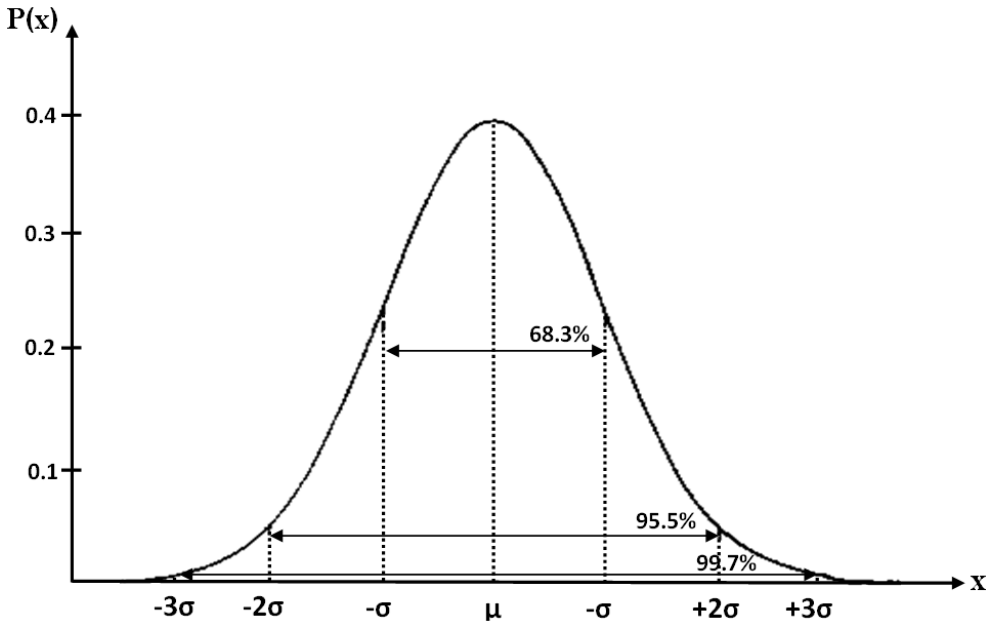


Figure 3.3 Percentage of Data Falling within Each Standard Deviation Range from the Mean under Gaussian Distribution

In a general case of arbitrary distribution, Chebyshev's inequality [51] can be applied, giving a looser bound than in a Gaussian case. Chebychev's inequality states that no more than $\frac{1}{k^2}$ of the data are k standard deviations away or further from the mean, in other words:

$$\Pr\left(\left|x_{li}-\mu_{C_j,i}\right| \geq k\sigma_{C_j,i} \,\middle|\, \bar{x}_l \in C_j\right) \leq \frac{1}{k^2}. \tag{3.5}$$

Therefore, choosing $k=6$, the chance of a sample $\bar{x}_l \in C_j$ lying at least six standard deviations away from the mean is only 2.8 percent or less. This signifies that a new cluster formation is expected, i.e. $\bar{x}_l \notin C_j$, if a pattern beyond the six standard deviation range is observed, since, in general, from Bayesian principle:

$$\Pr\left(\bar{x}_l \in C_j \,\middle|\, \left|x_{li}-\mu_{C_j,i}\right| \geq k\sigma_{C_j,i}\right) \propto \Pr\left(\left|x_{li}-\mu_{C_j,i}\right| \geq k\sigma_{C_j,i} \,\middle|\, \bar{x}_l \in C_j\right), \tag{3.6}$$

assuming uniform probability distributions for the priors: $\Pr\left(\bar{x}_l \in C_j\right)$ and $\Pr\left(\left|x_{li}-\mu_{C_j,i}\right| \geq k\sigma_{C_j,i}\right)$. As a result, the maximum span of clusters can be set as, for example, in the Figure 3.4 illustrated below:

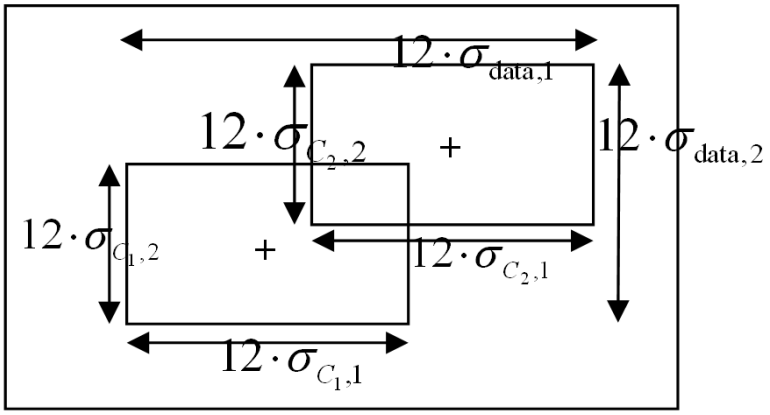


Figure 3.4 Tolerable Deviation before Generating a New Cluster

With the two underlying distributions C_1 and C_2 shown in the 2D space, each C_j has standard deviations $\sigma_{C_j,1}$ and $\sigma_{C_j,2}$ along dimension 1 and 2 respectively. A setting of the tolerance at $\delta_{C_j,i} = 6\sigma_{C_j,i}$ will signal a cluster expansion when a pattern outside the existing class emerges, where $\sigma_{C_j,i} = \sqrt{\frac{\sum\left(x_{li} - w_{ji}\right)^2}{n_j - 1}}; \bar{x}_l \in C_j, n_j = \left|C_j\right|$. In the original KFLANN, the tolerance is estimated from the order of the dataset's standard deviation σ_i as:

$$\delta_i = \sigma_i \cdot \psi , \tag{3.7}$$

with ψ being a cluster scaling factor used to correlate the cluster standard deviation with the dataset's expanse. This heuristic has been found useful in an actual application [6], since the standard deviation of each cluster normally has the same order as that of the dataset, e.g., in a meter's range or in a kilometer's range, etc. The factor ψ is to fine tune the tolerance further to an exact value. For example, assuming that all clusters C_j share the same standard deviations $\sigma_{C_j,i}$, then it is possible to estimate the proper scaling factor from $\delta_i = 6\sigma_{C_j,i} = \sigma_i \cdot \psi$; that is $\psi = {6\sigma_{C_j,i}}/{\sigma_i} = 1$.

Nonetheless, in a usual case that no prior information about the natural underlying distribution is available, the chicken-and-egg problem is encountered, since the tolerance needs to be determined before actually obtaining the clustering of C_j. Therefore, an interactive extension of KFLANN proposed in this work will solve this problem by repetitively adjusting the clustering of C_j and the scaling factor ψ simultaneously with the guidance from the reinforcement learning.

Another extension to be made in KFLANN is a local cluster size adjustment.

KFLANN assumes that all clusters share the same characteristic in terms of $\sigma_{C_j,i}$. The best cluster formation, however, could be achieved only if KFLANN can truly distinguish all clusters from one another correctly, which may not happen in a case of clusters with different sizes. This is a typical problem encountered in many prototype-based clustering algorithms like SOM or K-Means [8]. As shown in Figure 3.5, if merely a global tolerance value δ_i are used, the correct partitioning of C_1, C_2 and C_3 will not be achieved. For example, if the tolerance is gauged based on the spread of C_3, then C_1 and C_2 will be merged together and only two clusters will be perceived by KFLANN. However, if the tolerance is gauged based on C_1 and C_2, then C_3 will be treated as two separate clusters, resulting in four clusters perceived. In other words, in this specific scenario, KFLANN needs a higher attentiveness when perceiving C_1 and C_2 than C_3 since they are more obscure.

This variation in the clustering strategy requirement is a synonymous model of how humans could adjust attention levels to sift finer information from the more generalized initial assessment of information. The clustering strategy to be pursued by KFLANN changes when applied in different applications.

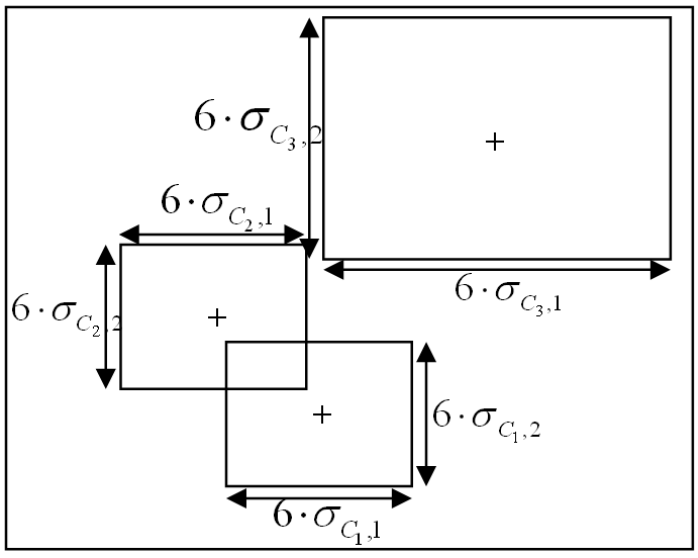


Figure 3.5 A Dataset with Unbalanced Underlying Distributions

For example, low attention is required in "separable" regions but high attention in "non-separable" regions (for classification tasks), low attention in "uninteresting" places and high attention in "dangerous" places (for robot navigation task [6]) which will be a source of discussion in a later chapter. Hence, the original KFLANN is modified in this work allowing local adaptations of the clustering parameter in order to reflect an accurate partitioning when the data possesses non-homogeneous properties. The local clustering adjustments will then propagate in the data space by the interactions among neighboring clusters.

In order to affect the cluster formation strategy in each region, a local scaling factor ψ_j is attached to each C_j, forming a pair $(\overline{w}_j, \psi_j)$ and a local tolerance vector $\overline{\delta}_j = \psi_j \cdot \overline{\sigma}$. When perceiving an input pattern $\overline{x}_l$, KFLANN recalls from experiences and estimates the importance of that pattern selecting the appropriate attention level to be used. Reinforcement learning therefore serves the purpose of learning from the experiences and determines the attention level associated with the distribution characteristic of each type of data in a particular region. The method of how an adaptive reinforcement KFLANN can be done is provided in the following chapters.

CHAPTER FOUR

REINFORCEMENT LEARNING

In this chapter, the background of reinforcement learning is provided beginning with a formal problem formulation. A general problem formulated as Markov Decision Process (MDP) problem is described in Section 4.1. After that, methods proposed in order to solve the MDP problem are introduced in the Section 4.2 to 4.3, namely, Dynamic Programming (DP) and the concept of Temporal Difference (TD) approaches. Subsequently, a model-free method using Q-Learning algorithm for solving the DP problem is given in the Section 4.4.

4.1 Problem formulation: Markov Decision Process

A reinforcement learning problem is defined as a problem of finding an optimal policy which maps states to actions such that the return received following the policy is maximized. It is different from supervised learning problems such that input-output pair supervision in terms of the correct action to be executed in each state is not available. The mathematical formulations of the reinforcement learning elements are provided here.

First of all, the main components of reinforcement learning are the reinforcement learning agent or just *agent*, and the *environment*. The agent is capable of performing

different *actions* $a(t)$ on the environment, while the environment is described by its *state* $s(t)$ which is altered by the agent's action. The state is considered a random variable whose transition follows an underlying a stochastic model called a *state transition model*. Therefore, the environment could be described by the model $T(s',s,a)$ as:

$$T(s(t+1),s(t),a(t)) \equiv \Pr(s(t+1) \mid s(t),a(t)), \tag{4.1}$$

with

$$\Pr(s(t+1) \mid s(t),a(t),s(t-1),a(t-1),...,s(0),a(0)) = \Pr(s(t+1) \mid s(t),a(t)). \tag{4.2}$$

From the equation above, the probability predicting the next state s(t+1) is only governed by the current state s(t) and action a(t), thus, the *Markov property* holds for this process. Also, the state transition is *stationary*, i.e., the model does not change with time:

$$T(s(t+1)=s',s(t)=s,a(t)=a) = T(s(t)=s',s(t-1)=s,a(t-1)=a) = ...$$
$$\forall s',s \in S, a \in A \tag{4.3}$$

After transiting to a new state, the agent receives a numerical feedback from the environment in terms of a *reward* which indicates the desirability of the state reached. From the biological system point of view, rewards and punishments (negative rewards) could be related to pleasure and pain [37] which motivates living beings. From the application point of view, the reward serves as a communication channel between the user and the agent to feedback on the performance. The reward r(t) is also a random variable. In making a decision, the expected value of the reward is used:

$$\Re(s(t+1), s(t), a(t)) \equiv E[r(t+1) \mid s(t+1), s(t), a(t)] \tag{4.4}$$

The system of a state space S, an action space A, a transition model T, and a reward function r described above constitutes [52] a *Markov Decision Process (MDP)* formulated by Bellman [53]. An MDP is considered as an extension of a Markov chain with additions of a choice of actions allowed to be chosen by the agent and rewards as a motivation in choosing an action.

The objective is to find a *policy* in which, starting from any state given, by following it, the maximum *return* based on *total discounted reward* is obtained. A policy $\pi(s)$ is a mapping from states to actions guiding the decision by which action is performed in each state. The return $R(t)$ starting from time t onwards is defined as follows:

$$R(t) \equiv r(t+1) + \gamma \cdot r(t+2) + \gamma^2 \cdot r(t+3) + ... + \gamma^n \cdot r(t+n+1) + ... + \gamma^{\mathrm{T}} \cdot r(t+\mathrm{T}+1), \tag{4.5}$$

where $\gamma \in [0,1]$ is a *discount factor* parameter indicating the relative importance between immediate reward and future rewards. When γ is zero, the immediate reward becomes the sole component of the objective function. As γ approaches one, both components reflecting the immediate reward and future rewards become relevant.

When T is finite, the task ends after a finite number of time step (*finite horizon*) and is called an *episodic task* where each episode ends with a *terminal state* $s(t+T+1)$. In some applications, the task continues infinitely with $T = \infty$ (*infinite horizon*). In such case, the task is called a *continuing task* and the discount factor parameter $\gamma < 1$ is usually required in order to obtain a finite value of total discounted reward, provided that the sequence $\{r(t+k+1)\}$ is bounded. In short, it is possible to have $\gamma = 1$ or $T = \infty$ but not both, unless

the series $\sum_{k=0}^{\infty} r(t+k+1)$ is presumed convergent. However, the actual reward sequence is in fact never known beforehand; only the parameter γ is known and controllable.

4.2 Solutions to the Markov Decision Process

In order to proceed, it is imperative to verify if there exists a single policy that will be optimal over all states, and whether the optimality could be reached by a policy that is stationary, i.e., a policy that does not change over time. In answering this, Ross, in 1983 [54], has proven that for every MDP there will be a stationary optimal policy. This holds by the reason of the Markov property of the process in which the current state provides completely all necessary information for a prediction of future, and hence, the same decision will always be optimal given the same state perceived.

The solution to the problem starts with a definition of an evaluation measure of a policy. A policy π is evaluated using the *expected return* following that policy, starting from a state s:

$$V^{\pi}(s) \equiv E[R(t) \mid \pi, s(t) = s] = E[\sum_{k=0}^{\infty} \gamma^{k} r(t+k+1) \mid \pi, s(t) = s] \tag{4.6}$$

This $V^{\pi}(s)$ is known as a *state-value function* for policy π. It indicates the viability of the policy π judging from the return expected following this policy. In the case where the policy π is optimal, the state-value function also reflects the goodness of being in that state, based on the return expected.

4.2.1 Dynamic Programming (DP)

The question is how to calculate the expected return for any policy $\pi(s)$ given, and how the optimal policy based on the knowledge of expected return can be produced. Some classical methods that solve these problems can be found under the field of Dynamic Programming. One important assumption taken in Dynamic Programming is that the models of the state transition $T(s',s,a)$ and expected reward $\Re(s',s,a)$ are fully known to the agent. Therefore, the state-value function for a deterministic policy π(s) can be obtained from:

$$\begin{aligned} V^{\pi}(s) &= E[\sum_{k=0}^{\infty}\gamma^{k} r(t+k+1) \mid \pi, s(t)=s] \\ &= E[r(t+1)+\gamma\sum_{k=1}^{\infty}\gamma^{k-1} r(t+k+1) \mid \pi, s(t)=s] \\ &= E[r(t+1)]+\gamma E[\sum_{k=0}^{\infty}\gamma^{k} r(t+k+1) \mid \pi, s(t)=s'] \\ &= \Re(s',s,\pi(s))+\gamma\sum_{s'} T(s',s,\pi(s))\cdot V^{\pi}(s'), \end{aligned} \tag{4.7}$$

by the definitions of $\Re(s',s,a)$ and $V_{\pi}(s)$, and the Markov property.

Writing in the most general form, a stochastic policy with $\pi(s,a)$ representing the probability of taking the action a in the state s may be used:

$$V^{\pi}(s) = \sum_{a}\pi(s,a)\cdot\{\Re(s',s,a)+\gamma E[\sum_{k=0}^{\infty}\gamma^{k} r(t+k+1) \mid \pi, s(t)=s]\} \tag{4.8}$$

This is also known as the *Bellman equation*. The beauty of Bellman equation lies in a fact that the Markov property provides a means to calculate $V^{\pi}(s)$ without having to trace

all possible sequences of next states and their rewards $r(t+k+1)$, where the number of possible sequences grows exponentially with k.

Based on Bellman equation, one way to calculate the state-value function is by computing $V^{\pi}(s)$ iteratively for all states belonging to the state space, $s \in S$. The value updating of $V^{\pi}(s)$ is based on all the successor states' values $V^{\pi}(s')$ according to the following:

$$V^{\pi}(s) = \sum_{a}\left[\pi(s,a)\sum_{s'}T(s',s,a)\cdot\left(\Re(s',s,a)+\gamma V(s')\right)\right] \tag{4.9}$$

The updating repeats until all $V^{\pi}(s)$ converge, i.e., neighbouring states' values satisfies Bellman equation or an acceptable inconsistency is achieved. This method is called *iterative policy evaluation*. The algorithm for iterative policy evaluation is given below:

The Iterative Policy Evaluation Algorithm

Step 1	Set a threshold θ. Initialize the state-value function $V^{\pi}(s) = 0, \forall s \in S$.
Step 2	Update the state-value for each $s \in S$: $$V^{\pi}(s) = \sum_{a}\left[\pi(s,a)\sum_{s'}T(s',s,a)\cdot[\Re(s',s,a)+\gamma V(s')]\right]$$
Step 3	If the changes in $V^{\pi}(s)$ of all states $s \in S$ are smaller than the threshold θ, terminate. Else, GOTO Step 2.

Figure 4.1 iterative policy evaluation

it is then possible to evaluate and attempt to improve any given policy $\pi(s)$. The two dynamic programming approaches that can be used for determining the optimal policy, namely, *policy iteration* and *value iteration*, are given next.

4.2.1.1 Policy iteration

The concept of policy iteration lies in the attempt to improve policy π by examining each step-wise decision $\pi(s)$. To illustrate this, consider a scenario of beginning in a state s. The suggested decision $\pi(s)$ is revised by unraveling the state-value function $V^{\pi}(s)$ with respect to different cases of actions a that could be taken in place of the current $\pi(s)$. From this reason, an *action-value function* $Q^{\pi}(s,a)$ of a deterministic policy π is defined as the expected return starting from state s, taking the action a, and thereafter following policy π:

$$Q^{\pi}(s,a) \equiv E[R(t) \mid s(t)=s, a(t)=a, a(t'>t)=\pi(s(t'))] \tag{4.10}$$

$$= E[\sum_{k=0}^{\infty} \gamma^{k} r(t+k+1) \mid s(t)=s, a(t)=a, a(t'>t)=\pi(s(t'))]$$

$$= \Re(s',s,a) + \gamma \sum_{s'} T(s',s,a) \cdot V^{\pi}(s'). \tag{4.11}$$

Recalling the Bellman equation for the state-value:

$$V^{\pi}(s) = \Re(s',s,\pi(s)) + \gamma \sum_{s'} T(s',s,\pi(s)) \cdot V^{\pi}(s'),$$

the action-value function provides a means for evaluating different choices of actions a to be taken at each step. Consequently, the decision $\pi(s)$ for the current state s can then be augmented as follow:

$$\pi'(s) = \arg\max_{a}(Q^{\pi}(s,a)) \tag{4.12}$$

Therefore, the expected value starting from the state s improves and, as a result, the overall expected values also increase for all other states:

$$Q^{\pi}(s,\pi'(s)) \geq Q^{\pi}(s,\pi(s)) = V^{\pi}(s), \text{ and}$$

$$V^{\pi'}(s) \geq V^{\pi}(s), \ \forall s \in S .$$

This process of policy updating is repeated until no single step's decision can be further improved; and thus, the optimal policy π^* and its corresponding action values $Q^*(s,a)$ are attained.

To summarize the policy iteration, the iteration between two main steps: *policy evaluation* and *policy improvement* is taken [55]:

1. Policy evaluation: Compute the values of all states based on the current policy π according to Bellman equation (4.7):

$$V^{\pi}(s) = \Re(s',s,\pi(s)) + \gamma \sum_{s'} T(s',s,\pi(s)) \cdot V^{\pi}(s')$$

2. Policy improvement: Improve the policy according to the current values according to equation (4.11) and (4.12):

$$Q^{\pi}(s,a) = \Re(s',s,a) + \gamma \sum_{s'} T(s',s,a) \cdot V^{\pi}(s'), \text{ and}$$

$$\pi'(s) = \arg\max_{a}(Q^{\pi}(s,a)) .$$

4.2.1.2 Value iteration

In every cycle of policy iteration, a complete run of policy evaluation has to be performed until $V^{\pi}(s)$ converges for all states $s \in S$; therefore, the process naturally requires long time spans. It is possible [37] that the policy evaluation is truncated after only one sweep. This leads to the value iteration concept. In each value iteration sweep, only one

sweep of policy evaluation and one sweep of policy improvement are performed. Therefore, combining the policy evaluation and policy improvement yields:

$$Q(s,a) = \Re(s',s,a) + \gamma \sum_{s'} T(s',s,a) \cdot \max_{a'} Q(s',a') \tag{4.13}$$

In comparison with the action-value defined previously in policy iteration:

$$Q^{\pi}(s,a) = \Re(s',s,a) + \gamma \sum_{s'} T(s',s,a) \cdot V^{\pi}(s'),$$

the action-value $Q(s,a)$ is now defined without any fixed policy associated with it, but instead, imposing the condition of choosing the action a' with the highest expected return as a subsequent decisions: $\pi(s') = \arg\max_{a'}(Q(s',a'))$ for one step of evaluation.

Repeating the value updating in equation (4.13), $Q(s,a)$ will converge to the same optimal $Q^*(s,a)$ as those obtained from policy iteration. Finally, once $Q^*(s,a)$ is known, the optimal policy π^* can be derived easily in a similar way as equation (4.12) before:

$$\pi^*(s) = \arg\max_{a}(Q^*(s,a)) \tag{4.14}$$

4.2.2 Temporal Difference (TD)

As mentioned above, policy iteration and value iteration methods require the state transition model $T(s',s,a)$ and expected reward $\Re(s',s,a)$ to be known. The TD method allows the agent to learn the value function without any need for such prior information of the environment. By using the information of the *experience tuple* (s,a,s',r') containing

the state transition, action, and the reward received from the interactions with the environment, the agent adjusts the estimated values towards the target value. Two TD methods, namely, actor-critic learning and Q-learning are discussed in the following sections.

4.2.2.1 Actor-critic learning

Firstly, to illustrate the TD concept, recall the expected return $V^{\pi}(s)$ calculation according to Bellman equation:

$$V^{\pi}(s) = \Re(s', s, \pi(s)) + \gamma \sum_{s'} T(s', s, \pi(s)) \cdot V^{\pi}(s')$$

The expected return $V^{\pi}(s)$ is to be estimated using the agent's current observation on the reward received r' and the actual state transition $s \to s'$ instead of their expected value and probability mass function, $\Re(s', s, \pi(s))$ and $T(s', s, \pi(s))$ as in the following equation:

$$V^{\pi}(s) = r' + \gamma \cdot V^{\pi}(s') \tag{4.15}$$

This estimation of $V^{\pi}(s)$ can be improved after each transition is observed by adjusting $V^{\pi}(s)$ according to the Temporal-Difference error between the new and previous estimation:

$$\begin{aligned} \text{Temporal-Difference error} &= V^{\pi}{}_{\text{new}}(s) - V^{\pi}{}_{\text{old}}(s) \\ &= [r' + \gamma \cdot V^{\pi}(s')] - V^{\pi}(s), \end{aligned} \tag{4.16}$$

and $$\Delta V^{\pi}(s) = \alpha \cdot [\text{Temporal-Difference error}], \tag{4.17}$$

therefore, $$V^{\pi}(s) = V^{\pi}(s) + \Delta V^{\pi}(s). \tag{4.18}$$

Given an appropriate *learning rate* α, the estimation of $V^{\pi}(s)$ is guaranteed to converge to the correct value [52]. This is known as TD(0) algorithm which can be used to perform a policy evaluation. Consequently, the policy iteration process can be attained by performing the policy improvement step as done previously. This TD implementation of policy iteration is known as an *actor-critic learning* [56] where the actor takes the policy improvement step and the critic takes a policy evaluation. The actor-critic learning is considered an *on-policy* method [37], i.e. the learning is attached with a policy π, and the process focuses on improving this policy.

4.2.2.2 Q-Learning

In contrast to actor-critic learning, the *off-policy* Q-learning was proposed by Watkins [54]. It is considered the simplest, most efficient and most popular [52] among other forms of reinforcement learning techniques. The advantages of Q-learning include the following: First, being one of the TD methods, it does not require a model of the environment's dynamics. Second, it is better than the actor-critic method in that it does not encounter the balancing problem between the actor and the critic components' learning rates in order to ensure that both converge simultaneously [52]. Finally, the convergence of the Q-values to the optimal values is not affected by the strategy taken by the agent in choosing the actions during the learning phase, provided that all state-action pairs are visited sufficiently [52]. The idea of Q-learning is as follows:

In relation to actor-critic learning, a TD implementation of policy iteration, Q-learning is regarded as a TD version of value iteration. First, recalling the value iteration, the expected return $Q(s,a)$ for a given state-action pair (s,a) is repeatedly updated to towards the optimal $Q^*(s,a)$ by:

$$Q(s,a) = \Re(s',s,a) + \gamma \sum_{s'} T(s',s,a) \cdot \max_{a'} Q(s',a')$$

In the same way as in actor-critic learning, the optimal expected return $Q^*(s,a)$ is to be estimated using the agent's observation of the reward received r' and the state transition $s \to s'$, instead of relying on the expected value and probability mass function information $\Re(s',s,\pi(s))$ and $T(s',s,\pi(s))$:

$$Q(s,a) = r' + \gamma \max_{a'} Q(s',a') \tag{4.19}$$

The estimation of $Q(s,a)$ could therefore be improved after each transition is observed by updating it according to the Temporal-Difference error between the new and previous estimation $Q_{\text{new}}(s,a)$ and $Q_{\text{old}}(s,a)$ with a learning rate α as:

$$\begin{aligned} \text{Temporal-Difference error} &= Q_{\text{new}}(s,a) - Q_{\text{old}}(s,a) \\ &= [r' + \gamma \max_{a'} Q(s',a')] - Q(s,a), \end{aligned} \tag{4.20}$$

$$\text{and} \qquad \Delta Q(s,a) = \alpha \cdot [\text{Temporal-Difference error}], \tag{4.21}$$

$$\text{therefore} \qquad Q(s,a) = Q(s,a) + \Delta Q(s,a). \tag{4.22}$$

As a result, the optimal action-value $Q^*(s,a)$ can be found. The optimal policy

$\pi^*(s)$ is then the policy which takes the action with the highest value for each state visited:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

The Q-learning algorithm is summarized in the Figure 4.2.

The Q-learning Algorithm

Step 1	Initialize the action-value function $Q(s,a)$.
Step 2	Check the stopping condition. If satisfied, terminate. Else, start a new episode.
Step 3	Perform an action a based on the current state s and the action-value function $Q(s,a)$.
Step 4	Receive new reinforcement r' and observe the new state s' reached.
Step 5	Update the Q-value of the previous state-action pair from the transition observed: $$Q(s,a) = Q(s,a) + \alpha\left[r' + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$
Step 6	Update the current state $s = s'$. If a terminal state is reached, GOTO Step 2. Else, GOTO Step 3.

Figure 4.2 Q-learning Algorithm

CHAPTER FIVE

REINFORCEMENT K-ITERATION FAST LEARNING ARTIFICIAL NEURAL NETWORK (R-KFLANN)

As presented earlier in the Section 3.3, the KFLANN attention and clustering behavior was discussed with respect to the underlying characteristics of the data. It was then modified to suit non-uniform clustering characteristics. With the concepts derived from the modified unsupervised KFLANN scheme, it is now possible to move across to the semi-supervised R-KFLANN clustering scheme, aiming to improve the unsupervised clustering process with respect to the external reinforcement received.

The framework architecture of the R-KFLANN system is presented in the Figure 5.1 below:

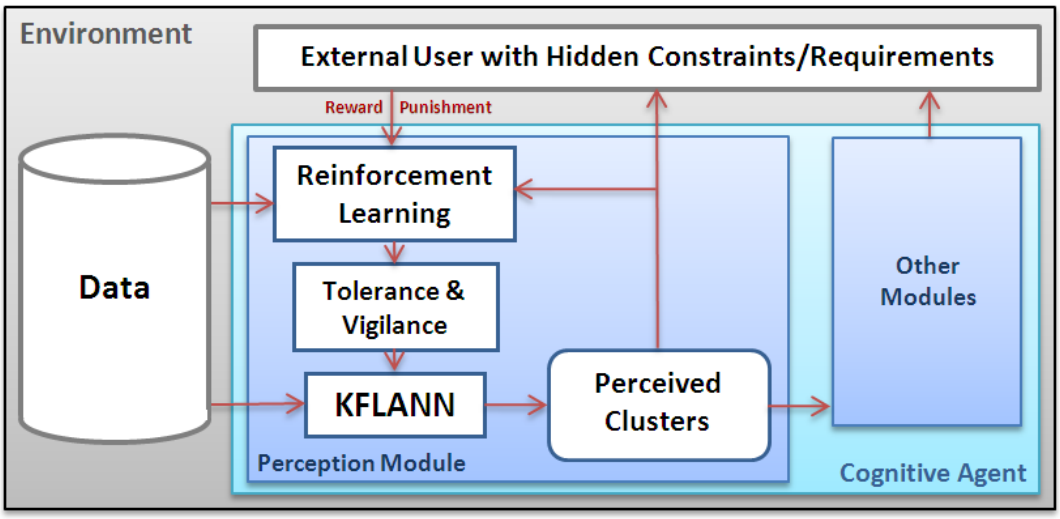


Figure 5.1 The R-KFLANN Framework

From Figure 5.1, the KFLANN algorithm takes the raw data as an input and produces a partition containing a small set of cluster prototypes that summarizes the dataset. The prototypes are then used by other external modules in an unknown task with a hidden requirement model. The reinforcement learning module subsequently influences the KFLANN behavior by directing its attention state, changing its perception of clusters, and providing a communication channel between the external system and the underlying clustering algorithm. After iterating through the learning process, the KFLANN adaptation behavior in response to the input data perceived is systematically generated. The whole closed-loop process forms an adaptive clustering system in which the output clusters change dynamically with the actual application requirements enforced. The formulation of the R-KFLANN reinforcement clustering system will be presented next.

5.1 Reinforcement clustering formulation

Since there are three components needed in defining our reinforcement learning problem: the state space S, the action space A, and the reward function r, the R-KFLANN system is formulated as follows:

First of all, the external mechanism, such as a human user or an end application process evaluating the clustering output is modeled as a system with an arbitrary hidden function $r : W \rightarrow \mathbb{R}$ that maps the clustering output W to a numeric reinforcement value. This reward function is hidden from the learning agent and is left undefined in the R-KFLANN design stage. The reinforcement is to be imposed on the R-KFLANN by the user only at run-time.

The reinforcement learning action is to adapt the clustering strategy given the

current state. The objective of this work is to focus on the tolerance aspect of KFLANN attention, where through each learning step *t*, the clustering scaling factor is modified incrementally as:

$$\psi_j(t+1) = \psi_j(t) + \Delta\psi_j(t);\ \Delta\psi_j(t) \in \{-0.1, 0, +0.1\}, \tag{5.1}$$

where j is the current state-determining cluster considered in the current episode. By performing an action of $\Delta\psi_j$, the data partitioning becomes more or less stringent. In the case that the action chosen $\Delta\psi_j(t) = 0$, the system has concluded that the current clustering cannot be improved further.

The state description is the main source of information on which the reinforcement learning agent relies when making a decision on the clustering strategy to be pursued. As a result, it has to be defined in a way that can reflect the characteristics of the data distribution in the region of interest, and also the behavior of the hidden reward function with respect to the changes in clustering strategies. The state definition is generalized by incorporating only the information directly observable to the learning agent: no prior knowledge tying to any specific application is included. The state vector is defined from all information available including the current and previous attention settings and the feedbacks received from the KFLANN clustering as follows:

$$s(t) = (\psi_j(t), \psi_j(t-1), \ldots, \psi_j(t-m+1), r(t), r(t-1), r(t-m+1)), \tag{5.2}$$

where $\psi_j(t)$ is the tolerance scaling factor used for the cluster j at step t. $r(t)$ is the reinforcement received from the clustering result, and m is the window size of the short-term memory maintained by the reinforcement learning agent.

This state representation is generic. In actual implementations, it is possible to include additional information from initial knowledge about the application domain into the state representation in order to further improve the clustering performance. As a natural fact, the more prior information, the better result. However, if no prior knowledge about the application is available, we have to do the best with only the basic information.

The implementation of short-term memory is to serve the scenario where the semi-supervised clustering process is non-Markov but is order-*m* Markov due to the hidden information of the task/user reward function, so that a Markovian state representation would still be achieved. In other words, the reinforcement learning agent uses past information to help better predict the state transition and future reward. The larger the memory window size, the better capability of the agent in disambiguating states, however, at a trade-off of the dimensionality of the state space. This use of the sequence of past observation-reward information to augment and reveal the hidden state in partially-observable problem follows the concept from the previous work on instance-based state identification called Nearest Sequence Memory (NSM) by McCallum [57, 58].

At each time step t, NSM records the snapshot of action-percept-reward instance $(a(t),o(t),r(t))$ into its short-term memory. To retrieve the Q-values, NSM identifies the current state from the k most similar situations in the past according to their preceding sequence of these action-percept-reward instances. The following similarity metric $n(\mathrm{s}(t_i),s(t_j))$ between two states $s(t_i)$ and $s(t_j)$ encountered at the time steps t_i and t_j is used (note the difference in the time index notation used):

$$n(s(t_i),s(t_j)) = \begin{cases} 1+n(s(t_i-1),s(t_j-1)), & \text{if } (a(t_i-1)=a(t_j-1) \wedge o(t_i)=o(t_j) \wedge r(t_i)=r(t_j)); \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

Hence the action selection and the Q-value update can then be performed using the standard Q-learning principle. The difference in this work is that, for simplicity, an exact matching with a fixed length *m* is used instead of k-nearest neighbors with a voting scheme. The action is also omitted from the action-percept-reward instance $(a(t),o(t),r(t))$ in the state vector since there can be only one possible sequence of $\Delta\psi_j$ given the same sequence of (ψ_j, r).

In addition to the Markov property discussed above, the convergence and stability of the R-KFLANN also depend on the stationary property and the learning rate [54] of the system. First of all, the learning rate α_1 should be initialized to a large value close to 1.0 and then monotonically decreased to zero as the learning steps approaches infinity, $\alpha_\infty \to 0$. Moreover, the series of α_l is to be defined such that its summation converges to a finite value.

On the other hand, given a Markovian state representation, the condition of the system stationary property can then be guaranteed. In other words, provided that the sequence $(\psi_j(t),\psi_j(t-1),..., \psi_j(t-m+1), r(t), r(t-1),...,r(t-m+1))$ is used to help the agent uniquely identify the true state *s**, the problem is stationary from the following behavior:

Firstly, the partially observable true state $s(t)=s^*$ could be visualized as "*the current tolerance factor* $\psi_j(t)$, and *the hidden underlying data and task requirements on* $\psi_j(t)$." The agent identify from the state signals that two states encountered are the same if and only if their sequence of percept and reward instances observed preceding them are the same. As a result, for any time $t > 0$, the new tolerance factor $\psi_j(t+1) = \psi_j^{'}$ following the

same state $s(t)$=s* (i.e. same $\psi_j(t) = \psi_j$) and action $a(t)$= $\Delta\psi_j$ will be identical: $\psi_j^{'} = \psi_j + \Delta\psi_j$ (stationary with t). Therefore, the resulting state $s(t+1) = (s*)'$ and reward $r(t+1) = r'$ will also be stationary. This is due to the fact that the new true state which is "*the current* $\psi_j(t+1) = \psi_j^{'}$, and *the hidden underlying characteristic of the data and the task requirement on* $\psi_j(t+1)$ " remain unchanged. Finally, the resulting state signal $s(t+1) = (\psi_j(t+1), \psi_j(t), ..., \psi_j(t-m+2), r(t+1), r(t), ..., r(t-m+2))$ will also be stationary, since both of the observed $\psi_j(t+1) = \psi_j^{'}$ and $r(t+1) = r'$ are stationary.

5.2 The R-KFLANN algorithm

The R-KFLANN algorithm is presented in Figure 5.2, where the R-KFLANN steps iterate between reinforcement learning and KFLANN clustering.

The R-KFLANN Algorithm

Step 1	NETWORK INITIALIZATION Given a dataset of n samples and d features, Set the vigilance. Initialize l = 1, tolerance factor ψ = 1.0 ; calculate σ_i, for all d features in the dataset and initialize $W = \varnothing$.
Step 2	Check the stopping condition. If satisfied, terminate. Else, start a new episode.
Step 3	Present the l^{th} pattern $\bar{x}_l$ to the input layer neurons. If the reinforcement learning step > 1, retrieve the previous output's local tolerance factor ψ=ψ_j attached to the $\bar{w}_j$ nearest to $\bar{x}_l$, then assign $\delta_i = \psi \cdot \sigma_i$. If the network does not have any output node yet, GOTO Step 9.

Step 4 *Determine all possible output node matches using*

$$\frac{\sum_{i=1}^{d} D[\delta_i^2 - (w_{ji} - x_{li})^2]}{d} \geq \rho,$$

$$D[m] = \begin{cases} 1; & m > 0 \\ 0; & m \leq 0 \end{cases}$$

D is the discriminant function imposed on the difference between the tolerance and the Euclidean distance.

Step 5 *Maintain a list of all matches that fulfilled Step 4. Using the winner take all criteria below, determine a single winner cluster from the matched list.*

$$winner_j = \arg\min_j [\sum_{i=1}^{d} (w_{ji} - x_{li})^2]$$

Step 6 *If winner is found, Assign the current pattern to the winning cluster. Else, Goto Step 9.*

Step 7 *If $l = n$, compute the mean values for each cluster and identify the cluster members closest to the means as the cluster centroids.*

If the collection of centroids is consistent through the epochs, report the output w and $\bar{\psi}$; GOTO Step 10.

Else, GOTO Step 8.

Else, increment l by 1 and GOTO Step 3.

Step 8 *Reshuffle all centroids to the top of the data list for the next epoch. The data sample size remains constant. Set $l = 1$. Reset all output nodes and GOTO Step 3.*

Step 9 *Create a new output neuron j. Attach the prototype-attention pair $(\bar{w}_j, \psi_j) = (\bar{x}_l, \psi)$ to the new neuron, assign the current pattern $\bar{x}_l$ to the cluster C_j created and GOTO Step 7.*

Step 10 *Receive new reinforcement r' on the output cluster quality, observe the new state s' reached. Update the Q-value of the previous state-action pair:*

$$Q(s,a) = Q(s,a) + \alpha \cdot [r' + \gamma \max_{a'} Q(s',a') - Q(s,a)].$$

Update the current state $s = s'$. If a terminal state is reached, GOTO Step 2.

Step 11 *Perform a new action a on the network's attention state based on the current $Q(s,a)$, reset all output nodes, and proceed to the next reinforcement learning step; GOTO Step 3.*

ρ is the vigilance, σ_i and δ_i are the standard deviation and the tolerance for i^{th} feature of the input space, $\bar{w}_j$ is the stored weight vector and $\bar{x}_l$ is the current pattern presented.

Figure 5.2 The R-KFLANN Algorithm

The algorithm starts off as an original KFLANN whose network attention is uniform throughout. The initial tolerance scaling factor $\psi_j(1)$ is set to a middle value of 1.0 to allow an easy adaptation in both ways.

Here the reinforcement learning paradigm is defined as a finite-horizon MDP where the reinforcement learning agent episodically explores each region in the data space, until it arrives at the terminal state in which either a fixed number of reinforcement learning steps is reached or no further clustering improvement could be done. An alternative to this incremental adaptation is to learn a policy for all the regions in the whole data space simultaneously. However, this approach will cause an exponential explosion in the state and action space sizes, and will slow down the learning drastically. In such situation, the action space size becomes: $|A'| = |A|^k$; and the state space size becomes: $|S'| = |S|^k$, when the total number of clusters is $k > 1$.

After the terminal state is reached, the episodes are then repeated to ensure that all data regions are sufficiently visited; the number of episodes required basically increases linearly with the number of clusters k. In each learning iteration t, the steps begin with a run of KFLANN using the current clustering parameters specified by $\bar{\psi}(t)$ vector attached to the prevailing prototype set $W(t)$. The network retrieves the tolerance from the relationship $\bar{\delta}_j(t) = \psi_j(t) \cdot \bar{\sigma}$ when a new input pattern is encountered, while $\psi_j(t)$ is judged by the region of the nearest $\bar{w}_j(t)$ where the pattern falls. The clustering then proceeds as in the original KFLANN, undergoing reshuffling processes. Whenever a new cluster C_j is formed, the attention ψ_j is inherited, creating a new prototype-attention pair $(\bar{w}_j, \psi_j)$. The reinforcement learning then determines the optimal Q-values and clustering policy using the Q-learning updating equation:

$$Q(s,a) = Q(s,a) + \alpha \cdot [r' + \gamma \max_{a'} Q(s',a') - Q(s,a)] .$$

The agent explores different strategies choosing the action of $\Delta\psi_j(t) \in \{-0.1, 0, +0.1\}$ as defined previously. If the tolerance factor $\psi_j(t)$ is lowered, the network grows denser output-layer neurons to store more prototypes in those particular regions. On the other hand, when ψ_j is raised, such as when the data is less complex or less important, the network becomes less attentive in learning and memorizing the training data, thus, merging the output clusters and pruning the output-layer neurons storing the prototypes in those regions, resulting in a relatively sparse network.

After the Q-table has been learned, it can then be used to automatically adjust the network's attention state. This is by choosing the action with the highest Q-value, i.e., following the optimal policy:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

In the next chapter, the experiments testing the performance of R-KFLANN will be presented.

CHAPTER SIX

EXPERIMENTS & DISCUSSION

In order to demonstrate the usefulness of R-KFLANN adaptation in improving the clustering results, two types of top-down objectives in the domain of data classification and robot navigation applications were investigated. In these experiments, the semi-supervised clustering worked directly on the reward/punishment feedback received from the higher-level user. In contrast to the traditional semi-supervised clustering, these scenarios encapsulate the generalized problem where the precise clustering constraints or requirements were hidden or could not be specifically formulated. The results and from these experiments are discussed in the Sections 6.1 and 6.2 below.

6.1 Data Classification Task

In this experiment, the well known UCI data classification problem [59] was used to test the R-KFLANN and its ability to adapt to external requirements. The objective was to examine whether the reinforcement given can effectively improve the clustering performance with respect to the classification task objective. In this semi-supervised clustering scenario, the class labeling information was hidden and only the overall failure rate was presented to R-KFLANN as a reinforcement signal. The labeling or pairwise relationship information was not available, e.g. due to the costliness or infeasibility in

obtaining it. This resembles the scenario of an automatic defect detection or letter sorting system, where the human inspector does not keep track of specific samples that are wrongly sorted or what their associated input features and correct classes are. After the inspection, he/she instantly throws away the defective parts/re-sorts the misplaced objects, incrementing the error count punishment. Therefore, the overall outcome yields a feedback $r \in [-100, 0]$ according to the high-level objective:

$$r(t) = -\left[\frac{\#Incorrect\ Predictions}{\#Total\ Predictions}\right] \times 100 \tag{6.1}$$

which indicates the degree of success/failure of the clustering system seen by the user.

The standard datasets from the UCI machine learning repository used in the experiments are summarized in the Table 6.1.

Table 6.1 Summary of the UCI datasets used

UCI Dataset	# Attributes	# Classes	#Instances	#Train	#Test
Iris	4	3	150	135	15
Wine	13	3	178	160	18
Glass	10	7	214	193	21
Ionosphere	34	2	351	316	35
Ecoli	8	8	336	302	34
Sonar	69	2	208	187	21
Pima	8	2	768	691	77
WDBC	32	2	569	512	57
Vowel	10	11	990	891	99

Initially the R-KFLANN produced a clustering output equivalent to that of the unsupervised KFLANN, ignoring any top-down characteristics existing in the datasets. The learning process was then carried out. In each reinforcement learning step, the classification accuracy performance of the R-KFLANN output was measured in order to

observe the R-KFLANN algorithm's response to the given signals. Figure 6.1 presents the plots of percentage classification accuracy against the reinforcement learning episode, showing the adaptation process of the clustering subjected to the external feedback on nine different datasets.

Subsequently, statistical t-test tests were then conducted. The generalization ability was evaluated using the ten-fold cross validation method.

Each of the datasets was initially divided into ten folds of stratified training and test subsets. The initial classification performance of the non-adaptive KFLANN output produced from each fold, and their respective generalization estimates on the unseen test set, were evaluated and recorded. Subsequently, the adaptation process took effect. The R-KFLANN algorithm was run ten times on each of the training sets with the same evaluation procedure repeated. As a result, four sets of classification performance observations, forming two pairs of KFLANN and R-KFLANN comparisons were obtained.

The significance paired-sample t-test was conducted on each pair. The objective of the significance test on the first pair was to ascertain the robustness of the R-KFLANN in improving the clustering output in various datasets used. The objective of the significance test on the second pair was to determine if the cluster adaptation made by R-KFLANN can also be generalized to provide an improvement in its performance on the unseen data.

The results in Figure 6.1 shows that, on all the datasets tested, the classification accuracy performances of the R-KFLANN improved notably with the reinforcement signals received. This indicates that R-KFLANN can successfully utilize the reinforcement feedbacks in adjusting the clustering behavior to suit the external requirements.

The mean and the standard deviation from each set of observations over the 10 folds are summarized in Table 6.2, Figure 6.2, and Figure 6.3. On each of the nine datasets, the-

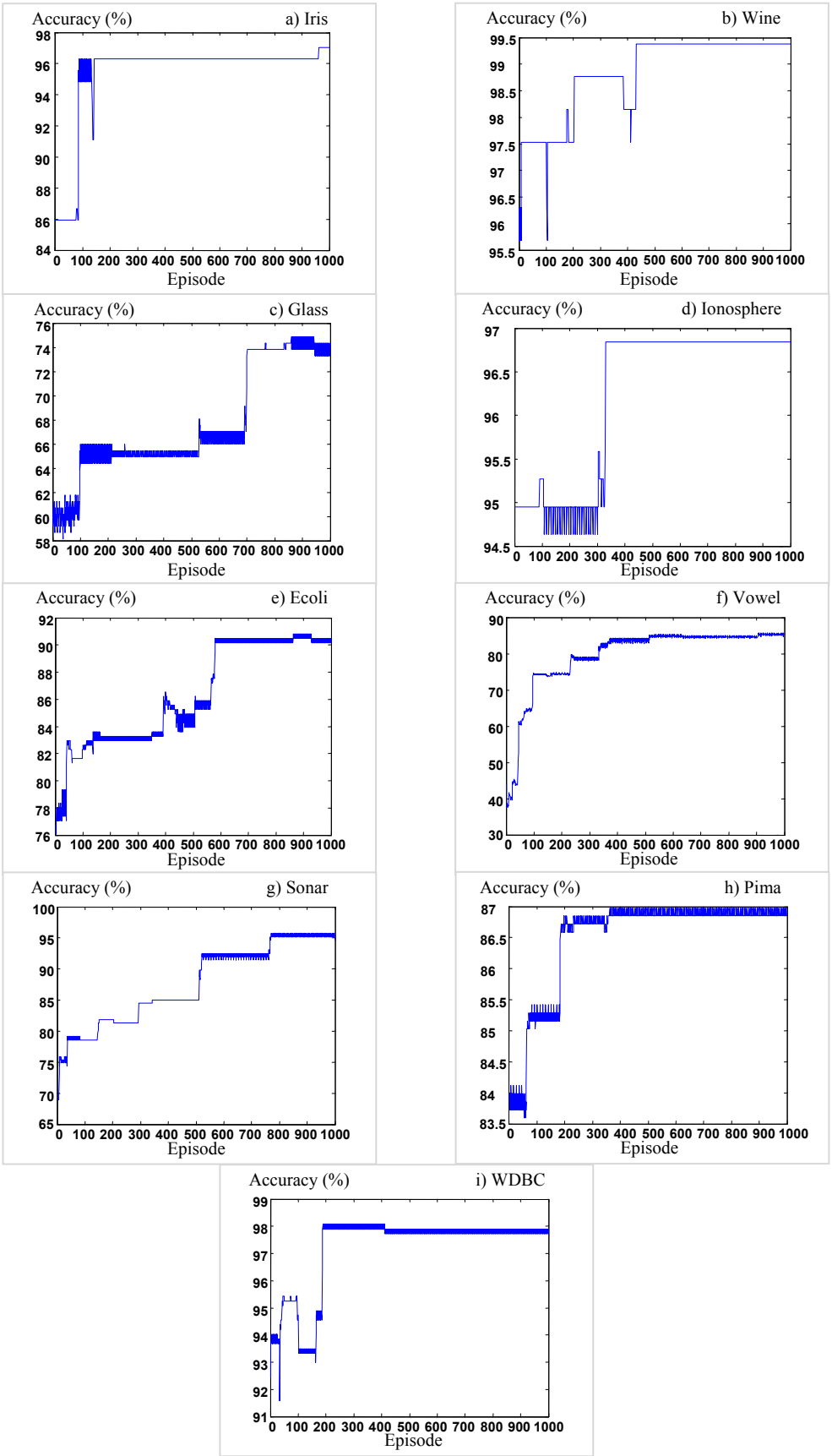


Figure 6.1 Adaptive behavior seen in the clustering performance plotted against the reinforcement learning episode for each UCI dataset

results from the two paired sample t-tests carried out are presented, where the pairs having a statistically significant improvement by R-KFLANN are indicated with a circle. In the tests, the significance level α applied was the commonly used level of 0.05. The p-values of the test were judged against the significance level set. The difference between two sets of observations is said to be significant when $p < \alpha$. The higher the p-value, the more likely the difference found in the mean occurred by chance.

It was found that on all nine datasets tested, the R-KFLANN adaptation was able to improve the training accuracy significantly with a p-value below 0.01 (indicated with a closed circle). Additionally, the generalization error estimates of R-KFLANN output clusters were found to be better on all of the datasets. On six out of nine datasets tested, the improvements were found to be significant where on the Iris, Glass, Sonar, and Vowel datasets, the R-KFLANN output performed significantly better with $p<0.01$. On Wine, Ionosphere, and Wisconsin Diagnostic Breast Cancer, the improvement is however not statistically conclusive as assessed on the same significance test.

Table 6.2 Comparison between KFLANN and R-KFLANN based on statistical tests on the 10-fold cross-validation results

UCI Dataset	**Training Accuracy (%)**		**Test Accuracy (%)**	
	KFLANN	**R-KFLANN**	**KFLANN**	**R-KFLANN**
Iris	88.59 ± 2.04	**97.11 ± 1.23** ●	90.00 ± 6.48	**97.33 ± 4.66** ●
Wine	93.13 ± 2.74	**99.50 ± 0.65** ●	93.30 ± 7.31	**96.09 ± 2.71**
Glass	61.22 ± 1.45	**74.91 ± 6.06** ●	51.51 ± 11.05	**67.57 ± 7.19** ●
Ionosphere	94.27 ± 1.80	**96.68 ± 1.42** ●	64.46 ± 6.36	**65.18 ± 7.14**
Ecoli	80.70 ± 3.73	**89.16 ± 1.92** ●	75.96 ± 11.68	**81.49 ± 8.86** ○
Sonar	75.96 ± 5.30	**93.96 ± 3.21** ●	67.34 ± 8.53	**85.09 ± 6.19** ●
Pima	84.92 ± 0.81	**87.01 ± 1.38** ●	70.45 ± 4.43	**71.49 ± 3.46** ○
WDBC	92.15 ± 0.98	**96.52 ± 0.70** ●	91.22 ± 3.17	**92.30 ± 3.14**
Vowel	40.38 ± 2.49	**88.38 ± 3.69** ●	40.30 ± 5.46	**78.38 ± 2.70** ●
Statistically significant improvement: ● $p < 0.01$, ○ $p < 0.05$				

The stabilities in the performances of the two algorithms in terms of the standard deviation of the accuracy results are comparable. In other words, the adaptation process did not degrade the stability of the clustering performance. In both KFLANN and R-KFLANN,

the test result has a higher standard deviation than the training result across the ten folds. This is due to the fact that the generalization result is dependent on each split between the training set and the test set of each run.

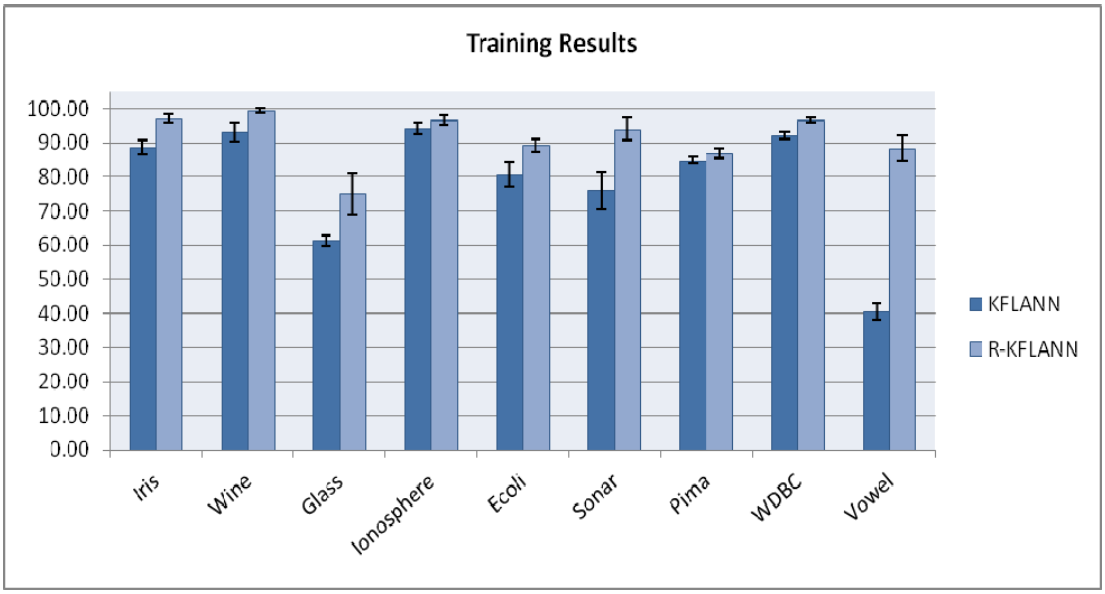


Figure 6.2 Comparisons between KFLANN and R-KFLANN based on the training results for different datasets

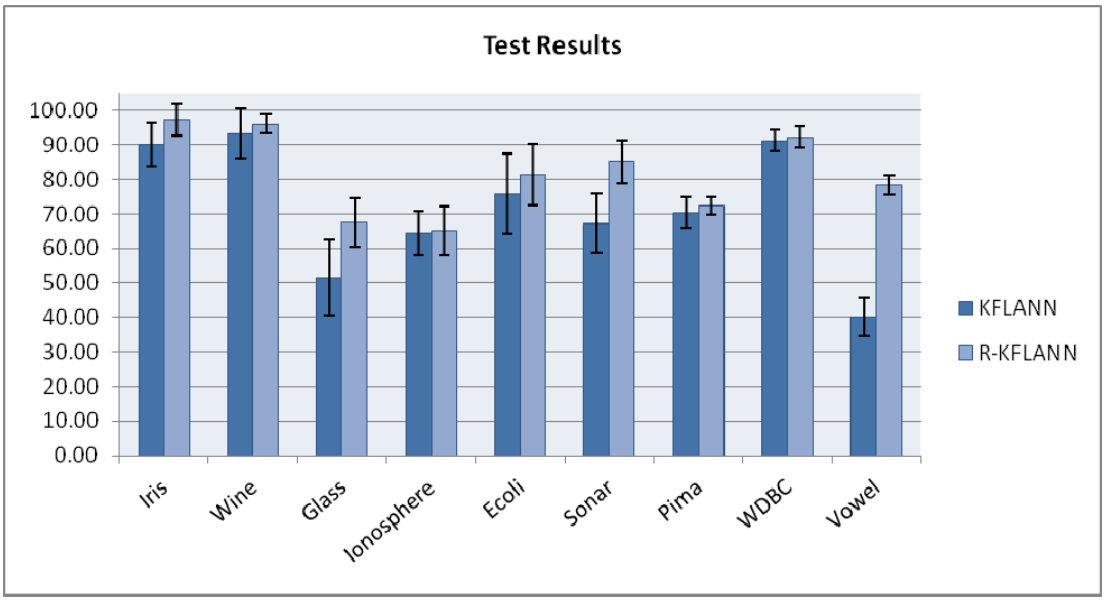


Figure 6.3 Comparisons between KFLANN and R-KFLANN based on the test results for different datasets

To illustrate the range of possible effects (improvement or degradation) in the cluster quality caused by the reinforcement process, the mean difference of the training and test results between KFLANN and R-KFLANN and their corresponding 95% confidence interval plotted against the respective datasets are provided in Figure 6.4. Both overall training and test results were improved after the learning (mean differences are lower than zero). The effect ranges from the largest improvement observed in Vowel dataset (48.00% in the training set and 38.09% in the test sets) to the smallest improvement seen in Pima dataset (2.08%) and Ionosphere dataset (0.72%). The overall improvement in the test results is less evident than in training except for the Glass dataset where the size of the mean difference in training (13.69%) is slightly lower then that in the test results (16.053%) and for the Sonar and Pima datasets where the mean differences are almost equal in both training and test sets.

This is expected because the R-KFLANN adaptation objective was to achieve the clusters of the best quality judging from the reinforcement feedback provided by the training samples, not by the test samples. In this sense, it is also possible for the adaptation of R-KFLANN to overfit the training data if the adaptation process continues for too long. When such problem is encountered, the early stopping technique could be employed in order to prevent the overtraining of the R-KFLANN.

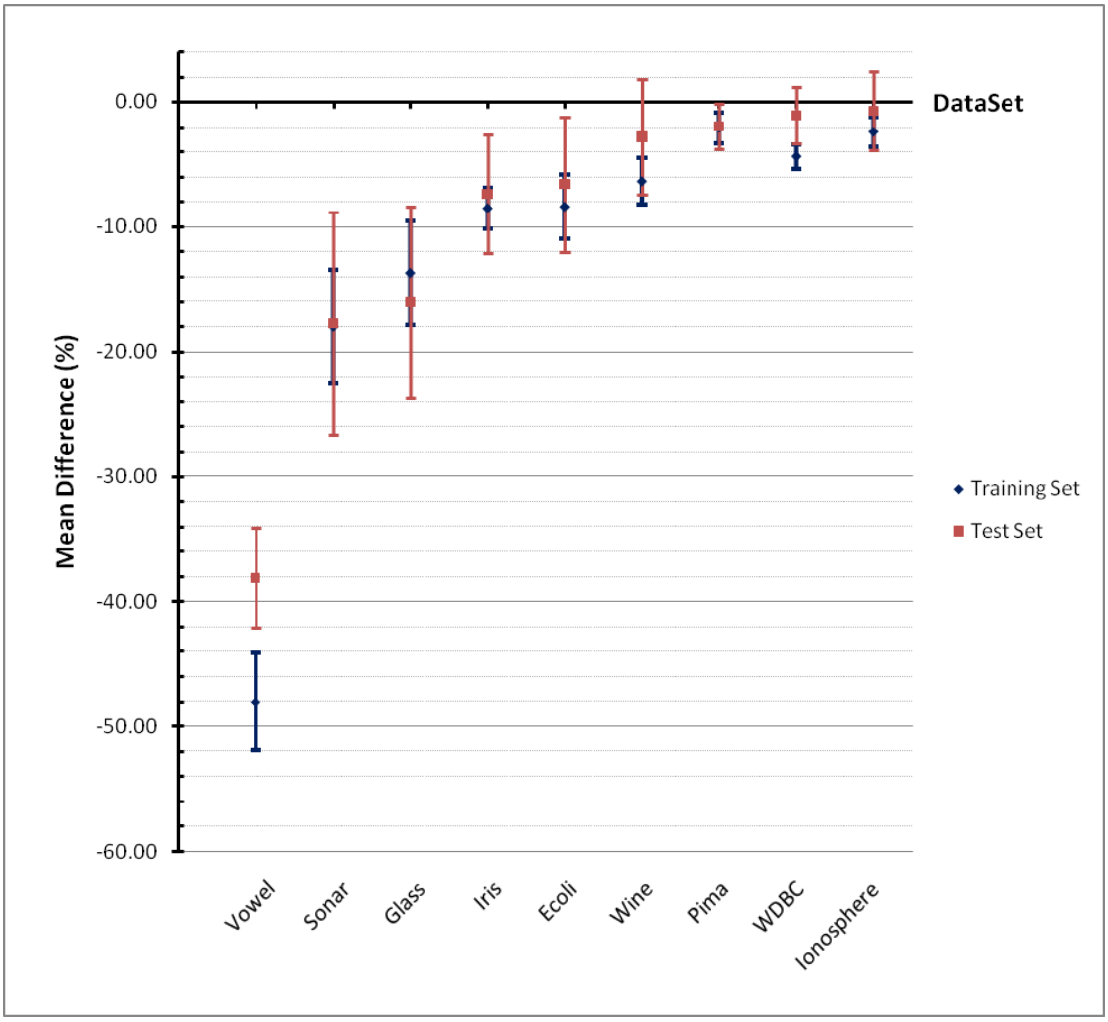


Figure 6.4 Mean difference of KFLANN VS R-KFLANN training and test results for different datasets

In most cases, given the 95% confidence interval, it can be seen that the mean difference result stays in the negative range. Therefore, the experimental results conclude that R-KFLANN can effectively make use of the reinforcement in adapting the clustering behavior and improve the clustering towards the desired outcome. The improvement is significant on all training datasets indicating the robustness in the learning performance and it can also generalize well in most of the datasets tested.

Additionally, R-KFLANN was also tested against four other supervised and unsupervised systems: one decision tree learning, one probabilistic-based, and two neural network frameworks. Though, the comparison with supervised algorithms is not a completely fair comparison since, with reference to Figure 2.1, among all learning schemes, the supervised classification uses the most precise information for the training of the system. As a result, their performance serves roughly as an upper bound for the other types of learning schemes. Table 6.3 summarizes the mean accuracy results of R-KFLANN, KFLANN, C4.5, Naïve Bayes classifier, Radial Basis Function (RBF), and Kohonen Self-Organizing Map (SOM). The statistical t-tests were conducted across the ten folds on all datasets.

Table 6.3 Comparisons of R-KFLANN with KFLANN, C4.5, Naïve- Bayes, RBF, and SOM Based on Statistical Tests on The 10-fold Crossed-Validation Results

	Test Accuracy (%)					
UCI Dataset	R-KFLANN	KFLANN	C4.5	Naïve Bayes	RBF	SOM
Iris	97.33 ± 4.66	90.00 ± 6.48 ●	94.00 ± 5.84	96.00 ± 4.66	95.68 ± 3.59	91.21 ± 4.24 ●
Wine	96.09 ± 2.71	93.30 ± 7.31	92.71 ± 7.00	97.25 ± 2.90	98.81 ± 3.62	90.74 ± 5.36 ●
Glass	67.57 ± 7.19	51.51 ± 11.05 ●	70.25 ± 10.45	49.69 ± 9.04 ●	61.73 ± 7.48	51.06 ± 6.86 ●
Ionosphere	65.18 ± 7.14	64.46 ± 6.36	91.50 ± 4.68 ■	82.15 ± 6.23 ■	84.16 ± 4.30 ■	71.54 ± 5.93 □
Ecoli	81.49 ± 8.86	75.96 ± 11.68 ○	83.47 ± 4.82	85.08 ± 7.52	83.90 ± 6.69	76.32 ± 6.17 ○
Sonar	85.09 ± 6.19	67.34 ± 8.53 ●	70.48 ± 7.16 ●	68.68 ± 7.53 ●	79.31 ± 6.83 ●	67.84 ± 6.38 ●
Pima	71.49 ± 3.46	70.45 ± 4.43 ○	75.01 ± 5.12	75.79 ± 4.64	73.52 ± 3.95	68.49 ± 2.73
WDBC	92.30 ± 3.14	91.22 ± 3.17	92.95 ± 4.01	92.96 ± 3.62	95.28 ± 2.43 □	90.21 ± 2.64
Vowel	78.38 ± 2.70	40.30 ± 5.46 ●	79.09 ± 3.30	67.88 ± 6.42 ●	75.03 ± 2.52 ○	61.82 ± 3.73 ●
Statistically significant: improvement ● $p < 0.01$, ○ $p < 0.05$; degradation ■ $p < 0.01$, □ $p < 0.05$.						

The result indicated by a circle or a rectangle denotes a significant improvement or degradation of R-KFLANN from each algorithm. First of all, regarding supervised systems, it was found that R-KFLANN obtained a significantly better result on one dataset, i.e., Sonar, and one significantly worse result on Ionosphere when comparing with a C4.5 classifier. On the other hand, with respect to Naïve Bayes classifier, R-KFLANN obtained three significantly better results on Glass, Sonar, and Vowel, and one significantly worse result on Ionosphere. Finally, for an RBF network, R-KFLANN obtained two significantly

better results on Sonar and Vowel, and two significantly worse results on Ionosphere and WDBC. On the rest of the datasets, the improvement or degradation is not statistically conclusive. In other words, the performances of the two classes of learning are not significantly different. This indicates that, even when only weaker reward/punishment information was available for the training, the semi-supervised framework was still able to achieve a performance comparable with these supervised classifiers which were given full labeling information. In contrast, when comparing with an unsupervised SOM neural network, the R-KFLANN provided a significantly better result. It outperformed SOM in six out of nine datasets, namely, Iris, Wine, Glass, Ecoli, Sonar, and Vowel, and was inferior in only one dataset, i.e. Ionosphere. This overall improvement is due to the fact that the R-KFLANN was supplied with additional reinforcement information during the training phase which, as a result, shifts it across from an unsupervised to semi-supervised learning paradigm.

6.2 Robot Navigation Task

Since a clustering-based mapping has recently become of interest in the robot navigation research field [6], this section discusses the experimental results obtained from the semi-supervised clustering in a robot mapping scenario.

The first part of this section provides a brief background on robot mapping and navigation, together with the system implementation and its specific requirements on the low-level clustering system. Subsequently, the comparisons between the robot navigation behaviour using different clustering schemes are then discussed.

In autonomous robot navigation, a map is usually employed to store information about the environment. Unlike traditional localization and mapping techniques, the clustering-based approach allows the robot to recognize different places by using only allothetic or externally-represented cues such as the laser or visual signatures. This approach reduces the system's dependency on the idiothetic information, e.g. the x-y position, derived based on the robot's estimated internal state which requires an initialization and is susceptible to cumulative error [60-63]. Allothetic navigation has also been found to be utilized by animals. Experiments on rats indicate that they do not rely on mere idiothetic information but also allothetic cues [64, 65]. In the studies on animal navigation, it was found that the hippocampus plays an important role in the localization and navigation abilities using special neurons called place cells [66, 67]. A place cell has a special function of being reactive to only a specific region in the space, which defines the place field of the cell [68].

In Figure 6.5a, an example of an allothetic cue from the laser and compass readings are illustrated. The robot uses laser readings in eight directions: N, NE, SE, S, SW, W, NW.

A clustering system serves to summarize the raw samples collected during the exploration and form a topological map as shown in Figure 6.5b. The topological map created is then used by the path-planning and the motor control modules to navigate the robot from one place to another as required by the user.

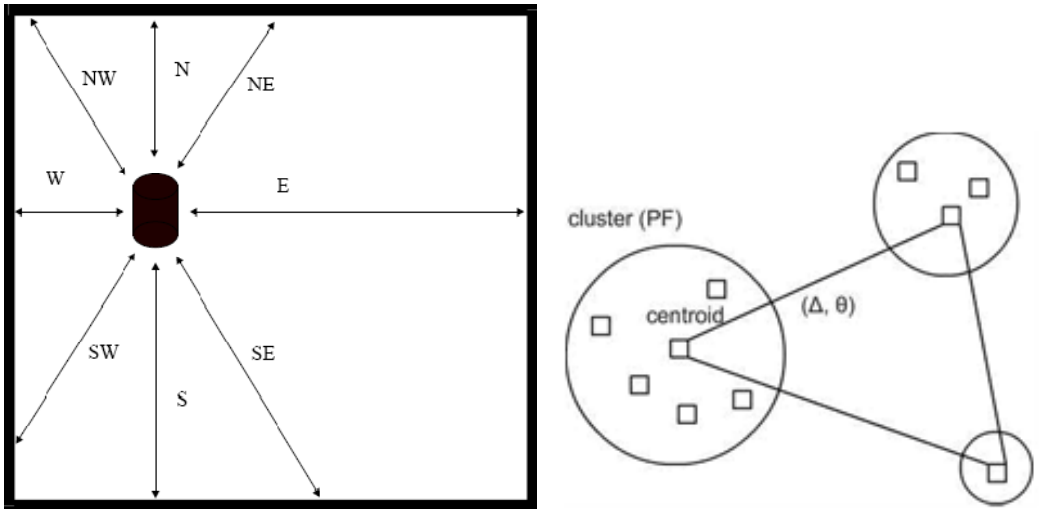


Figure 6.5 a) Environment signature and b) Cluster-based topological map

The navigation system was implemented on a PowerBot AGV shown in Figure 6.6 with the dimension of approximately 90 cm x 70 cm and about 70 cm high. The experiment was conducted in a confined space of a size about 7m x 3.5m. The robot was equipped with two URG laser sensors and a digital compass. The two laser sensors were mounted back-to-back with a 1-cm gap separation on top of the robot in order to provide a 360^{o} scanning coverage. The exploration data were collected by manually driving the robot through the environment using a wireless controller at an average speed of 0.3-0.35m/s. The robot acquired laser sensor information continuously at each displacement of 0.3m until the environment was fully explored. These data would then be used to produce topological maps for subsequent navigation of the robot.

The robot's behaviour was monitored by means of a video camera in order to assess

the performance. It was given delivery tasks of repetitively visiting each of the three destinations marked on the floor with the dimension of 75 cm x 75 cm for ten times. The success or failure of the tasks was evaluated according to the centre position of the robot with respect to the destinations boundaries given.

Figure 6.6 Experimental Setup

The objective of this experiment was to test whether the proposed R-KFLANN semi-supervised clustering could also adapt to the reinforcement signal imposed from the high-level constraint of the navigation task. These top-down specifications was described in terms of the two high-level criteria measurable in terms of the navigation end result as stated below. The explicit high-level labeling of places was not available. The relationship between the low-level clustering constraints and the high-level navigation result with respect to the evaluation criteria could not be formulated by the user or the mapping system designer: it had to be learned on-the-fly by the R-KFLANN clustering agent.

The first evaluation feedback was whether the map could correctly and efficiently represent the destination places required by the tasks. To illustrate this in a real situation, a

simple scenario could be given as, for example, there is a refrigerator at one corner of the room and the robot's task is to fetch some cooking ingredients from it and deliver them to the drop-off location at the other corner. The navigation module needs to get the robot close to the fridge enough for the vision and manipulator modules to recognize the fridge door and open it to reach the objects inside. Hence, in this scenario, after each navigation task is completed, the end processes which are the robot's vision and manipulator can then provide a feedback on how well it has performed. They could tell whether the fridge is out of reach and the robot should move closer to it, or whether the fridge is nowhere near enough to be seen at all. In this implementation, a simple feedback $r_1 \in [0,100]$ was calculated based on the location of the boundary marked on the floor. Failing to reach the boundary, the system was given a punishment for not completing the task. This was determined according to the distance from the robot's final location to the target space $d_{miss} \geq 0$ as in the equation below. The larger the error d_{miss}, the smaller the r_1.

$$r_1(t) = \frac{d_{max} - d_{miss}}{d_{max}} \times 100 \tag{6.2}$$

The normalizing factor d_{max} is the maximum error possible, which could be the size of the room or the maximum detection range of the camera in the above example.

The second evaluation feedback was on the path representation. A good-quality mapping should in turn allow a more efficient path to be represented. Specifically, it should be able to cater a more detailed path for important or dangerous regions but disregard unimportant, irrelevant regions to speed-up the navigation. Therefore, the time taken in completing the task t_{nav} was penalized with respect to the navigation timeout t_{max}. The reinforcement $r_2 \in [0,100]$ used is described in the following equation:

$$r_2(t) = \frac{t_{max} - t_{nav}}{t_{max}} \times 100 \tag{6.3}$$

From the criteria above, the total reinforcement fed back to the robot was the superposition of these two independent components with the same range. Different mapping schemes were tested in performing the task. Firstly, four maps were built based on non-adaptive KFLANN. These consisted of one default map built using the standard deviation (SD-KFLANN), i.e., the representation of the space was solely determined in a bottom-up manner from the existing patterns in the environment, and three using tolerance values adjusted manually by the user. Subsequently, the adaptive semi-supervised R-KFLANN was compared. The navigation performance and mapping behaviour based on each of these clustering schemes are discussed as follows.

Firstly, the default map obtained using the standard deviation is shown in Figure 6.7. In this mapping scheme, the global tolerance used for all clusters was automatically determined from the standard deviation calculated within all samples collected during the robot exploration phase. There was no human interference in the clustering process and the map formation was completely guided by the environment (bottom-up). It is noted that the circles however do not show the actual place field representation, since the actual field of influence is in a d dimensional space with $d > 2$. The circles only assist the observer in estimating the field of influence in the two-dimensional map.

As seen from the Figure 6.7, since the testing space was an empty room with no turnings or corners, the KFLANN clustering faithfully reflected this nature of the test environment by having a relatively course clusters over the whole region, except for the lower-right corner where the bottom-up attention draw the network's interest to in the irregularity found at the place where the small opening used as the entrance to the testing area was located. Even though these low-level effects of the environment were

well-represented, it was found that the effects from the task's and human user's perspectives were not incorporated into the map. As a result, the robot was not able to distinguish the places labelled on the floor, and hence, failed to reach all of the locations assigned by the higher-level tasks. In practice, this problem is usually solved manually [6] by adjusting the tolerance scaling factor or using a preset tolerance instead of using the standard deviation as will be discussed next.

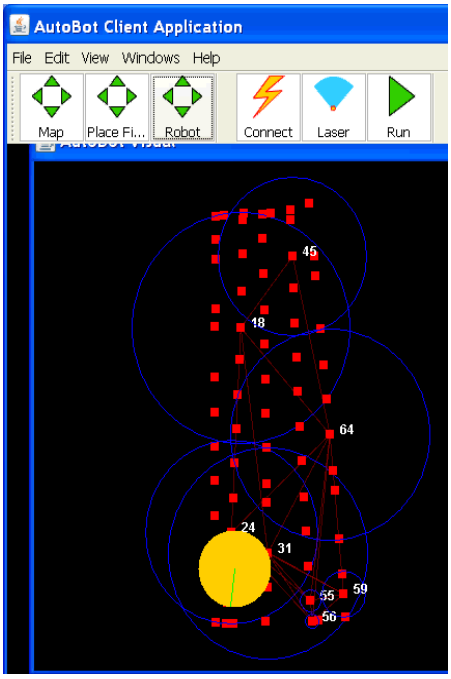
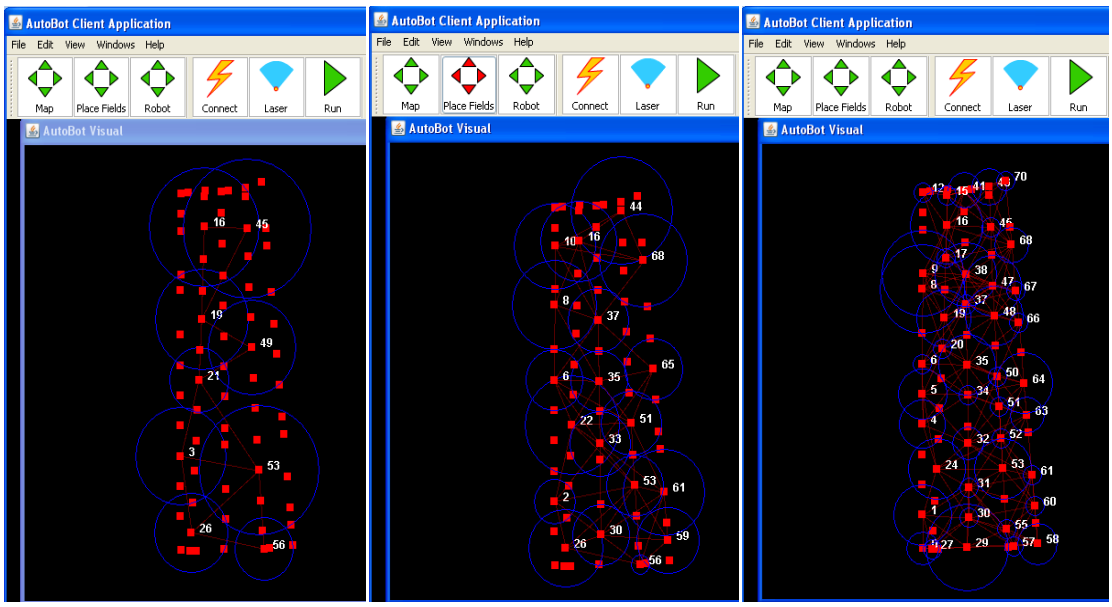


Figure 6.7 KFLANN map based on the standard deviation

By manually adjusting the tolerance of the original KFLANN, three maps are shown in Figure 6.8a-c using the tolerances of 60, 45, and 30 cm. In this scheme, the system was set with a decreasing level of tolerance and an increasing level of attention until the delivery task was satisfied.



(a) (b) (c)

Figure 6.8 KFLANN maps based on different manually-set tolerance levels

It was found that, using the first two maps in Figure 6.8a-b, similarly to the previous experiment, the robot was still not able to correctly reach the three destinations set, especially the places near the extremity as the two corners of the room.

However, by increasing the attention level further to Figure 6.8c, the robot could successfully fulfilled the tasks given in eight out of ten trials. The accuracy-memory trade-off could be clearly observed among these maps with the attention change. When the network attention increased, the navigation accuracy results improved while the number of clusters used in representing the entire space also increased. The average time taken over all trials was 30.67 seconds. The navigation results of the robot using this map were subsequently compared with the ones using the semi-supervised R-KFLANN map as presented next.

The final clustering output learnt by R-KFLANN is illustrated in Figure 6.9. The R-KFLANN was given the semi-supervised feedbacks specified previously so that it could adjust its clustering according to the high-level requirements.

It was observed that the emerging clusters were not homogeneous in their sizes as the system tried to focus on the patterns belonging to the clusters corresponding to the destination places given. Smaller clusters were observed around the destinations (i.e. resulting from a top-down information) and the place where the entrance to the test area was located, giving an irregularity in the laser signature (i.e. resulting from a bottom-up information). On the other hand, larger clusters were observed in other plain space, where nothing interesting was perceived during the navigation. At the beginning, the robot failed to fulfill the delivery task requirement as it could not correctly locate the destinations. However, after the adaptation process, the robot could eventually recognize all of the destinations and successfully fulfilled all of the tasks given. The resulting navigation behaviour of the robot and the efficiency of the map produced was subsequently analysed in comparison with the map built using the Manual KFLANN clustering presented earlier.

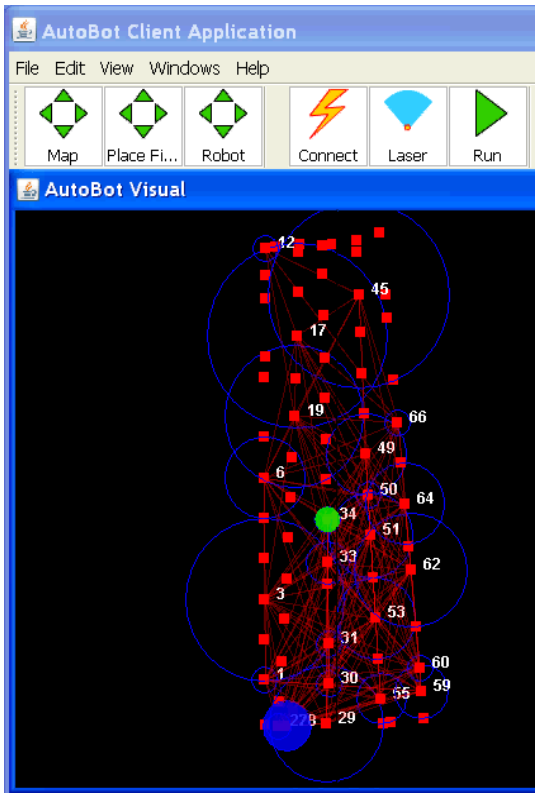


Figure 6.9 The R-KFLANN map

Firstly, the resultant path planned using each map is illustrated in Figure 6.10. In order to fulfil the same delivery task, it is seen that the R-KFLANN mapping allows a better path with less complexity for the navigation. For the same pair of starting point and destination from cluster 34 to 12, the path planned from the Manual KFLANN map is 34→35→37→38→17→16→12, requiring six steps of re-localization, planning, and heading adjustment procedure before reaching the destination. On the other hand, the path planned using an adaptive semi-supervised R-KFLANN map suggests 34→19→17→12 which involves only three steps of such process. The average of the actual number of times the robot executed this process before completing the same task is given in Table 6.4, which are 5.67 for the Manual KFLANN map and 4.33 for the semi-supervised R-KFLANN map, with a 23.63% decrease. Similarly, the time taken in completing a delivery task reduced from the Manual KFLANN to the R-KFLANN system. As shown in Table 6.3, it took 30.67 seconds on average for non-adaptive and 24.67 seconds on average for adaptive system, yielding a 19.56% reduction.

Table 6.4 Average Performances of Manual and Semi-supervised Mapping Systems

	Manual Mapping	Semi-supervised Mapping	Difference
Time (seconds)	30.67	24.67	-19.56%
#Re-localization, planning, & heading adjustment (times)	5.67	4.33	-23.63%
Memory (clusters)	37	24	-35.14%

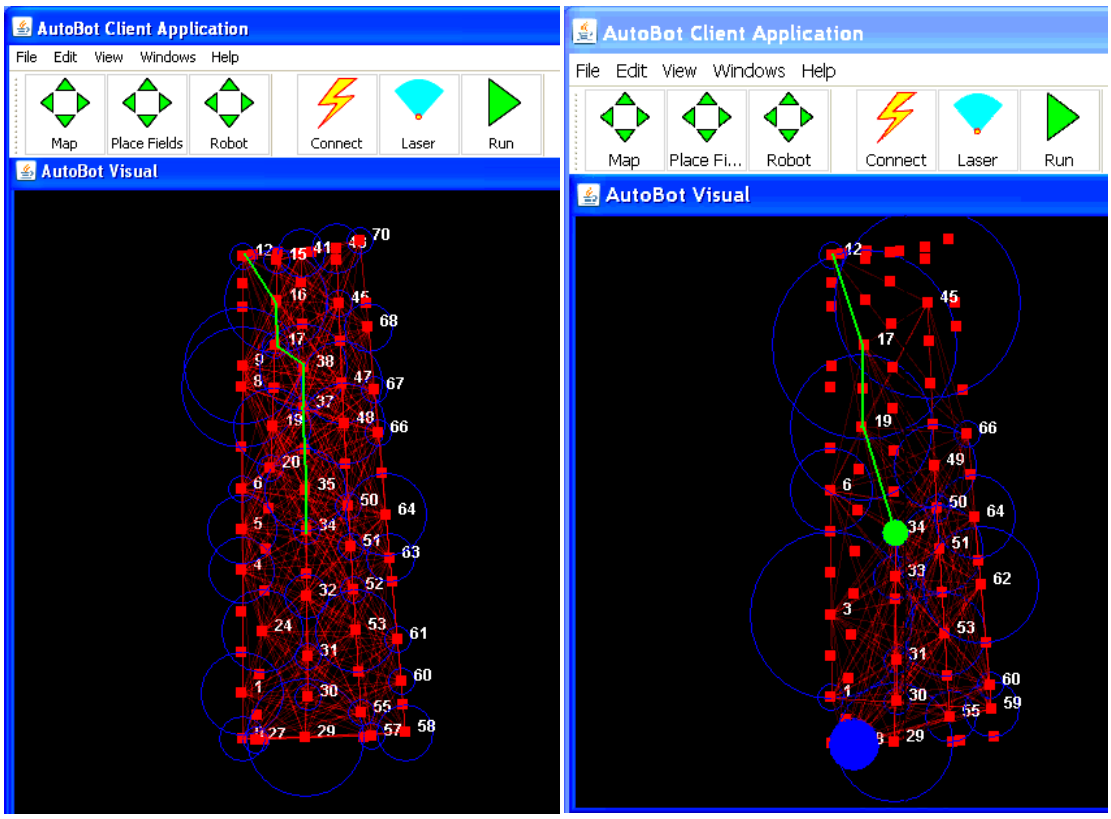


Figure 6.10 Sample paths taken by the robot navigating between the same pair of starting point and destination using: a) Manual KFLANN mapping and b) Semi-supervised R-KFLANN mapping

In addition, the memory needed by each system to store the representation of the whole space was reduced. From Table 6.4, the number of clusters required by R-KFLANN map was 24, which reduced by 35.14% from 37 clusters required by non-adaptive KFLANN. This improvement in the efficiency is because, using the KFLANN mapping, the robot could not actually incorporate any characteristic of individual places in the environment with respect to the high-level task objectives but rather paying high attention to every pattern perceived, regardless of the relevance or importance of it. Using the R-KFLANN system, the resultant map shows better representativeness since the robot was able to learn to focus only on the important places relevant to the task and relax its perception otherwise.

In conclusion, this experiment provides positive results, demonstrating an advantage of R-KFLANN semi-supervised clustering framework in improving the clustering process with respect to the high-level task requirements in a robot mapping application.

The results show that R-KFLANN was able to adapt its clustering based on the navigation performance feedback when the low-level clustering constraints could not be directly formulated by the user. Using top-down reinforcement information, the R-KFLANN-based mapping adapted the output clusters to provide a map which illustrated a better representativeness of the whole space in terms of the high-level task pursued. From this reason, the new cluster model of the space resulted in a better efficiency in terms of path complexity, navigation and time and action, and memory used. The semi-supervised clustering based mapping system demonstrated an improvement by eventually fulfilling the navigation task requirement which would not be achievable unless the system was tuned manually using a small tolerance. The adaptation of R-KFLANN clusters in the robot navigation tasks was carried out online, i.e. the updating of Q-values and the action selection were done incrementally with the delivery tasks performed. However, even though the online learning was successfully done in a laboratory setting, in real scenarios it is usually more feasible for the learning to be done off-line with a fixed set of data readily provided. This is due to the large number of reinforcement learning steps required. Therefore, in order to accelerate the learning process to better serve online application requirements, the speed of R-KFLANN could also be improved as follows:

First of all, in this implementation the Q-learning was used where the complexity increases with state-action space size. From the definitions, the size of the action space was fixed while the size of the state space increased with the window size of the memory maintained by the reinforcement learning agent. Therefore, to reduce the state-space size,

the optimization of window size could be done by applying the methods from the previous work on state identification, which directly address this issue of memory trade-off. For example, Nearest Sequence Memory (NSM) [57], with its variable length of sequence matching, has been shown to learn faster than using a fixed length memory. Otherwise, Utile Suffix Memory (USM) and U-Tree [69, 70] determines the amount of memory needed by adding a new experience only when the statistical test suggests an aliasing in the current state representation. Additionally, instead of using Q-learning which is based on a tabular representation for the state-action values, it is also possible to employ function approximation methods [37] using parameterized functional form, where the number of parameters is much less than the number of state-action pairs. Furthermore, prioritized sweeping [71] could also be applied to guide the exploration of state space in order to accelerate the learning process with a large state space.

CHAPTER SEVEN

CONCLUSION

This chapter summarizes the research work done in developing the R-KFLANN semi-supervised clustering framework. The review of the current state of research on data clustering shows that there have been great interests in improving clustering results with respect to high-level objectives imposed from the user or task given auxiliary information. In the previous work, the clustering systems are able to learn to improve their results by specifically working with constraint information in a form of partial labeling or pairwise relationships of the data.

As discussed by Cohn et al., 2008 [14], Wagstaff, 2007 [32], and Basu, 2003 [11], the possible feedbacks given by the user in semi-supervised clustering problems are not restricted to only these two formats. The users may have in their mind other types of constraints suitable for specific applications, otherwise, they may not be able to articulate any precise low-level constraints at all, but instead only know when the clustering looks correct to them. In contrast to the amount of research in this area, little work has been done specifically to solve the problem using other types of constraints or using abstract “good” or “bad” feedback to guide the clustering.

This thesis aims to provide a general semi-supervised clustering framework which can be interactively guided by only information pertaining to reward or punishment; no

explicit formulation of data relationships or constraints is required. This weak coupling between the clustering adaptation mechanism and the external constraint information provides a structure which allows the end users to define arbitrary clustering characteristics desired beyond those related to classification measures. It also allows the constraint type to be imposed at the run-time by the user as opposed to a predefinition by the clustering algorithm designer.

The Reinforcement-KFLANN clustering proposed inspired by the concept of neural network's attention state was presented. The investigation was documented to illustrate how control over the network's attention state can direct its perception of clusters towards what desired by the users. This documentation was initiated with an extension of KFLANN to support the locally adaptive attention as introduced in Chapter 3. This extension resolved the problem associated with non-uniform cluster characteristics which typical prototype-based clustering algorithms were inadequate [8]. The R-KFLANN algorithm with its adaptive clustering framework was formulated in Chapter 5, providing a communication channel for the unsupervised augmented KFLANN to interact with its environment.

Experiments were conducted to demonstrate the usefulness of the proposed semi-supervised clustering system under two scenarios of high-level tasks: data classification and robot navigation. Without prior information on the task or constraint types, the clustering system was left to adapt its clustering output according to the guidance from the reinforcement and improve the performances on-the-fly.

The R-KFLANN was initially tested in the classification scenario of automatic defect detection/letter sorting where only the miss-sorting count information weakly coupled with the hidden class label was given. It was found that the R-KFLANN system was able to significantly improve the clustering using the feedback information available.

This clustering performance was also found to be comparable with the two supervised classifiers. Furthermore, in a scenario of a robot navigation problem, R-KFLANN was able to improve the clustering-based mapping performance even when the low-level clustering constraint could not be explicitly formulated by the user. To begin with, the R-KFLANN mapping could fulfill the navigation tasks which would not be achievable by the KFLANN, unless the system was tuned manually by the human using a small tolerance. In addition, the R-KFLANN system was able to adapt its clusters to provide a more representative map, resulting in a better efficiency in terms of path complexity, navigation and time and action, and memory used. These results show that, without being restricted to the traditional format of specific semi-supervised constraints, the semi-supervised R-KFLANN can be used to employ the feedback information received and leverage the clustering process towards a better performance according to top-down objectives stated by the user.

This investigation has resulted in the following publications. While the first on the list is under a second review, the other two papers have been accepted.

List of Publications

1. R. Tse, A. Tay, and J. M. Zurada, "Towards General Semi-Supervised Clustering Using A Cognitive Reinforcement K-Iteration Fast Learning Artificial Neural Network (R-KFLANN)," *IEEE Transactions on Neural Networks,* under review.

2. R. Tse and L. P. Tay, "Towards a fully-autonomous cognitive robot navigation," in *The 13th International Conference on Cognitive and Neural Sciences*, Boston, MA, 2009.

3. R. Tse, L. P. Tay, and W. Hutama, "Robot navigation using KFLANN place field," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Singapore, 2008.

REFERENCES

[1] L. O. Hall, A. M. Bensaid, L. P. Clarke, R. P. Velthuizen, M. S. Silbiger, and J. C. Bezdek, "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain," *IEEE Transactions on Neural Networks,* vol. 3, pp. 672-682, 1992.

[2] K. Tasdemir and E. Merenyi, "Exploiting data topology in visualization and clustering of self-organizing maps," *IEEE Transactions on Neural Networks,* vol. 20, pp. 549-562, 2009.

[3] P. Muneesawang and L. Guan, "Automatic machine interactions for content-based image retrieval using a self-organizing tree map architecture," *IEEE Transactions on Neural Networks,* vol. 13, pp. 821-834, 2002.

[4] L. Zhao, A. de Carvalho, and Z. Li, "Pixel clustering by adaptive pixel moving and chaotic synchronization," *IEEE Transactions on Neural Networks,* vol. 15, pp. 1176-1185, 2004.

[5] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks,* vol. 16, pp. 645-78, 2005.

[6] R. Tse, L. P. Tay, and W. Hutama, "Robot navigation using KFLANN place field,"

in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Singapore, 2008.

[7] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*: SIAM, 2007.

[8] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2005.

[9] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: a brief survey," in *A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

[10] Y. Liu, R. Jin, and A. K. Jain, "BoostCluster: boosting clustering by pairwise constraints," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 450-459.

[11] S. Basu, "Semi-supervised clustering: Learning with limited user feedback," The University of Texas at Austin, TX 2003.

[12] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 59-68.

[13] S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised clustering by seeding,"

in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML'2002)*, 2002, pp. 27-34.

[14] D. Cohn, R. Caruana, and A. McCallum, "Semi-supervised clustering with user feedback," in *Constrained Clustering: Advances in Algorithms, Theory, and Applications, S. Basu, I. Davidson, K. Wagstaff, Eds.*: CRC Press, 2008, pp. 17-31.

[15] B. J. Kronenfeld, "Implications of a data reduction framework to assignment of fuzzy membership values in continuous class maps," *Spatial Cognition & Computation,* vol. 3, pp. 223-239, 2003.

[16] S. Zhong and J. Ghosh, "A unified framework for model-based clustering," *The Journal of Machine Learning Research,* vol. 4, pp. 1001-1037, 2003.

[17] 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 International Conference on Knowledge Discovery and Data Mining*, 1996.

[18] A. Bouchachia and W. Pedrycz, "Data clustering with partial supervision," *Data Mining and Knowledge Discovery,* vol. 12, pp. 47-78, 2006.

[19] Y. Kamiya, T. Ishii, S. Furao, and O. Hasegawa, "An online semi-supervised clustering algorithm based on a self-organizing incremental neural network," in *Proceedings of International Joint Conference on Neural Networks IJCNN*, Orlando, FL, 2007, pp. 1061-1066.

[20] N. Cebron and M. R. Berthold, "Mining of cell assay images using active

semi-supervised clustering," in *Proceedings of the Fifth IEEE International Conference on Data Mining ICDM, Workshop on Computational Intelligence in Data Mining*, Houston, TX, 2005, pp. 63–69.

[21] N. Cebron and M. R. Berthold, "Adaptive fuzzy clustering," in *Annual meeting of the North American Fuzzy Information Processing Society, NAFIPS 2006.* , 2006, pp. 188-193.

[22] W. Pedrycz and J. Waletzky, "Fuzzy clustering with partial supervision," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 27, pp. 787-795, 1997.

[23] C. F. Eick, A. Rouhana, A. Bagherjeiran, and R. Vilalta, "Using clustering to learn distance functions for supervised similarity assessment," *Engineering Applications of Artificial Intelligence,* vol. 19, pp. 395-401, 2006.

[24] J. Sinkkonen and S. Kaski, "Clustering based on conditional distributions in an auxiliary space," *Neural Computation,* vol. 14, pp. 217-239, 2002.

[25] F. Wang, T. Li, and C. Zhang, "Semi-supervised clustering via matrix factorization," in *Proceedings of SIAM Conference on Data Mining SDM*, 2008.

[26] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing semi-supervised clustering: a feature projection perspective," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, USA, 2007, pp. 707-716.

[27] H. Chang and D. Y. Yeung, "Locally linear metric adaptation with application to semi-supervised clustering and image retrieval," *Pattern Recognition,* vol. 39, pp. 1253-1264, 2006.

[28] H. Chang and D. Y. Yeung, "Locally linear metric adaptation for semi-supervised clustering," in *Proceedings of the 21th International Conference on Machine Learning*, Banff, Alberta, Canada 2004, pp. 153-160.

[29] H. Chang, D. Y. Yeung, and W. K. Cheung, "Relaxational metric adaptation and its application to semi-supervised clustering and content-based image retrieval," *Pattern recognition,* vol. 39, pp. 1905-1917, 2006.

[30] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the 21th International Conference on Machine learning*, Banff, Alberta, Canada, 2004, pp. 81-88.

[31] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-Means clustering with background knowledge," in *Proceedings of 18th International Conference on Machine Learning*, 2001, pp. 577-584.

[32] K. L. Wagstaff, "Value, cost, and sharing: open issues in constrained clustering," *Lecture Notes in Computer Science,* vol. 4747, pp. 1-10, 2007.

[33] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," MSR-TR-2000-65, Microsoft Research 2000.

[34] A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng, "Constraint-based

clustering in large databases," *Lecture Notes in Computer Science,* vol. 1973, pp. 405-419, 2001.

[35] B. Dai, C. Lin, and M. Chen, "On the techniques for data clustering with numerical constraints," in *Proceedings of the SIAM International Conference on Data Mining*, 2003.

[36] A. Banerjee and J. Ghosh, "On scaling up balanced clustering algorithms," in *Proceedings of SIAM International Conference on Data Mining SDM*, 2002, pp. 333–349.

[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*: MIT Press, Cambridge, MA, USA, 1998.

[38] L. P. Wong and L. P. Tay, "Centroid stability with K-Means Fast Learning Artificial Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks* 2003.

[39] R. C. K. Ho, A. L. P. Tay, and X. Zhang, "Progressive learning paradigms using the parallel K-Iterations Fast Learning Artificial Neural Network," in *International Joint Conference on Neural Networks IJCNN* 2007, pp. 552-557.

[40] L. P. Tay, J. M. Zurada, L. P. Wong, and J. Xu, "The Hierarchical Fast learning Artificial Neural Network (HieFLANN)—an autonomous platform for hierarchical neural network construction," *IEEE Transactions on Neural Networks,* vol. 18, pp. 1645-1657, 2007.

[41] A. Bagherjeiran, C. F. Eick, and R. Vilalta, "Adaptive clustering: Better representatives with reinforcement learning," University of Houston 2005.

[42] A. Bagherjeiran, C. F. Eick, C. S. Chen, and R. Vilalta, "Adaptive clustering: obtaining better clusters using feedback and past experience," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005, pp. 565-568.

[43] W. Barbakh and C. Fyfe, "Clustering with reinforcement learning," in *Intelligent Data Engineering and Automated Learning - IDEAL 2007*. vol. 4881: Springer, 2007, pp. 507-516.

[44] A. Likas, "A reinforcement learning approach to online clustering," *Neural Computation,* vol. 11, pp. 1915-1932, 1999.

[45] C. H. Oh, E. Ikeda, K. Honda, and H. Ichihshi, "Parameter specification for fuzzy clustering by Q-learning," in *Proceedings of the International Joint Conference on Neural Networks IJCNN* Como, Italy 2000, pp. 9-12

[46] C. Fyfe and W. Barbakh, "Immediate reward reinforcement learning for clustering and topology preserving mappings," in *Similarity-Based Clustering*. vol. 5400/2009: Springer, 2009, pp. 35-51.

[47] D. J. Evans and L. P. Tay, "Fast learning artificial neural networks for continuous input applications," *Kybernetes,* vol. 24, pp. 11-23, 1995.

[48] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics,*

and Image Processing, vol. 37, pp. 54-115, 1987.

[49] L. P. Wong, "Hierarchical Clustering Using K-iterations Fast Learning Artificial Neural Networks (KFLANN)," Nanyang Technological University, Singapore, 2005.

[50] C. R. Rao, *Linear Statistical Inference and Its Applications, 2nd. Ed.*: Wiley, NY, 2001.

[51] A. Papoulis, *Probability, Random Variables and Stochastic Processes, 2nd ed.* New York: McGraw-Hill, 1984.

[52] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research,* vol. 4, pp. 237-285, 1996.

[53] S. Dreyfus, "Richard Bellman on the birth of dynamic programming," *Operations Research,* vol. 50, pp. 48-51, 2002.

[54] C. Watkins, "Learning from delayed rewards," King's College London, 1989.

[55] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed.: Prentice Hall, NJ, 2002.

[56] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning problems," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 13, pp. 834-847, 1983.

[57] R. A. McCallum, "Hidden state and reinforcement learning with instance-based state identification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 26, pp. 464-473, 1996.

[58] G. Shani, "A Survey of Model-Based and Model-Free Methods for Resolving perceptual aliasing," *Department of Computer Science, Ben-Gurion University,* 2004.

[59] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]," Irvine, CA: University of California, School of Information and Computer Science, 2007.

[60] L. Moreno, S. Garrido, and D. Blanco, "Mobile robot global localization using an evolutionary MAP filter," *Journal of Global Optimization, Springer Netherlands,* vol. 37, pp. 381-403, 2007.

[61] D. Filliat and J. A. Meyer, "Global localization and topological map-learning for robot navigation," *Proceedings of the 7th International Conference on Simulation of Adaptive Behavior on From Animals To Animats, B. Hallam, D. Floreano, J. Hallam, G. R. Hayes, and J. Meyer, Eds. Cambridge, MA: MIT Press,* pp. 131-140, 2002.

[62] D. Filliat and J. A. Meyer, "Map-based navigation in mobile robots: I. A review of localization strategies," *Cognitive Systems Research,* vol. 4, pp. 243-282, 2003.

[63] S. Thrun, "Robotic mapping: a survey," in *Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nebel, Eds.* CA: Morgan Kaufmann, 2003.

[64] E. Save, L. Nerad, and B. Poucet, "Contribution of multiple sensory information to place field stability in hippocampal place cells," *Hippocampus* vol. 10, pp. 64–76, 2000.

[65] K. Louie and M. A. Wilson, "Temporally structured REM sleep replay of awake Hippocampal ensemble activity," *Neuron,* vol. 29, pp. 145-156, 2001.

[66] D. Sheynikhovich, R. Chavarriaga, T. Strosslin, and W. Gerstner, "Spatial representation and navigation in a bio-inspired robot," *Biomimetic Neural Learning for Intelligent Robots,* vol. 3575, pp. 245-264, 2005.

[67] N. Burgess, A. Jackson, T. Hartley, and J. O'Keefe, "Predictions derived from modelling the hippocampal role in navigation," *Biological Cybernetics,* vol. 83, pp. 301-312, 2000.

[68] A. Arleo, F. Smeraldi, S. Hug, and W. Gerstner, "Place cells and spatial navigation based on vision, path integration, and reinforcement learning," *Advances in Neural Information Processing Systems,* 2001.

[69] G. Shani, "A survey of model-based and model-free methods for resolving perceptual aliasing," *Ben-Gurion University,* 2004.

[70] A. Jonsson and A. G. Barto, "Automated State Abstraction for Options using the U-Tree Algorithm," *Advances in neural information processing systems,* vol. 13, pp. 1054-1060, 2001.

[71] A. W. Moore and C. Atkeson, "Prioritized sweeping: Reinforcement learning with

less data and less time," *Machine Learning,* vol. 13, pp. 103-130, 1993.