Extreme Learning Machines for Clustering and Classification: An Empirical Study

A thesis submitted during 2016 to the University of Hyderabad in partial fulfillment of the award of a **Ph.D. degree** in School of Computer and Information Sciences

by

Abobakr Khalil Nasr Al-Shamiri

School of Computer and Information Sciences

University of Hyderabad
P.O. Central University, Gachibowli
Hyderabad – 500 046
Telangana
India

May 2016

CERTIFICATE

This is to certify that the thesis entitled **Extreme Learning Machines for Clustering and Classification: An Empirical Study** submitted by **Abobakr Khalil Nasr Al-Shamiri** bearing **Reg. No. 11MCPC05** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under our supervision and guidance which is a plagiarism free thesis.

The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof. Bapi Raju Surampudi
(Supervisor)
School of Computer and
Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

Dr. Alok Singh
(Supervisor)
School of Computer and
Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

DECLARATION

I, **Abobakr Khalil Nasr Al-Shamiri**, hereby declare that this thesis entitled **Extreme Learning Machines for Clustering and Classification: An Empirical Study** submitted by me under the guidance and supervision of **Prof. Bapi Raju Surampudi** and **Dr. Alok Singh** is a bonafide research work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Library is enclosed.

Date :

Name: Abobakr Khalil Nasr Al-Shamiri

Signature of the Student:

Reg. No.: 11MCPC05

Signature of the Supervisors:

Abstract

From the early stages of neural networks, there have been several attempts at developing training algorithms that are capable of training neural networks in reasonable time and obtaining good generalization performance. Traditional gradient based learning algorithms, such as back-propagation (BP), for training feedforward neural networks usually get stuck in local minima and take much time to converge. These algorithms require all the weights and biases of the networks to be tuned iteratively. The extreme learning machine (ELM), has gained popularity in recent years, is a new training method for randomized feedforward neural networks. In ELM, the hidden layer parameters are randomly initialized and remain fixed during the learning process and the output weights are analytically determined. Due to its fast training speed and broad applicability, the ELM framework has become very popular in the past decade.

This thesis is focused on extreme learning machine (ELM) and its applications to clustering and classification problems. In this regard five contributions are made which are: (i) integration of ELM with classical K-means algorithm for clustering, (ii) development of artificial bee colony (ABC) based clustering approach which performs clustering in ELM feature space, (iii) Combining ELM with a dimensionality reduction technique, namely, random projection (RP) for classification and clustering problems, (iv) development of two swarm intelligence techniques for optimizing ELM and (v) providing a comparative analysis of ELM and No-Prop algorithms for data classification.

First, a detailed discussion on ELM and research issues related to it are provided. Swarm intelligence techniques are described briefly. Recent controversy surrounding ELM is also briefly commented on. Next, a novel

ELM K-means method is developed. It is based on using ELM to project the data into high dimensional feature space and then performing clustering in this feature space using K-means algorithm. Thereafter, a hybrid approach called ELM-ABC is described which uses ABC algorithm to perform clustering in ELM feature space. This approach overcomes the limitations of ELM K-means method.

Next, we integrate ELM with random projection (RP) for data classification and clustering. There are two scenarios. One for high dimensional data and another for low dimensional data. Thereafter, we present two metaheuristic techniques, namely, artificial bee colony algorithm and invasive weed optimization (IWO) algorithm for optimizing the ELM input weights and hidden biases. The goal is to obtain a compact network which can produce good generalization performance with small number of neurons in the hidden layer.

Next, we compare the classification performance of ELM and No-Prop algorithms. The difference between ELM and No-Prop is that ELM uses analytical procedure to compute the output weights whereas No-Prop uses the iterative least mean square error (LMS) algorithm.

All experiments in the thesis are conducted using UCI benchmark data sets and other publicly available data sets so that future researchers can easily compare the performance of their approaches with those proposed in this thesis. The results successfully demonstrate the viability of the proposed approaches. Several directions for future research have also been suggested.

Acknowledgements

I would like to take this opportunity to express my sincere appreciation and gratitude to my supervisors **Prof. Bapi Raju Surampudi** and **Dr. Alok Singh** for their guidance and tireless help during my research. They have provided me with invaluable guidance, generous support and encouragement needed for the successful completion of my thesis. Their vast experience and profound knowledge have been a constant source for me throughout my work. I am very grateful to their patience and thoughtful consideration which helped me a lot, especially in those tough moments.

I am also grateful to my doctoral committee members Prof. K. N. Murthy, Prof. Chakravarthy Bhagvati and Dr. Vineet Padmanabhan Nair for their valuable suggestions which helped in improving the quality of my work. I am equally grateful to Dr. S. Durga Bhavani for her constant encouragement and motivation. I am thankful to the Dean of the School Prof. Arun Agarwal for making available all the necessary facilities.

I would like to express my sincere thanks to Dr. Ahmed Sultan Al-Hegami from Sana'a University for his support and several fruitful discussions.

I would never forget all the chats and beautiful moments I shared with some of my friends and classmates. In particular, I would like to thank Hamzah Ali Jamel (good friend of mine), Wael Al-Hakemi, Esam AL-Jaberi, Dr. Fahim Baggash, Dr. Mokhtar Al-Amrani, Dr. Amer Farea Sallam, Dr. Abdulmalik Mansour, Mohammed Nasher, Abdulrahman Osman, Abdu Mahyoub, Zaid Al-Huda, Eyad Himdiat, Ahmed Ali Gumaan El-Shekeil, Abdulsalam Al-Ammari, Hassan Al-Shehari, Ahmed Al-Haidari,

Antar Shaddad, Dhiaya Al-Sanawi, Mohammed Al-Sanawi, Mokhtar Al-Khulaidi, Yousuf Al-Dubaei and Bassam Al-Dubaei for their support during these stressful and difficult moments.

A special thanks goes to my labmates, B. Jayalakshmi, Sachchida Nand Chaurasia, Venkatesh Pandiri and Gaurav Srivastava for sharing ideas and many interesting discussions.

I am thankful to the Ministry of Higher Education and Scientific Research, Yemen for financial support.

I am extremely grateful to my parents as well as my uncle Ali Nasr Al-Shamiri for their love, support and encouragement. Finally, my love and deepest gratitude go to my wife for her support, care and patience during my research.

Abobakr Khalil Nasr Al-Shamiri

Contents

CONTENTS

List of Figures

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Extreme learning machine

Extreme learning machine (ELM) is a new learning algorithm for single hidden layer feedforward neural networks (SLFNs). It initializes weights of hidden neurons randomly and determines the output weights analytically by making use of Moore-Penrose (MP) generalized inverse [4, 5, 6]. Unlike the slow gradient descent-based learning algorithms for SLFN, where all the parameters of the network need to be tuned iteratively, ELM requires no iteration when determining the SLFN parameters as it initializes the hidden layer parameters randomly and keep them fixed during the learning process and determines the output weights analytically. The ELM hidden layer nonlinearly transforms the input data into a high dimensional space called ELM feature space. This transformation often makes the input data more separable in the ELM feature space, and hence simplifies the solution of the underlying or associated tasks. ELM algorithm not only learns at extremely fast speed, but also obtains good generalization performance [7].

Suppose we are given a set of training examples $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, then the output of SLFNs with L hidden neurons and activation function $g(x)$ can be represented as [5]:

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{y}_j, \quad j = 1, \ldots, N. \tag{1.1}$$

where $\mathbf{w}_i = [w_{i1}, \ldots, w_{id}]^T$ and b_i are the input weight vector and the bias of the hidden

neuron, respectively. $\beta_i = [\beta_{i1}, \ldots, \beta_{im}]^T$ is the output weight vector. $\mathbf{w}_i \cdot \mathbf{x}_j$ is the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$. With that standard SLFNs, the parameters $\beta_i, i = 1, \ldots, L$ can be estimated such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \ j = 1, \ldots, N. \tag{1.2}$$

Equation (1.2) can be written as [5]:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{1.3}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L} \tag{1.4}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{1.5}$$

$\mathbf{H}$ is the hidden layer output matrix of the network [8]. The computation of output weights $\boldsymbol{\beta}$ can be done using the following equation:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{1.6}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$ [9].

The illustration of the primal ELM architecture with L hidden neurons is shown in Figure 1.1, which is adapted from [1]. In ELM, the number of neurons in the hidden layer of the ELM should be large enough to achieve good generalization performance. The main steps of the ELM algorithm are given in Algorithm 1 [10].

In Algorithm 1, classification is achieved by providing the correct class labels as part of the matrix $\mathbf{T}$. In clustering, matrix $\mathbf{T}$ is not available so one can use the projected hidden layer output matrix $\mathbf{H}$ as part of clustering algorithm to achieve unsupervised clustering.

Algorithm 1: ELM Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons
$g(x)$: activation function

1 Initialize hidden layer parameters $(\mathbf{w}_i, b_i)$, $i = 1, \ldots, L$ randomly;
2 Compute the hidden layer output matrix $\mathbf{H}$;
3 Compute the output weight $\boldsymbol{\beta}$;

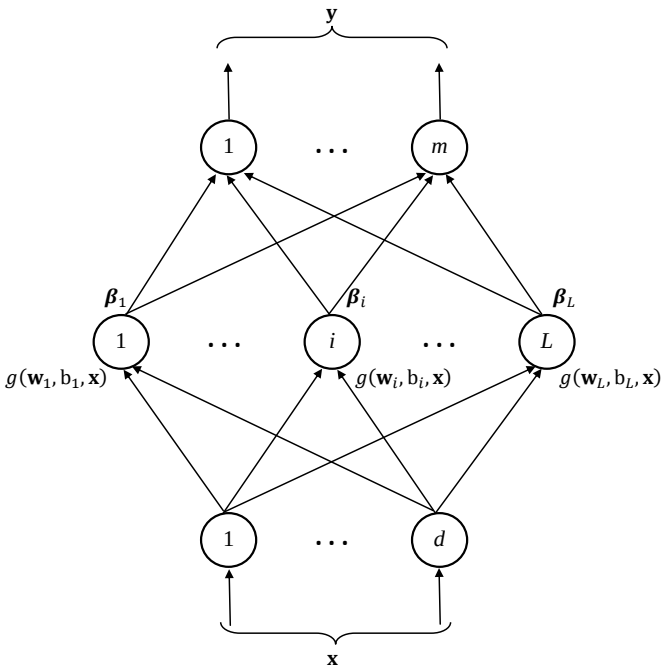

Figure 1.1: ELM network (Adapted from [1])

1.1.1 ELM Controversy

From the early stages of neural networks, there have been several attempts at single hidden layer feedforward neural networks (SLFNs) with randomized hidden layer [11, 12, 13, 14]. However, extensive investigation, viability and applications of SLFNs with random initialization of the hidden layer have been explored by research group at Nanyang Technological University (NTU), Singapore under the name extreme learning machines (ELM) [1, 4, 5, 6, 15, 16, 17, 18, 19, 20]. The research group at NTU, Singapore and other research groups have looked at ELM comprehensively and extensively

in variety of application domains [3, 7, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. Even though there is prior work exist on SLFNs with randomized hidden layer, the capabilities of SLFNs have been identified and popularized through the publications of NTU research group and other research groups on ELM. However, there seems to be some recent discussion on inadequate credit given to the past attempts of SLFNs with randomized hidden layer [36, 37, 38]. In general, the history of machine learning research and in particular neural networks research shows how ideas have been independently discovered by several people either simultaneously or at different points in time. For example, the idea of training neural networks through back propagation of errors has been discovered by several people at different points of time (i.e., Werbos in his PhD thesis has this idea in 1974 [39] but it was rediscovered again in 1982 by Parker [40], in 1985 by LeCun [41] and in 1986 by Rumelhart et al. [42]). Recently, Bernard Widrow, not knowing about ELM, has again proposed neural network architecture similar to ELM with different iterative scheme of learning [43, 44, 45]. This again goes to show that it is possible for ideas to be independently discovered. It is worth mentioning that one of the past attempts at randomized neural networks, namely, Random Vector Functional Link (RVFL) networks in which there is a direct connection from the input layer to the output layer and the hidden layer weights and biases are randomly initialized [14], has got more attension from researchers recently and several papers have been published on the applications of RVFL networks for classification, prediction and other problems [46, 47, 48, 49].

1.2 Swarm Intelligence

Over the last two decades, metaheuristic techniques inspired by intelligent behavior of swarms have got increased attention from researchers in the field of optimization. These techniques are based on simulating the collective intelligent behavior of the self-organizing individuals. Swarm refers to a group of natural agents, such as honey bees and ants, or artificial agents that exhibit collective behavior. All agents in a swarm behave stochastically based on their local perception of the neighborhood. This stochastic behavior is performed in a parallel (all agents perform tasks simultaneously) and distributed (no centralized control) manner. These self-organized agents do not have any

knowledge about group behavior or the global pattern. However, local rules and interactions between self-organized agents lead to the emergence of collective intelligent behavior. Such a collective intelligence is termed *swarm intelligence*. This term was first introduced by Beni and Wang [50] in the context of cellular robotic systems.

The swarm in Nature has two major characteristics, viz. division of labor and self-organization. These two characteristics are necessary and sufficient to obtain swarm intelligent behavior. Division of labor enables the swarm to perform a variety of tasks simultaneously. Each task is assigned to specialized agents. For example, some agents have a duty of searching and bringing food while other agents are responsible for protecting the nest. This kind of labor division can be observed clearly in honey bees and ants swarm. Self-organization helps the swarm to achieve global level behavior by means of low level interactions and without any global supervision. Self-organization relies on the following characteristics [51]:

1. *Positive feedback* promotes the formation of convenient structures composed of many agents. Reinforcement and recruitment, which can be respectively observed in pheromone trail laying in ants and dances in honey bees, are examples of positive feedback.

2. *Negative feedback* is required in order to stabilize the collective pattern. Food source exhaustion during bee foraging and pheromone evaporation in ants are examples of negative feedback.

3. *Fluctuations* refer to random fluctuations in the behavior of individuals belonging to a swarm. This randomness often helps in finding new solutions.

4. *Opportunities for multiple interactions* is required in a swarm. Normally, in a single interaction, exchange of information takes place between few (usually two) individuals. Hence, multiple interactions are necessary to spread the information globally.

In this thesis, we have used two swarm intelligence inspired techniques, viz. artificial bee colony (ABC) algorithm and invasive weed optimization (IWO) algorithm. In the following sections, ABC and IWO algorithms are described.

1.3 Artificial bee colony (ABC) algorithm

The artificial bee colony (ABC) algorithm, proposed by Karaboga [52] for optimizing numerical problems, is a relatively new population-based meatheuristic technique which simulates the intelligent behavior performed by the honey bees while seeking food around their hives. Usually, the colony of bees consists of three types of bees, namely, employed bees, onlookers and scouts. The employed bees are responsible for exploiting the food sources and bringing the loads of nectar to the hive. They also gather the relevant information about their food sources such as location, quality and quantity of nectar, and share this information with the onlooker bees by dancing in a designated area of the hive. The nature and duration of a dance depends on the characteristics of the food source that the dancing bee is currently exploiting. Onlooker bees observe numerous dances before choosing a food source. Onlooker bees choose a food source to exploit with a probability directly proportional to its quality and become employed. Therefore, good food sources get more onlookers in comparison to the poor ones. Scout bees are those bees which look for new food sources in the vicinity of the hive. Once a scout bee finds a food source, it becomes employed. When a food source is exhausted, each of its associated employed bee abandons it and becomes either a scout or an onlooker. Equating the foraging behavior of bees with the search process carried out by a metaheuristic techniques, we can say that scout bees do the *exploration*, whereas employed bees and onlooker bees are responsible for *exploitation*.

Inspired by this intelligent foraging behavior of honey bee swarm, Karaboga [52] proposed the ABC algorithm which has been extended further [53, 54, 55, 56, 57, 58, 59, 60]. ABC algorithm also divides the (artificial) bees into same three types, viz. employed, onlooker and scout with functions similar to those explained above. In ABC algorithm, a food source represents a possible solution to the problem to be optimized and the nectar amount of a food source corresponds to the quality (fitness) of the solution represented by that food source. Usually, half of the colony consists of employed artificial bees and the other half contains the onlooker bees. Only one employed bee is assigned to each food source. Therefore, the number of the employed bees or the onlooker bees is equal to the number of food sources [52, 54]. In ABC algorithm, the employed bee of an exhausted food source always becomes a scout. A

new food source is generated for this scout and it is immediately turned into employed by associating it with this newly generated food source.

The ABC algorithm is an iterative algorithm, and it starts by generating randomly distributed initial solutions (food sources), evaluating their fitness and assigning the employed bees to the food sources. Only one employed bee is assigned to each food source. After initialization, the algorithm tries to improve the population of solutions and finds the optimal by subjecting the population to repeated iterations of three search phases, viz. employed bee phase, onlooker bee phase, and scout bee phase. A simple scheme of the original ABC algorithm is shown in Algorithm 2.

Employed bee phase: Each employed bee determines a new food source in the vicinity of its originally assigned (or old) food source. Once the new food source is determined, its nectar amount (fitness) is evaluated. If the fitness of the new food source is better than that of the old one, the employed bee replaces the old food source with the new one, otherwise the old one is retained. The exact method of determining a new food source varies from one problem to the other and even for the same problem from one implementation of ABC algorithm to the other.

Onlooker bee phase: Once all employed bees have finished the process of determining a new food source and have taken a decision to move to newly determined food source or not, they share the information about their food sources with the onlooker bees. An onlooker bee evaluates the fitness information taken from all employed bees and chooses a food source to exploit with a probability proportional to its fitness. The higher the fitness of the food source is, the more the probability of it being selected by the onlooker bees. Once the onlooker bee has selected her food source, she finds a new food source in the neighborhood of the selected food source. As in the case of the employed bee, if the new food source has a better quality than the old one, it will replace the old one, otherwise, the old one is retained.

Scout bee phase: If the quality of the food source cannot be improved further over a predetermined number of attempts *limit*, then the food source is deemed to be exhausted and the employed bee associated with that food source leaves it to become a scout. A new food source is generated for this scout and it is immediately turned into employed by associating it with this newly generated food source. Usually, the new food source for scout bee is generated randomly from scratch. However, some ABC algorithm versions generate this food source by suitably perturbing either the current

or the best solution. These three phases are repeated until the termination condition is met. The main steps of the algorithm are given below:

1. Randomly generate the population of initial food sources.
2. Evaluate the fitness of each member of the population.
3. Generate new food sources for the employed bees and evaluate their quality.
4. Perform a comparison between the old and new food sources and select the better ones.
5. Calculate the probabilities of the current food sources and assign onlooker bees to them.
6. Generate new food sources for the onlooker bees and evaluate their quality.
7. Perform a comparison between the old and new food sources and select the better ones.
8. Replace the abandoned food source with the new one discovered by scout bee.
9. Memorize the best food source found so far.
10. If the termination condition is not met, go to step 3, otherwise stop the algorithm.

Algorithm 2: The artificial bee colony algorithm

```
1 Initialize the population of food sources;
2 Evaluate the population;
3 repeat
4 |   Employed Bees Phase;
5 |   Onlookers Bees Phase;
6 |   Scouts Bees Phase;
7 |   Memorize the best food source achieved so far;
8 until termination condition is met;
```

1.4 Invasive weed optimization (IWO) algorithm

Invasive weed optimization algorithm (IWO) is recently proposed metaheuristic which mimics the colonizing behavior of invasive weeds. It is devoped by Mehrabiana and Lucas [61]. The rapid growth and reproduction of these weeds pose a threat to the cultivated plants. Weeds have great surviving capability due its ability to adapt to the change in environment. Several different variants and improvements of the original IWO algorithm [61] have been proposed in the literature [62, 63, 64, 65, 66, 67, 68, 69].

In IWO algorithm, each weed represents a possible solution to the problem to be optimized and the fitness of a weed corresponds to the fitness of the solution represented by that weed. The IWO algorithm is an iterative algorithm. It starts with a finite number of weeds (solutions) being randomly generated and dispersed over the search area. Then the fitness of each weed is evaluated. After initialization, the algorithm tries to find the optimal solution by subjecting the population of solutions to repeated cycles of three search steps, viz. reproduction, spatial dispersal and competitive exclusion. In every iteration, the weeds produce seeds (child solutions) depending on their fitness. The produced seeds are randomly distributed over the search area, usually around the producing weeds, by normally distributed random numbers with mean equal to zero and varying variance. The process of reproduction and spatial dispersal continues until the maximum number of weeds is reached. Once the maximum number of weeds is reached, only the fittest weeds are allowed to produce and the weeds with least fitness are eliminated (competitive exclusion). The reproduction, spatial dispersal and competitive exclusion steps are repeated until the termination condition is met.

The steps of IWO algorithm are explained in detail as follows:

Step 1: Population Initialization

A set of initial solutions dispersed over the search space are generated randomly.

Step 2: Reproduction

The number of seeds a weed can produce depends on its own, the highest and lowest fitness of the colony. The higher the fitness of a weed is, the more seeds it produces. The number of seeds produced by a weed is calculated as follows [62]:

$$N_{seed} = \frac{fit - fit_l}{fit_h - fit_l} (S_{max} - S_{min}) + S_{min} \tag{1.7}$$

where fit is the fitness of the current weed. fit_h and fit_l correspond to the highest and lowest fitness of the population, respectively. S_{max} and S_{min} are respectively the maximum and the minimum number of seeds a weed can produce.

Step 3: Spatial dispersal

The produced seeds are randomly distributed over the search area, usually around the producing weeds, by normally distributed random numbers with zero mean and varying variance. The standard deviation (σ) at every iteration will be reduced from a pre-defined initial value ($\sigma_{initial}$) to a pre-defined final value (σ_{final}) and is given by the following expression:

$$\sigma_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}}\right)^{nmi} (\sigma_{initial} - \sigma_{final}) + \sigma_{final} \tag{1.8}$$

where σ_{iter} is the standard deviation at the current iteration. $iter_{max}$ is the maximum number of iterations and nmi is the nonlinear modulation index. Usually, nmi is set to 3.

Step 4: Competitive exclusion

Initially, all weeds will reproduce fast and all seeds (newly produced weeds) will be included in the population until the maximum number of weeds WN_{max} in the colony is reached. Once this happens, only the fittest weeds are allowed to replicate and survive to the next generation and the weeds with least fitness are eliminated.

1.5 Research issues related to ELM and thesis contributions

As described in Section 1.1, ELM consists of three major features that this thesis addresses as shown in Figure 1.2.

1.5.1 Thesis contributions related to ELM-based clustering

First feature is related to the types of applications ELM can be applied. Upon looking at the literature in 2013 (when the research work on this thesis began), we found that while ELM has been extensively used for classification and regression applications [1, 3, 10, 70, 71, 72], the clustering problem using ELM had not been addressed. So in this thesis, we have addressed clustering applications extensively using K-means and

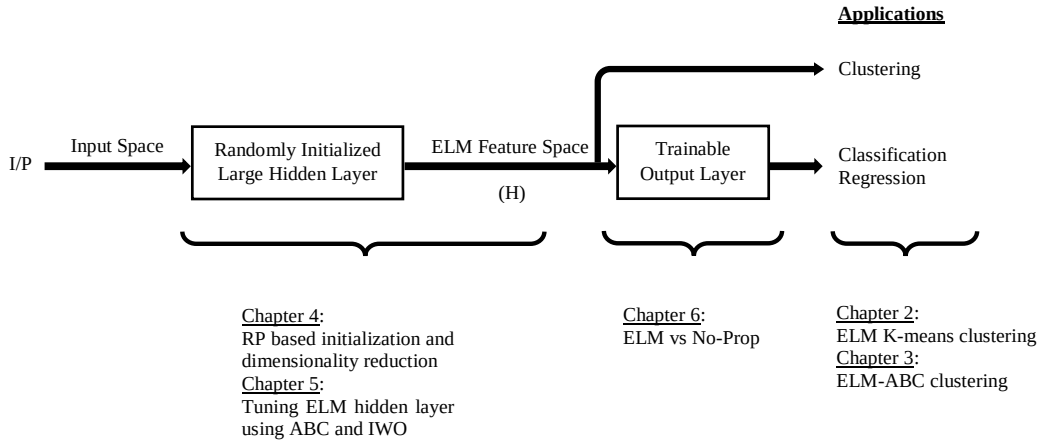

Figure 1.2: Block diagram of generic ELM depicting various processes. Contributions of the thesis are listed in the bottom row.

artificial bee colony algorithms in Chapter 2 and Chapter 3, respectively. It is worth mentioning that, when we started this work on ELM in 2013, variant of ELM for clustering has not been investigated. However, later on several papers have been published [21, 73, 74] which address the clustering problem in ELM feature space but our paper [22] was one of the first papers that attempted ELM for clustering.

1.5.2 Thesis contributions related to ELM hidden layer

Second feature is focused on ELM hidden layer size and its random initialization. In general, ELM requires large hidden layer in order to obtain good generalization performance. Numerous approaches exist in the literature to tackle the problem of random initialization and controlling the size of ELM hidden layer. Several researchers tried to address this issue by proposing pruning methods [72, 75], incremental methods [6, 76, 77] or evolutionary methods [3]. However, we have identified that there is scope for further improvement and in this thesis we have proposed two ways of addressing this issue. The first way uses random projection (RP) method and addresses both classification and clustering problems in Chapter 4. The second way uses two swarm intelligence techniques, viz. artificial bee colony (ABC) algorithm and invasive weed

optimization (IWO) algorithm and addresses classification problem in Chapter 5.

1.5.3 Thesis contributions related to learning in ELM

Third feature is related to the trainable output layer of ELM. Typically, ELM output layer is trained for regression and classification problems. The training of the output layer can be done, in batch mode learning, using analytical procedure in which the pseudo-inverse of the hidden layer output matrix **H** is multiplied with the desired output in order to find the solution. This is batch type of learning. However, the training process could be done iteratively. Recently, Widrow et al. [43] proposed iterative approach called No-Propagation (No-Prop) with the same features as a single hidden layer feedforward neural network without propagating errors to the hidden layer. So in this thesis we actually compare ELM and No-Prop algorithms head-to-head and give some recommendations in Chapter 6.

1.6 Outline of the thesis

In this thesis, we have first investigated the performance of ELM in combination with K-means algorithm, artificial bee colony algorithm, random projection for clustering/classification. Next, we have tuned the input weights and hidden biases of ELM using two metaheuristic techniques, viz. artificial bee colony algorithm and invasive weed optimization algorithm so as to obtain good generalization performance with less number of neurons. At the last, a comparative analysis based on empirical studies regarding the stability and convergence performance of ELM and No-Prop algorithm for data classification is provided. This thesis is divided into 7 chapters beginning with this introductory chapter. In the following, we summarize the content of remaining chapters.

Chapter 2 deals with the problem of clustering of patterns that are not linearly separable in the input space. We have incorporated the extreme learning machine method into K-means algorithm and proposed a novel K-means clustering with the ELM feature space. The ELM method is used to project the data into a high dimensional feature space and K-means algorithm performs clustering within this feature space by making use of the Euclidean distance in this feature space to measure the similarity among the

objects. The experimental results demonstrate that integration of ELM method with K-means algorithm improves the quality of clustering.

Chapter 3 presents a hybrid approach combining artificial bee colony algorithm and extreme learning machine method for data clustering. The method builds on ELM projection of input data into a high dimensional feature space and followed by unsupervised clustering using ABC algorithm. While ELM projection facilitates separability of clusters, a metaheuristic technique such as ABC algorithm overcomes problems of dependence on initialization of cluster centers and convergence to local minima suffered by conventional algorithms such as K-means. We have evaluated the proposed ELM-ABC algorithm on wide range of real world benchmark data sets. Computational results show that ELM-ABC gives significantly better time-accuracy tradeoff compared to ELM K-means algorithm. These results also demonstrate that the integration of ELM method with ABC algorithm improves the quality of clustering performed by the ABC algorithm itself.

Chapter 4 presents a systematic study on the application of Random projection (RP) method in conjunction with ELM for both low and high dimensional data classification and clustering. Random projection, as a simple and powerful technique for dimensionality reduction, is used for projecting high dimensional data into low dimensional subspace while ensuring that distances between data points are approximately preserved. This study is based on the following intuition: the very fact that RP technique can do dimensionality reduction without distorting significantly the structure of data, can be used for different scenarios. First, for high dimensional data, RP can be used to reduce the dimensionality by shrinking the number of neurons in the ELM hidden layer, as the solution may lie in a lower dimensional subspace. The resulting projections can be utilized for clustering and classification in the reduced space. Second, for low dimensional data, we can use ELM to project the data into ELM feature space, where we expect the data to be linearly separable, and then we use RP to bring the data down to a lower space while preserving the linear separability of data. The second scenario tackles the problem of nonlinear separability of data in the input space and reduces the computational cost of classification and clustering in the ELM feature space by projecting the data into a lower space while preserving the linear separability of the

data. Results demonstrate that the integration of ELM with RP for data classification and clustering not only obtains satisfying results, but also provides a potential tradeoff between generalization performance and computational complexity.

Chapter 5 presents two swarm intelligence inspired metaheuristic techniques, viz. artificial bee colony (ABC) algorithm and invasive weed optimization (IWO) algorithm for selecting the ELM input weights and hidden biases. ELM tends to produce good generalization performance with large number of hidden neurons due to the random determination of the input weights and hidden biases. Since ELM requires large number of hidden neurons, there may exist some suboptimal input weights and hidden biases. For some applications, the high number of neurons in the hidden layer renders the ELM respond slowly to unseen testing data. To overcome these shortcomings, we have proposed two metaheuristic techniques. The first approach is based on ABC algorithm in which the input weights and hidden neuron biases are selected using an ABC algorithm and the output weights are analytically determined using Moore-Penrose (MP) generalized inverse, whereas in the second approach, an IWO algorithm is used to optimize the hidden layer parameters and the output weights are calculated using MP generalized inverse. The proposed approaches are tested on different benchmark classification data sets and results show that the proposed approaches obtain good generalization performance in comparison to the other techniques reported in the literature.

Chapter 6 presents a comparative analysis of ELM and a newly proposed method called No-Prop algorithm [43] for data classification. In No-Prop algorithm, the weights of the hidden layer neurons are set and fixed with random values and the output weights are trained using least mean square error (LMS) algorithm. The difference between ELM and No-Prop algorithm lies in the training method for the output layer. While latter uses the LMS gradient algorithm to train the output weights, ELM determines the output weights using Moore-Penrose generalized inverse. The computational experiments show that No-Prop algorithm is stable and provides good generalization performance, but it is slow. The experiments also show that ELM is fast but it provides poor results if it is used without regularization parameter. ELM with regularization parameter outperformed No-Prop algorithm, but there is a cost involved in searching for the best value of the regularization parameter used with ELM.

The last chapter of the thesis, i.e., chapter 7 summarizes the contributions of the thesis. It also outlines some directions for future research.

Chapter 2

Clustering in ELM feature space using K-means

2.1 Introduction

Clustering is the method that discovers natural grouping(s) of a set of patterns, points, or objects. The goal of data clustering, also known as cluster analysis, is to partition a data set into clusters (also called groups) of similar objects on the basis of a *similarity* (or *dissimilarity*) measure, thereby minimizing the similarities between objects belonging to different clusters and maximizing the similarities between objects belonging to the same cluster, i.e., the objects within a group are more similar to each other (high intra-cluster similarity) than objects belonging to different groups (low inter-cluster similarity) [78][79][80]. There are several *similarity* (or *dissimilarity*) measures, and, the choice of an appropriate measure depends on the data under analysis and the purpose of the analysis. Clustering has been successfully applied in a large variety of applications, for example, image segmentation, object and character recognition, document retrieval, remote sensing, data compression, etc.

Clustering techniques can be broadly classified into two categories based on the structure of abstraction, viz. hierarchical clustering and partitional clustering [79][80]. Hierarchical clustering techniques group data objects with a sequence of partitions, either from singleton clusters towards a cluster containing all individuals or vice versa. The result is a hierarchical structure of partitions known as *dendrogram* in which each partition is nested within the partition at the next level in the hierarchy. Hierarchical

methods can be either agglomerative or divisive. Agglomerative algorithms place each pattern in its own cluster at the outset and then successively merge (or *agglomerate*) pairs of clusters until all clusters have been merged into a single cluster that contains all patterns or the desired objective is obtained. Divisive algorithms begin with all patterns placed in one cluster and then proceed by splitting a cluster into smaller clusters recursively until individual patterns are reached or a stopping criterion is met [81][82]. Partitional clustering, on the other hand, assigns a set of patterns into specified or estimated number of clusters without any hierarchical structure. An important problem in partitional clustering is to partition a given set of data into specified number of clusters so that some criterion function is optimized (maximized or minimized). One of the most widely used criteria is to minimize the sum of squared errors. K-means is one of the simplest and the most well known algorithms of the partitional methods [83].

K-means algorithm is distance-based because the similarity of the patterns is determined by computing Euclidean distance or any other distance measure. Effectiveness of distance-based clustering algorithms depends on nature of data. Data with ellipsoidal or hyper-spherical distribution are easy to deal with. On the other hand, the distance-based clustering algorithms will fail if the separation boundaries between clusters are nonlinear. One of the approaches to handle this problem is to nonlinearly transform the input data into a high dimensional feature space, using kernel functions or extreme learning machine (ELM), and then perform clustering within this feature space [84]. Performing a nonlinear transformation of a set of nonlinearly separable patterns into a high dimensional feature space increases the probability of the linear separability of these patterns within the transformed space, thereby simplifying the associated structure of data and enabling easier clustering [85].

Kernel function defines a nonlinear transformation implicitly. This transformation increases the separability of the data by mapping them from their original input space to a high dimensional space called *feature space*. Suppose we are given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$, and a mapping function ϕ that maps the data $\mathbf{x}_i$ from the input space R^d to a new feature space F. The kernel function is defined as the dot product in the feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{2.1}$$

Some commonly used kernel functions are given below:

- Polynomial of degree p:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p, \qquad p \in N. \tag{2.2}$$

- Gaussian:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|^2}{2\sigma^2}\right), \ \sigma \in R. \tag{2.3}$$

- Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i \cdot \mathbf{x}_j + b), \ a, b \in R. \tag{2.4}$$

Kernel methods have become very popular in the past few years and several clustering methods have been modified incorporating kernels. Kernel-based clustering algorithms exploit the notion that performing a nonlinear transformation of a set of nonlinearly separable patterns into a higher dimensional feature space increases the possibility to separate these patterns linearly. The linear partitioning in this feature space corresponds to a nonlinear partitioning in the input space. Consequently, kernel-based clustering methods may achieve better generalization performance by working in this feature space. Various kernel-based clustering methods have been proposed in the literature. Girolami [85] proposed a kernel method for clustering in feature space. The method also provides estimation of the possible number of clusters using the kernel matrix. Scholkopf et al. [86] proposed kernel K-means method where the standard K-means algorithm was presented in the feature space by employing the kernel trick. The main drawbacks of the kernel K-means clustering method are the local minima and scalability. The latter drawback exists because of the need to compute the full kernel matrix. The size of this matrix is quadratic in the number of data points. Tzortzis and Likas [87] proposed the global kernel K-means algorithm to alleviate the local minima problem. This algorithm minimizes the clustering error in the feature space by finding near-optimal solutions. A solution to large scale kernel clustering is presented in [88][89]. Camastra and Verri [90] proposed a kernel method for clustering inspired by the classical K-means algorithm and based on one-class Support Vector Machine (SVM) description of a data set. An ant-based clustering algorithm integrated with

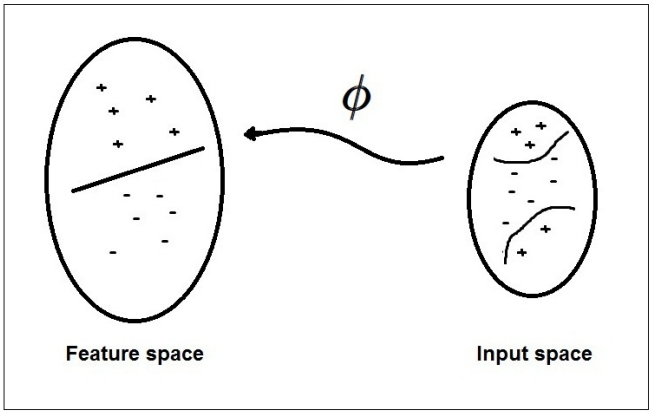

Figure 2.1: Mapping from input space to feature space. The mapping function ϕ maps the data from their original input space to a high dimensional feature space where the data are expected to be more separable.

the kernel method was presented in [84]. The integration of kernels with K-means, fuzzy K-means, SOM, Neural gas, and one-class SVM has been shown to be effective in improving the quality of clustering [80]. However, performance of kernel based methods depends on the choice of kernel function which is highly data specific. Most of these methods are also compute intensive, because of the need to compute the full kernel matrix. In addition, the most frequently used kernel, viz. RBF kernel [85] requires tuning of its width. Unlike the feature mapping in the kernel-based clustering methods which is implicitly defined by the use of the kernel functions and which is not computationally efficient, the ELM feature mapping is explicit, very intuitive and straightforward. Thus, compared to kernel-based clustering algorithms, clustering in ELM feature space is more convenient

In this chapter, we have incorporated the extreme learning machine (ELM) method into K-means algorithm and proposed a novel K-means clustering with the ELM feature space (ELM K-means). The ELM method is used to project the data into a high dimensional feature space and the K-means algorithm performs clustering within this feature space by making use of the Euclidean distance in this feature space to measure the similarity among the objects.

The remainder of this chapter is organized as follows: Section 2.2 introduces K-means algorithm. In Section 2.3, we present the proposed ELM K-means algorithm.

Experiments and their results are presented and discussed in Section 2.4. Finally, Section 2.5 concludes this chapter.

2.2 K-means algorithm

K-means algorithm can be regarded as an unsupervised learning algorithm which partitions the data set into a given number of clusters that based on some optimization measure. The clustering problem, which K-means algorithm is designed to solve, can be stated as follows: Given a representation of N patterns, find K clusters based on a measure of similarity such that the patterns within a cluster are more similar to each other (high intra-cluster similarity) than patterns belonging to different clusters (low inter-cluster similarity).

Let $X = \{\mathbf{x}_i, i = 1, \ldots, N\}$ be the the set of N patterns to be clustered into a set of K clusters, $C = \{c_k, k = 1, \ldots, K\}$. Typically $K \ll N$ and each pattern is a vector of dimension d ($\mathbf{x}_i \in R^d$). K-means algorithm finds a partition such that the squared Euclidean distance between the center of a cluster and the patterns in the cluster is minimized. Let μ_k be the mean of cluster c_k and it is defined as:

$$\mu_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in c_k} \mathbf{x}_i \tag{2.5}$$

where N_k is the number of patterns in cluster c_k.

The squared error between μ_k and the patterns in cluster c_k is defined as in [91]:

$$J(c_k) = \sum_{\mathbf{x}_i \in c_k} \| \mathbf{x}_i - \mu_k \|^2 \tag{2.6}$$

The main objective of the K-means algorithm is to minimize the sum of the squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in c_k} \| \mathbf{x}_i - \mu_k \|^2 \tag{2.7}$$

K-means is an iterative algorithm. It starts by initializing the clusters' centers randomly. In every iteration, each pattern is assigned to its closest cluster, based on

the distance between the pattern and the cluster center. The cluster centers in the next iteration are determined by computing the mean value of the patterns for each cluster. The algorithm terminates when there is no reassignment of any pattern from one cluster to another. The main steps of K-means algorithm are as follows:

1. Initialize K cluster centers.

2. Assign each pattern to its closest center.

3. Compute new cluster centers using Eq. (2.5).

4. Repeat steps 2 and 3 until there is no change for each cluster.

2.3 Proposed ELM K-means algorithm

This section describes the proposed approach which combines both the ELM and K-means algorithms. ELM performs a nonlinear transformation of the input data into a high dimensional feature space. This transformation increases the separability of the input data in the high dimensional feature space. The incorporation of ELM enables K-means algorithm to explore the inherent data structure in the new space. In ELM, the hidden layer maps the data from the input space R^d to the high dimensional feature space R^L $(L > d)$ where the data clustering is performed. This idea of ELM mapping is similar to the idea behind the use of kernels, i.e., linearly non-separable features in the input space often become linearly separable after they are mapped to a high dimensional feature space (see Figure 2.1).

Given two data points $\mathbf{x}$ and $\mathbf{z}$, and an ELM with L neurons defining a mapping ϕ from the input space R^d to the feature space F

$$\phi \colon R^d \rightarrow F.$$

The Euclidean distance between $\mathbf{x}$ and $\mathbf{z}$ in the input space is

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\| \mathbf{x} - \mathbf{z} \|^2} \tag{2.8}$$

After the points $\mathbf{x}$ and $\mathbf{z}$ are mapped into the feature space, the Euclidean distance between $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ in the feature space becomes [84]:

$$\begin{aligned} d_F(\mathbf{x}, \mathbf{z}) &= \sqrt{\| \phi(\mathbf{x}) - \phi(\mathbf{z}) \|^2} \\ &= \sqrt{\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) - 2\phi(\mathbf{x}) \cdot \phi(\mathbf{z}) + \phi(\mathbf{z}) \cdot \phi(\mathbf{z})} \end{aligned} \tag{2.9}$$

Eq. (2.8) can be replaced by Eq. (2.9) as the similarity measure in the clustering algorithms working in a high dimensional feature space. Based on this principle, the proposed ELM K-means algorithm can be visualized as mapping the data into a high dimensional feature space and then performing clustering in this feature space.

The ELM K-means algorithm consists of three steps and can be summarized as in Algorithm 3

Algorithm 3: ELM K-means Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons
$g(x)$: activation function

1 Initialize hidden layer parameters $(\mathbf{w}_i, b_i)$, $i = 1, \ldots, L$ randomly;
2 Compute the hidden layer output matrix $\mathbf{H}$;
3 Apply K-means algorithm on the matrix $\mathbf{H}$;

2.4 Experimental study

In this chapter, twelve benchmark data sets are used for evaluating the performance of the proposed ELM K-means algorithm. These data sets, except USPST data set, can be downloaded from the UCI machine learning repository[1]. The data sets and their characteristics, viz. the number of patterns, the number of features and the number of classes are given in alphabetical order in Table 2.1.

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Table 2.1: Data sets characteristics

Data sets	Patterns	Features	Classes
Balance	625	4	3
Cancer-Diagnostic	569	30	2
Cancer-Original	683	9	2
Cardiotocography-3	2126	21	3
Cardiotocography-10	2126	21	10
CNAE	1080	856	9
Dermatology	358	34	6
Glass	214	9	6
Iris	150	4	3
LIBRAS	360	90	15
Spam	1534	57	2
USPST	2007	256	10

2.4.1 Data sets

The data sets considered in this chapter can be briefly described as follows. Balance data set was created for modelling psychological experimental results. Each pattern is classified as having the balance scale tilt to the left, tilt to the right, or remain exactly in the middle. This data set consists of 4 features, 3 classes and there are 625 patterns. Cancer-Diagnostic and Cancer-Original data sets are based on the "breast cancer Wisconsin - Diagnostic" and "breast cancer Wisconsin - Original" data sets, respectively. Both data sets classify a tumor as either benign or malignant. Cancer-Diagnostic data set contains 569 patterns, 30 features. Cancer-Original data sat contains 699 patterns. After removing the 16 database patterns with missing values, the database consists of 683 patterns, 9 features. Cardiotocography data set consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms diagnosed by expert obstetricians. Classification was both with respect to a morphologic pattern (10 classes; 1 to 10) and to a fetal state (three classes; normal, suspect and pathologic). Therefore, this data set can be used either for 10-class (Cardiotocography-10) or 3-class (Cardiotocography-3) experiments. The data set consists of 2126 21-dimensional

patterns. CNAE data set contains 1080 documents of free text business descriptions of Brazilian companies categorized into a subset of 9 categories. The original texts were pre-processed so that each document can be represented as a vector, where the weight of each word represents its frequency in the document. This data set is highly sparse (99.22% of the matrix consists of zeros). Dermatology data set aims to determine the type of Eryhemato-Squamous Disease. The data set contains 366 patterns. After the removal of the 8 data set patterns with missing values, the data set consists of 358 34-dimensional patterns belonging to six different classes. Glass data set contains 214 patterns, 9 features and 6 glass types. The glass types are float processed building windows, non-float processed building windows, vehicle windows, containers, tableware and head lamps. The Fisher Iris data [92] is one of the most popular data sets to test the performance of novel methods in pattern recognition and machine learning. There are three classes in this data set (Setosa, Versicolor and Virginica), each having 50 patterns with four features (sepal length, sepal width, petal length and petal width). One of the classes (viz. Setosa) is linearly separable from the other two, while the remaining two are not linearly separable (see Figure 2.2, in which only two features are used). LIBRAS data set is based on the Libras Movement data set. The data set contains 15 classes of 24 patterns each. Each class references to a hand movement type in LIBRAS (Portuguese name 'LÍngua BRAsileira de Sinais', oficial brazilian sign language). The Spam data set consists of 1534 patterns from two different classes, spam and not-spam. Each pattern is represented by a 57-dimensional feature vector. The USPST data set is a subset of the handwritten digit recognition data set USPS. USPST data set consists of 256 features, 10 classes and there are 2007 patterns.

2.4.2 Results and discussion

For each data set, we report the average of correctly clustered patterns (ACC) which is defined as

$$ACC = \frac{\sum_{i=1}^{\#\text{ of runs}} \#\text{ of correctly clustered patterns}}{\#\text{ of runs}} \tag{2.10}$$

We also report the percentage of average correct clustering (PACC) which is the average of correctly clustered patterns (ACC) percentaged to the size of the data set.

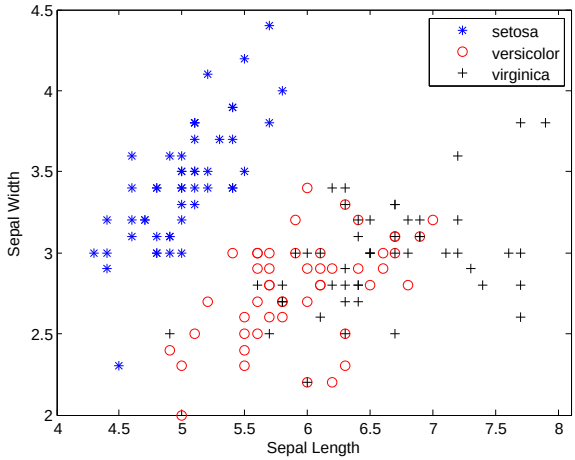

Figure 2.2: Iris data set. There are three classes, Setosa class is linearly separable from the other two classes. Versicolor and Virginica classes are not linearly separable.

Table 2.2: Average performance, in terms of correctly clustered patterns, of the K-means and the ELM K-means algorithms

Data sets	K-means	ELM K-means
Balance	419.3 ± 24 (67.09%)	434.2 ± 16.84 (69.47%)
Cancer-Diagnostic	528 ± 0 (92.79%)	536 ± 0 (94.2%)
Cancer-Original	656 ± 0 (96.05%)	659 ± 0 (96.49%)
Cardiotocography-3	1655 ± 0 (77.85%)	1666.9 ± 16.64 (78.41%)
Cardiotocography-10	984.35 ± 35.73 (46.3%)	1022.35 ± 45.88 (48.09%)
CNAE	527.6 ± 55.13 (48.85%)	602.25 ± 56.95 (55.76%)
Dermatology	296.2 ± 20.54 (82.74%)	304.15 ± 22.78 (84.96%)
Glass	122.5 ± 3.8 (57.24%)	126.15 ± 8.7 (58.95%)
Iris	128.9 ± 12.46 (85.93%)	129.9 ± 22.51 (86.6%)
LIBRAS	168.1 ± 7.5 (46.69%)	176.3 ± 6.89 (48.97%)
Spam	1097.85 ± 155.77 (71.57%)	1101.65 ± 158.08 (71.82%)
USPST	1365.45 ± 46.09 (68.03%)	1425.4 ± 76.51 (71.02%)

$$PACC = 100 \times \frac{ACC}{\text{size of data set}} \tag{2.11}$$

The proposed ELM K-means algorithm is tested on aforementioned 12 benchmark data sets and its performance is compared with K-means algorithm. Sigmoid function is used as an activation function in the ELM hidden layer. For all the data sets, we have used 1000 hidden neurons. The input weights and biases were randomly generated from a uniform distribution over [-1, 1]. We have executed both K-means and ELM K-means algorithms 20 independent times. The average performance, in terms of correctly clustered patterns, of these algorithms are reported in Table 2.2. The PACC values are shown in parenthesis in this table. From the results, it is obvious that the proposed ELM K-means outperformed K-means consistently on all data sets, which demonstrate the advantages of performing clustering in the ELM high dimensional feature space. The results are also in line with the concept that the data which are not linearly separable in the input space often become linearly separable in the high dimensional space which enables the clustering algorithm to achieve significantly better performance than that in the input space. As far as the time complexity of the proposed algorithm is concerned, it is definitely more in comparison to traditional K-means clustering algorithm due to higher number of dimensions involved. Moreover, there is an additional cost involving one-time computation of the hidden layer output matrix **H** of the ELM algorithm.

2.5 Conclusion

In this chapter, extreme learning machine, which is a new and simple technique, is used with K-means algorithm (ELM K-means) for unsupervised clustering of twelve benchmark data sets. The results of the experiments demonstrate that the integration of ELM method with K-means algorithm improves the quality of clustering.

Chapter 3

Artificial bee colony algorithm for clustering: an extreme learning approach

3.1 Introduction

The classical K-means algorithm is probably the most popular technique of partitional clustering [83]. Due to its simplicity and efficiency, K-means algorithm has been widely used in the literature. However, K-means algorithm has the shortcomings of dependence on the initial cluster centers and convergence to local minima. In addition, K-means algorithm will fail to obtain meaningful clusters if the separation boundaries between clusters are nonlinear. Several heuristic clustering algorithms have been introduced in the literature in order to overcome the shortcomings of K-means algorithm. These algorithms perform clustering either in the input space or in the kernel feature space. Krishna and Murty [93] proposed a hybrid approach called genetic K-means algorithm for clustering. They have designed new operators, such as distance-based mutation operator, in order to achieve global search and fast convergence. Selim and Al-Sultan [94] proposed a simulated annealing approach for solving the clustering problem. A particle swarm optimization approach for data clustering is proposed by Merwe and Engelhrecht [95]. Shelokar et al. [96] proposed an ant colony optimization algorithm for data clustering. Zhang and Cao [84] proposed a novel ant-based clustering algorithm integrated with the kernel method. Karaboga and Ozturk [97] and Zhang et al. [98] used

the ABC algorithm to solve the clustering problem. Yan et al. [99] proposed a hybrid artificial bee colony algorithm for data clustering. In these algorithm, the crossover operator of genetic algorithm (GA) is introduced to enhance the information exchange among bees. In the previous chapter, K-means algorithm is integrated with ELM to perform clustering in ELM feature space. This combination (ELM K-means) has obtained better clustering performance compared to classical K-means algorithm. This is due to the possibility of linear separability of data patterns in the ELM feature space. However, similar to classical K-means, ELM K-means algorithm has shortcomings of dependence on the initial cluster centers and convergence to local minima.

In this chapter, to take advantage of the linear separability of data in the high dimensional ELM feature space and to overcome the shortcomings of K-means algorithm, we have incorporated ELM method into an artificial bee colony (ABC) algorithm based clustering approach and proposed a novel ABC clustering algorithm with the ELM feature space (ELM-ABC). The ELM method is used to project the data into a high dimensional feature space and ABC algorithm performs clustering within this feature space. Clearly, clustering is a grouping problem, i.e., a problem that seeks a partitioning of a given set of objects (instances in case of clustering) into various groups (clusters) subject to some constraints so that a given cost function is optimized [100]. Recently, ABC algorithm has been applied to solve several grouping problems where it obtained better results in comparison to existing state-of-the-art metaheuristic approaches [101, 102, 103, 104]. The success of ABC algorithm in solving these grouping problems has motivated us to develop the proposed ABC approach for clustering.

The remaining part of this chapter is organized as follows: Section 3.2 presents ABC-based clustering algorithm and ELM-ABC algorithm. Experiments and their results are presented and discussed in Section 3.3. Finally, Section 3.4 concludes this chapter.

3.2 Artificial bee colony algorithm-based clustering

This section presents our ELM-ABC algorithm for clustering after describing the salient features of ABC algorithm for clustering.

3.2.1 Initial population

ABC algorithm begins by creating a randomly distributed initial population of SN solutions (food sources), where SN denotes the number of employed bees or onlooker bees. Each solution (food source) $\mathbf{S}_i(i = 1, 2, ..., SN)$ in our case is a N-dimensional vector. Here, N denotes the number of instances in the problem. An initial solution is constructed by assigning each instance to one of the K clusters randomly. Let $\mathbf{S}_i$ represent the ith solution in the population, then each element of the solution is generated as follows:

$$\mathbf{S}_{i,j} = rand(K), \tag{3.1}$$

where $i = 1, 2, \ldots, SN$ and $j = 1, 2, \ldots, N$ and $rand$ is a function that returns an integer uniformly at random in the interval [1, K]. Figure 3.1 shows the solution representation for problem with nine instances and two clusters where the instances 1, 3, 6 and 9 are assigned to cluster one and the instances 2, 4, 5, 7 and 8 belong to cluster two.

1	2	1	2	2	1	2	2	1

Figure 3.1: Solution representation

3.2.2 Neighboring solution generation

The procedure given in Algorithm 4 is used to obtain a new solution $\mathbf{V}$ in the neighborhood of the current solution $\mathbf{S}$. In this procedure, the tunable parameter cp is used to control how much percentage of the current solution can be copied to the new solution.

3.2.3 ABC algorithm for clustering

The algorithm starts by generating randomly distributed initial solutions using Eq. (3.1) and then evaluating their fitness. In partitional clustering problem, the goal is to find a partition of the given data that minimizes (or maximizes) some criterion function. The sum of squared error function is one of the most widely used criteria. The objective function used in ABC-based clustering algorithm as defined in Eq. (2.7) computes the sum of squared distances between all data patterns and their associated

Algorithm 4: The neighboring solution generation procedure

input : Let $\mathbf{S}_i$ be current solution and $\mathbf{V}_i$ is its neighbor

for $j = 1$ *to* N **do**
  if $random \leq cp$ **then**
    Copy $\mathbf{S}_{i,j}$ to $\mathbf{V}_{i,j}$;
  else
    Set $\mathbf{V}_{i,j} = -1(unassigned)$;
  end
end
Calculate cluster centers from assigned instances ($\mathbf{V}_{i,j} \neq -1$);
for *each unassigned instance* $\boldsymbol{V}_{i,j} == -1$ **do**
  Reassign to the closest cluster center using Euclidean distance;
end
if $\boldsymbol{V}_i$ *is infeasible* **then**
  Set $\mathbf{V}_i = \mathbf{S}_i$;
end

cluster center, which reflects that data patterns within the individual cluster must be similar. The fitness of the ith solution in the population is calculated as follows:

$$fit_i = \frac{1}{1 + J_i(C)} \tag{3.2}$$

where J_i is the objective function value of the ith solution in (2.7). Instead of using the objective function directly as a measure of fitness where the lower objective function value corresponds to a more fit solution, we have used this fitness function to follow the convention that the higher value of fitness function corresponds to a more fit solution. The term $1 + J_i(C)$ in the denominator of the fitness function facilitates the use of this function even when $J_i(C)$ is zero. This fitness function is same as the one used in [97].

After associating each employed bee with an initial solution, every employed bee produces a new solution by using the steps in Algorithm 4 and then evaluates its fitness. If the new solution has better fitness value than the old one, the old one is replaced by the new one. The employee bees tend to share the fitness information of their solutions with the onlooker bees. Based on this information, each onlooker bee selects an employed bee solution with a probability related to its fitness value. The probability of selecting a solution i is defined as follows:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{3.3}$$

As in the case of an employed bee, each onlooker bee produces a new solution in the neighborhood of its selected solution, evaluates its fitness and applies a greedy selection between the new solution and the selected solution. The new solution replaces the selected solution in case it is better. The solution of which the fitness value is not improved after performing a predetermined number of trials *limit* is abandoned, and the associated employed bee becomes a scout. The scout produces a new solution randomly using Eq. (3.1). The process of abandoning and replacing an exhausted food source (solution) enables the algorithm to avoid suboptimal solutions. The search processes of the employed, onlooker and scout bees are repeated until the termination condition is met. The pseudo-code of the ABC algorithm for clustering is given in Algorithm 5.

3.2.4 Proposed ELM-ABC algorithm for clustering

The ABC algorithm presented in Algorithm 5 is a distance-based clustering algorithm. As already mentioned in Section 2.1, the distance-based clustering algorithm may get stuck in suboptimal solutions if the separation boundaries between clusters are nonlinear [85]. In ABC algorithm, the process of abandoning and replacing non-improving solution is one way to avoid suboptimal solutions. The other way is to project the input data into a high dimensional space and then perform clustering in that space. The ELM hidden layer is used to nonlinearly transform the input data into a high dimensional ELM feature space. This transformation often increases the separability of the input data in the ELM feature space. The combined algorithm will be referred to as ELM-ABC algorithm subsequently. The pseudo-code of the ELM-ABC algorithm is given in Algorithm 6.

3.3 Experimental study

In this chapter, twelve benchmark data sets have been used to evaluate the performance of the proposed ELM-ABC algorithm. The specifications of the data sets are given in Table 2.1. The data sets are described briefly in Section 2.4.1.

Algorithm 5: The pseudo-code of the ABC algorithm for clustering

Initialize the population of SN solutions $\mathbf{S}_{ij}, i = 1, 2, \ldots, SN, j = 1, 2, \ldots, N$. Assign employed bees to the solutions;
Evaluate the fitness $fit(\mathbf{S}_i)$ of the population, $i = 1, 2, \ldots, SN$;
Set $trial_i = 0$, $i = 1, 2, \ldots, SN$;
repeat
    /* Employed bees phase */
    for $i = 1$ *to* SN **do**
        Produce a new solution $\mathbf{V}_i$ from $\mathbf{S}_i$ using Algorithm 4;
        Evaluate the fitness of the new solution using Eq. (3.2);
        if $fit(\boldsymbol{V}_i) > fit(\boldsymbol{S}_i)$ **then**
            Replace $\mathbf{S}_i$ with $\mathbf{V}_i$ and set $trial_i = 0$;
        else
            $trial_i = trial_i + 1$;
        end
    end
    Calculate the probability values p_i for the solutions using Eq. (3.3);
    /* Onlooker bees phase */
    for *each onlooker bee* **do**
        Select a solution $\mathbf{S}_i$ depending on p_i;
        Produce a new solution $\mathbf{V}_i$ from $\mathbf{S}_i$ using Algorithm 4;
        Evaluate the fitness of the new solution using Eq. (3.2);
        if $fit(\boldsymbol{V}_i) > fit(\boldsymbol{S}_i)$ **then**
            Replace $\mathbf{S}_i$ with $\mathbf{V}_i$ and set $trial_i = 0$;
        else
            $trial_i = trial_i + 1$;
        end
    end
    /* Scout bees phase */
    for *each solution* $\boldsymbol{S}_i$ **do**
        if $trial_i > limit$ **then**
            Replace $\mathbf{S}_i$ with a randomly produced solution using Eq. (3.1);
        end
    end
    Memorize the best solution achieved so far;
until *termination condition is satisfied*;

Algorithm 6: The pseudo-code of the ELM-ABC algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons
$g(x)$: activation function

1 Randomly generate input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, L$;
2 Compute the hidden layer output matrix $\mathbf{H}$;
3 Apply Algorithm 5 on the matrix $\mathbf{H}$;

3.3.1 Results and discussion

The proposed ELM-ABC algorithm is tested on 12 benchmark data sets and its performance is compared with the K-means, ELM K-means and ABC algorithms. While comparison with K-means provides a baseline, comparing with ELM K-means enables assessment of advantages of metaheuristic-based clustering approach. Sigmoid function is used as an activation function in the ELM hidden layer. For all the data sets, we have used 1000 hidden neurons. The input weights and biases were randomly generated from a uniform distribution over [-1, 1]. For ABC and ELM-ABC, the number of employed bees and the number of onlookers are set to be equal to the number of food sources (SN), which is 10, the *limit* = 10 and the value of the *cp* is 0.7. Both ABC and ELM-ABC algorithms terminate if they have executed for 1000 iterations or if the best solution found so far is not improved after performing a predetermined number of iterations *glimit*. We set *glimit* = 200. We have executed all algorithms 20 times independently. All the simulations are carried out in MATLAB environment running in Core i5-2400, 3.10 GHZ CPU with 4 GB RAM.

The average performance, in terms of correctly clustered patterns, of these algorithms are reported in Table 3.1. From the results, we can observe that ELM-ABC algorithm has obtained satisfying results on all the data sets. It yielded the best clustering performance among the four algorithms on ten out of twelve data sets. ELM-ABC algorithm, on some data sets, has obtained similar results as ELM K-means algorithm which may be due to the use of the same ELM feature space. However, on other data sets, such as Dermatology, Iris, Spam and USPST, ELM-ABC obtains significantly better results than ELM K-means. This may be due to the fact that ELM-ABC algorithm provides a way of avoiding suboptimal solutions through the use of *scout mechanism*. For CNAE data set, the standard deviation is high for all algorithms. This can be

Table 3.1: Average performance, in terms of correctly clustered patterns, of the K-means, ELM K-means, ABC and ELM-ABC algorithms

Data sets	K-means	ELM K-means	ABC	ELM-ABC
Balance	419.3 ± 24 (67.09%)	434.2 ± 16.84 (69.47%)	407.9 ± 7.23 (65.26%)	429.5 ± 1.82 (68.72%)
Cancer-Diagnostic	528 ± 0 (92.79%)	536 ± 0 (94.2%)	528 ± 0 (92.79%)	536 ± 0 (94.2%)
Cancer-Original	656 ± 0 (96.05%)	659 ± 0 (96.49%)	656 ± 0 (96.05%)	659 ± 0 (96.49%)
Cardiotocography-3	1655 ± 0 (77.85%)	1666.9 ± 16.64 (78.41%)	1655 ± 0 (77.85%)	1655 ± 0 (77.85%)
Cardiotocography-10	984.35 ± 35.73 (46.3%)	1022.35 ± 45.88 (48.09%)	1024 ± 5 (48.17%)	1048.5 ± 2.3 (49.3%)
CNAE	527.6 ± 55.13 (48.85%)	602.25 ± 56.95 (55.76%)	496.35 ± 31.60 (45.96%)	602.45 ± 40.71 (55.78%)
Dermatology	296.2 ± 20.54 (82.74%)	304.15 ± 22.78 (84.96%)	310 ± 0 (86.59%)	315.2 ± 10.67 (88.04%)
Glass	122.5 ± 3.8 (57.24%)	126.15 ± 8.7 (58.95%)	126 ± 0 (58.89%)	127 ± 0 (59.35%)
Iris	128.9 ± 12.46 (85.93%)	129.9 ± 22.51 (86.6%)	134 ± 0 (89.33%)	146 ± 0 (97.33%)
LIBRAS	168.1 ± 7.5 (46.69%)	176.3 ± 6.89 (48.97%)	164.35 ± 1.14 (45.65%)	179.55 ± 2.26 (49.88%)
Spam	1097.85 ± 155.77 (71.57%)	1101.65 ± 158.08 (71.82%)	930 ± 0 (60.62%)	1241 ± 0 (80.9%)
USPST	1365.45 ± 46.09 (68.03%)	1425.4 ± 76.51 (71.02%)	1389.4 ± 8 (69.23%)	1486.3 ± 5.8 (74.06%)

attributed to the presence of outliers as the data set is highly sparse. The results obtained by ELM K-means and ELM-ABC algorithms demonstrate the advantages of performing clustering in the ELM high dimensional feature space. These results are also in line with the concept that the data which are not linearly separable in the input space often become separable in high dimensional space which enables the clustering algorithms to achieve significantly better performance than that in the input space. Figures 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 ,3.12, 3.13 show the clustering performance (PACC) of ELM K-means and ELM-ABC with different number of hidden neurons $[50, 100, \ldots, 1000]$. From these results, we can observe the following:

- In most cases, the highest clustering performance (PACC) obtained by ELM-ABC is generally higher than that of ELM K-means.
- The ELM-ABC performance dominated ELM K-means for different choices of number of ELM hidden neurons.

- The performance of ELM-ABC and ELM K-means reaches steady-state after the number of ELM neurons is 500 and above for most data sets.

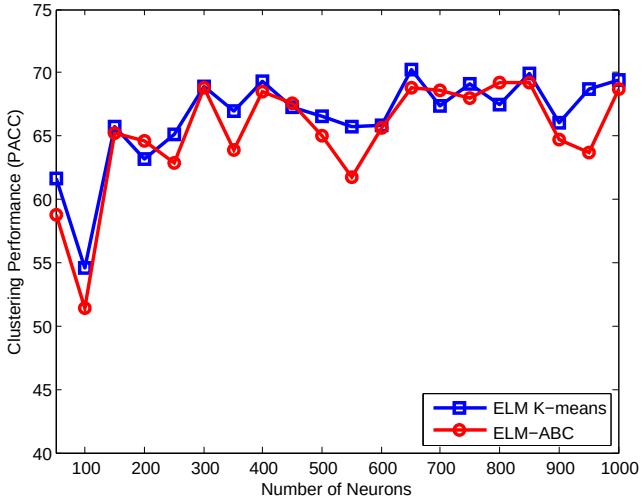

Figure 3.2: Clustering performance on Balance with respect to different number of neurons

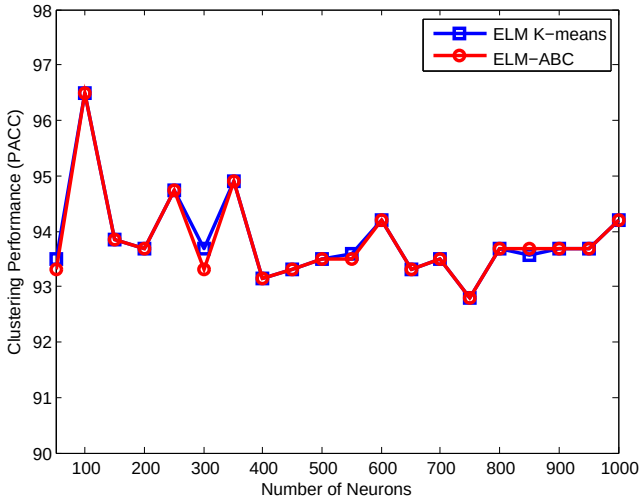

Figure 3.3: Clustering performance on Cancer-Diagnostic with respect to different number of neurons

To see how the time increases as more patterns are considered, we have conducted an experiment on USPST data set with different number of patterns. Figure 3.14 shows

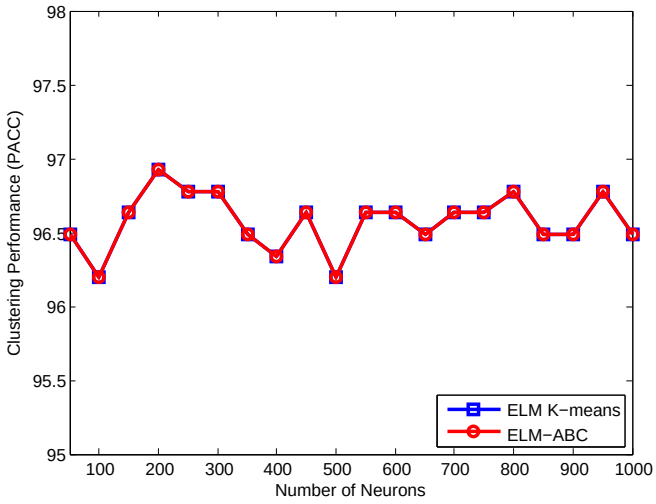

Figure 3.4: Clustering performance on Cancer-Original with respect to different number of neurons

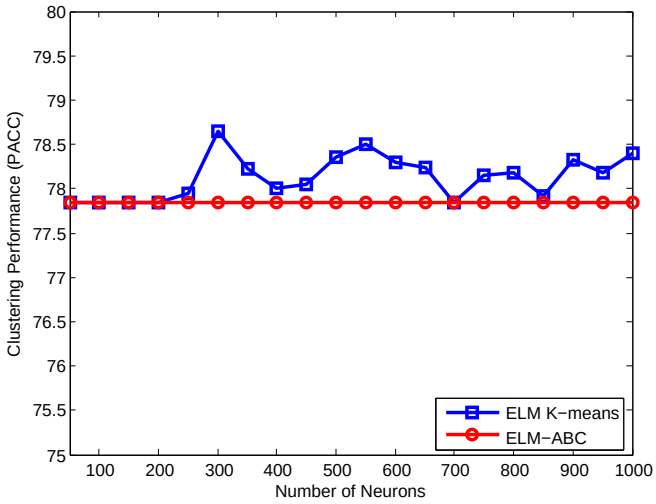

Figure 3.5: Clustering performance on Cardiotocography-3 with respect to different number of neurons

the time spent by ELM-ABC algorithm on USPST data set with different number of patterns $[100, 200, \ldots, 1000, 1200, \ldots, 1800, 2007]$. The number of hidden neurons in ELM-ABC is fixed to 1000. The graph indicates approximately linear increase in time with respect to number of patterns.

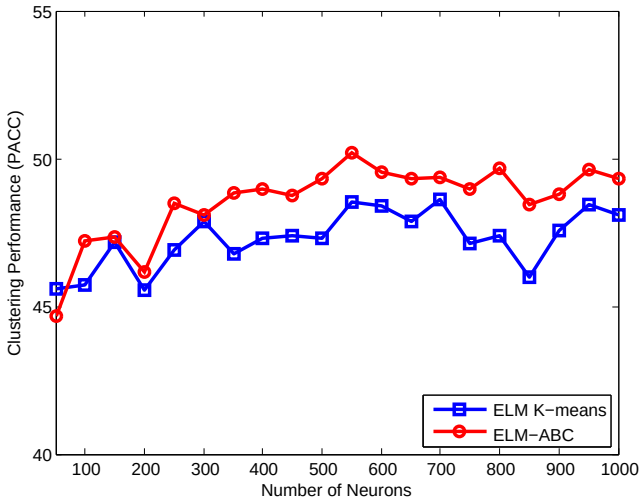

Figure 3.6: Clustering performance on Cardiotocography-10 with respect to different number of neurons

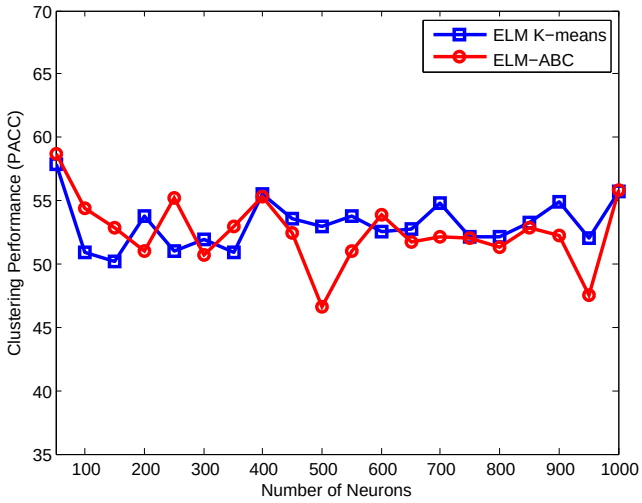

Figure 3.7: Clustering performance on CNAE with respect to different number of neurons

3.3.2 Computational efficiency

In this chapter, the idea is to perform clustering in ELM feature space. In projecting the data into a high dimensional ELM feature space, there is a time cost, and, in this section, we evaluate the time-accuracy tradeoff. ELM K-means and ELM-ABC are the variations of K-means and ABC algorithms, respectively. Direct comparison of

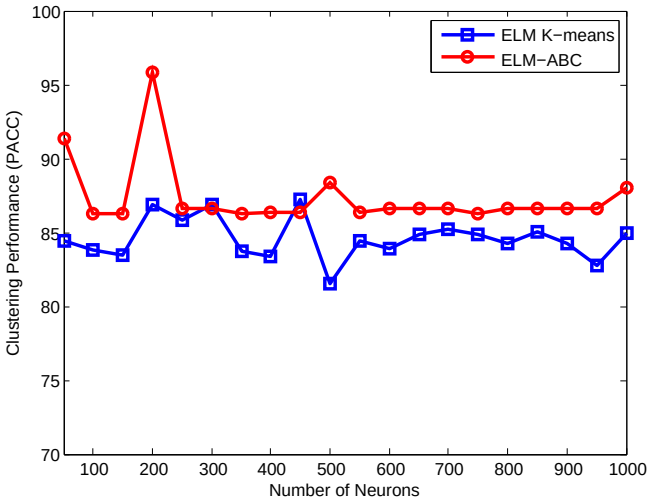

Figure 3.8: Clustering performance on Dermatology with respect to different number of neurons

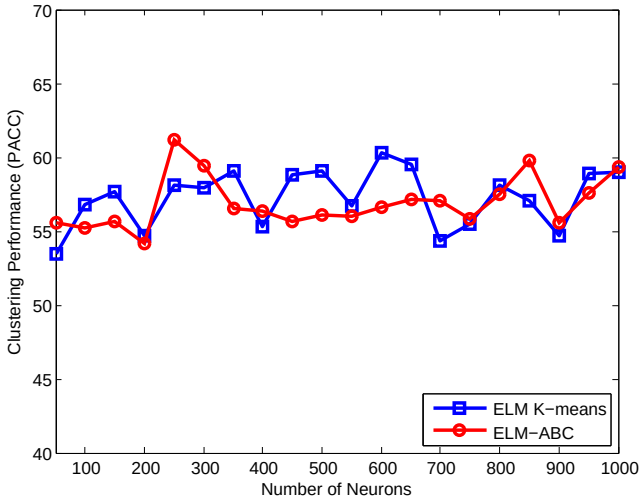

Figure 3.9: Clustering performance on Glass with respect to different number of neurons

absolute running times between these two algorithms would be rather unfair because metaheuristic techniques are known to be compute intensive. In addition, the proposed ELM-ABC approach is not for real time clustering tasks, but for offline clustering. Therefore, instead of making a direct absolute time comparison, we will compare the time-accuracy tradeoff of both algorithms. Table 3.2 shows time-accuracy tradeoff

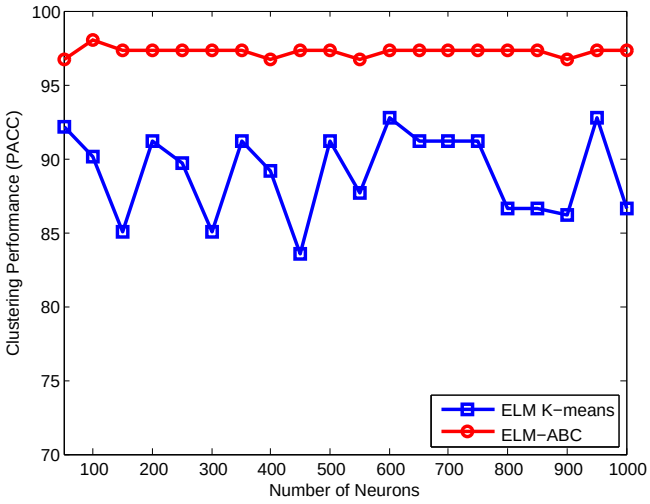

Figure 3.10: Clustering performance on Iris with respect to different number of neurons

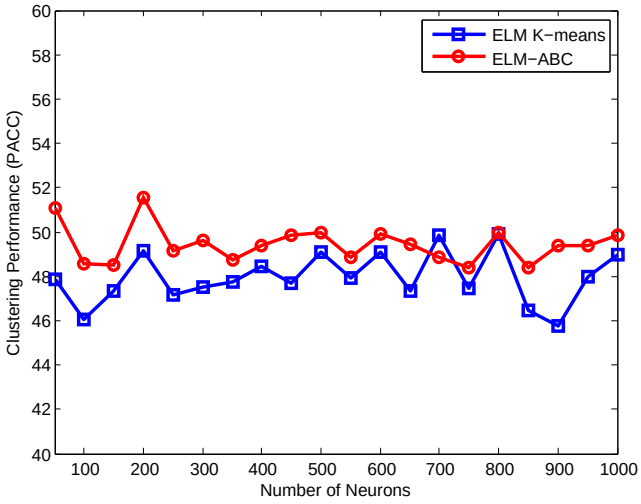

Figure 3.11: Clustering performance on LIBRAS with respect to different number of neurons

of ELM-ABC and ELM K-means algorithms, where NPIA is normalized percentage improvement in accuracy which is defined as:

$$NPIA = 100 \times \frac{\text{PACC in ELM space} - \text{PACC in input space}}{\text{PACC in input space}} \tag{3.4}$$

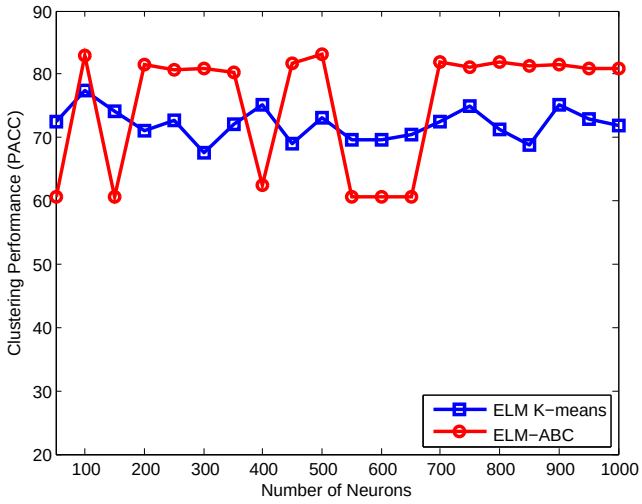

Figure 3.12: Clustering performance on Spam with respect to different number of neurons

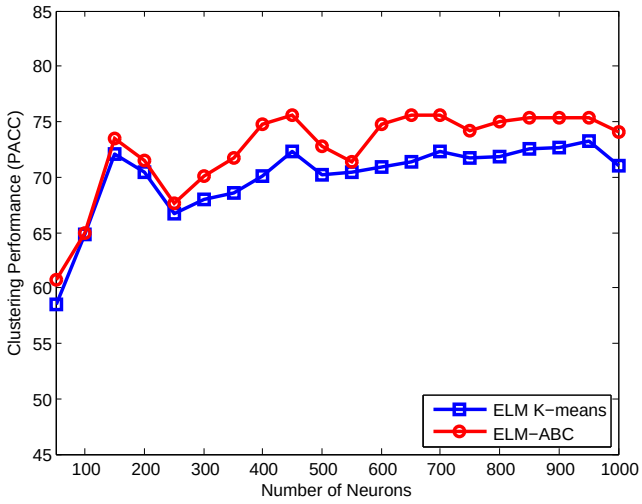

Figure 3.13: Clustering performance on USPST with respect to different number of neurons

and NCT is normalized change in time which is defined as:

$$NCT = \frac{\text{Time spent in ELM space} - \text{Time spent in input space}}{\text{Time spent in input space}} \tag{3.5}$$

The NPIA and NCT of ELM-ABC are with respect to ABC algorithm and those for ELM K-means are with respect to K-means algorithm. From the Table 3.2, we can

Table 3.2: Time-accuracy tradeoff of ELM-ABC and ELM K-means

	ELM-ABC		**ELM K-means**	
Data sets	NPIA	NCT	NPIA	NCT
Balance	5.3	24.61	3.55	62.81
Cancer-Diagnostic	1.52	17.98	1.52	14.54
Cancer-Original	0.46	20.35	0.46	11.29
Cardiotocography-3	0	19.62	0.72	40.26
Cardiotocography-10	2.39	30.1	3.87	114.58
CNAE	21.37	0.11	14.15	2.2
Dermatology	1.67	19.17	2.68	17.45
Glass	0.78	14.3	2.99	11.48
Iris	8.96	9.57	0.78	3.33
LIBRAS	9.27	11.25	4.88	15.31
Spam	33.45	11.79	0.35	4.6
USPST	6.98	5.28	4.4	3.41
AVERAGE	7.68	15.34	3.36	25.11

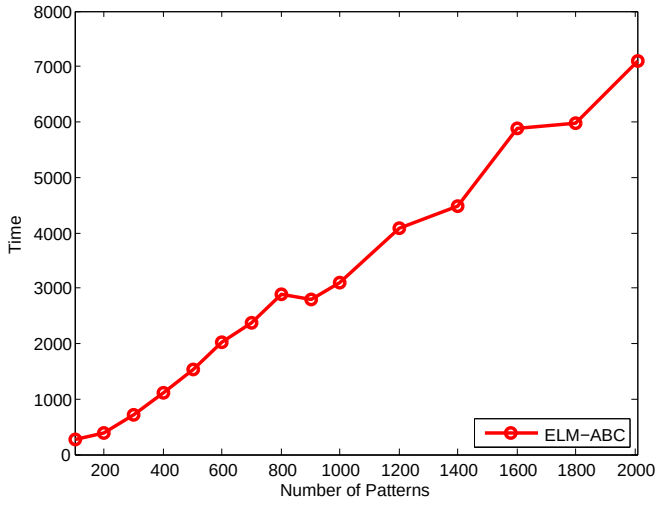

Figure 3.14: Execution time of ELM-ABC on USPST with respect to different number of patterns

Table 3.3: P-values from two-samples t-tests of ELM-ABC against the other techniques

Data sets	K-means	ELM K-means	ABC
Balance	0.0657	0.2222	< 0.0001
Cancer-Diagnostic	N/A	N/A	N/A
Cancer-Original	N/A	N/A	N/A
Cardiotocography-3	N/A	0.0028	N/A
Cardiotocography-10	< 0.0001	0.0151	< 0.0001
CNAE	< 0.0001	0.9899	< 0.0001
Dermatology	0.0007	0.0568	0.0356
Glass	< 0.0001	0.6646	N/A
Iris	< 0.0001	0.0028	N/A
LIBRAS	< 0.0001	0.0522	< 0.0001
Spam	0.0002	0.0003	N/A
USPST	< 0.0001	0.001	< 0.0001

see that ELM combined with ABC improves the clustering accuracy on average by 7.68%, but with average NCT of 15 units, whereas in ELM K-means, the improvement in accuracy on average is 3.36% at the average NCT of 25 units. The NCT of ELM-ABC on average is about two third of that of ELM K-means, while the clustering accuracy improvement is double. So ELM-ABC has a better time-accuracy tradeoff in comparison to ELM K-means.

To further verify the effectiveness of the proposed ELM-ABC, the two-tailed t-test has been conducted at 5% significance level to compare proposed ELM-ABC with three other algorithms. The results of the t-tests are shown in Table 3.3. The N/A in Table 3.3 stands for Not Applicable, covering those cases for which both compared algorithms have zero standard deviation. The t-test results show that the difference between ELM-ABC and the other algorithms is statistically significant in most cases. If we look in Table 3.1 for results corresponding to 11 N/A cases in Table 3.3, we can observe that there are 7 cases where ELM-ABC approach obtained better results. For example, on Cancer-Diagnostic data set, we can not perform t-test between ABC and ELM-ABC as both the approaches have standard deviation of zero. However, on this data set, ELM-ABC correctly classifies 536 instances in each of the 20 runs, whereas

ABC correctly classifies only 528 instances in each of the 20 runs. As in this case and in other 6 cases, results are obtained with standard deviation of zero, therefore it is extremely improbable that randomness has any role in the better performance of ELM-ABC over the other method in consideration, and hence, the result of ELM-ABC in these 7 cases can also be considered significant.

3.4 Conclusion

In the previous chapter, encouraging results of clustering performance of K-means algorithm have been obtained using the ELM high dimensional feature space. However, ELM K-means algorithm is also limited by its dependence on the choice of initial cluster center locations and convergence to (suboptimal) local minima, the problems that usually plague conventional algorithms such as K-means. To overcome these limitations, in this chapter we combine the ELM approach with ABC algorithm for unsupervised clustering. We have evaluated ELM K-means and ELM-ABC on wide range of real world benchmark data sets. Experimental results show that ELM-ABC gives significantly better time-accuracy tradeoff compared to ELM K-means algorithm. These results also demonstrate that the integration of ELM method with ABC algorithm improves the quality of clustering performed by the ABC algorithm itself.

.

Chapter 4

Combining ELM with Random projections for low and high dimensional data classification and clustering

4.1 Introduction

Dimensionality reduction is the process of obtaining a low dimensional representation of a given high dimensional data. Generally, there are two types of dimension reduction techniques: feature selection and feature extraction. Feature selection techniques reduce the dimensions of data by finding a subset of relevant features from original dimensions. Feature extraction techniques reduce the dimensionality by constructing new dimensions either by using some transformations or by combining two or more original dimensions.

Random projection (RP) is a feature extraction-based dimensionality reduction technique which represents high dimensional data in a low dimensional space while approximately preserving the distances between the data points. The ability of random projection to approximately preserve the distances among the data points in the transformed space is an important property for machine learning applications in general. Fradkin and Madigan [105] have compared the performance of different machine learning methods such as nearest neighbor methods, decision trees and SVM in con-

junction with the dimensionality reduction techniques RP and principle component analysis (PCA). In their work, PCA outperformed RP, but the computational overhead associated with PCA is high compared to RP. Deegalla and Bostrom [106] have first used RP and PCA for dimensionality reduction of data and then applied nearest neighbor classifier on the reduced data. Their experiments show that the performance of PCA depends heavily on the value we choose for the number of reduced dimensions. After reaching a peak, the accuracy for PCA decreases with the increase in number of dimensions. On the other hand, the accuracy for RP increases with the increase in number of dimensions. Fern and Brodley [107] investigated the application of RP integrated with ensemble methods for unsupervised learning of high dimensional data. Cardoso and Wichert [108] proposed an iterative method for clustering high dimensional data using RP and K-means algorithm. Recently, Gastaldo et al. [109] have proposed a new method which combines ELM with random projections. The performance of their method has been tested on two binary classification problems. The experiments demonstrated the ability of the proposed model to balance classification accuracy and computational complexity.

In this chapter, we provide a systematic study on the application of RP in conjunction with ELM for both high and low dimensional data classification and clustering tasks. This study is based on the following intuition: the very fact that RP technique can do dimensionality reduction without distorting significantly the structure of data, can be used for different scenarios. First, for high dimensional data, RP can be used to reduce the dimensionality by shrinking the number of neurons in the ELM hidden layer, as the solution may lie in a lower dimensional subspace, and then we can use regression and K-means for classification and clustering, respectively. Second, for low dimensional data, we use ELM to project the data into ELM feature space, where we expect the data to be linearly separable, and then we use RP to bring the data down to a lower space while preserving the linear separability of data. The second scenario tackles the problem of nonlinear separability of data in the input space and reduces the computational cost of classification and clustering in the ELM feature space by projecting the data into a lower space while preserving the linear separability of the data.

The remainder of this chapter is organized as follows: Section 4.2 introduces random projection (RP). In Section 4.3, the first scenario which combines ELM with RP for high

dimensional data classification and clustering is introduced. Section 4.4 presents the second scenario which combines ELM with RP for low dimensional data classification and clustering. Experimental results and their analysis are presented in Section 4.5. Finally, Section 4.6 concludes this chapter.

4.2 Random projection

Random projection, as a simple and powerful technique for dimensionality reduction, is used to project high dimensional data into low dimensional subspace while approximately preserving the distances between the points.

Suppose we are given a data set $\mathbf{X} = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in R^d, i = 1, \ldots, N\}$, and a random $d \times r$ matrix $\mathbf{R}$, then the original d-dimensional data $\mathbf{X}$ is projected into r-dimensional ($r \ll d$) subspace using a matrix $\mathbf{R}$ as follows:

$$\mathbf{X}^{rp} = \mathbf{XR} \tag{4.1}$$

where $\mathbf{X}^{rp}$ is the projection of the data into a lower r-dimensional subspace. The idea of random projection originates from the Johnson-Lindenstrauss (JL) lemma [110]. The JL lemma states that if we are given a set of N points in a d-dimensional space, then we can project these points down into r-dimensional subspace, with $r \geq O(\ln(N)/\epsilon^2)$, so that all pairwise distances are preserved up to a $1 \pm \epsilon$ factor, where $\epsilon \in (0, 1)$. Several algorithms have been proposed to construct a random matrix that satisfies the JL lemma [111]. In this chapter, we use a matrix whose entries are independent realizations of ± 1 Bernoulli random variables [112]. Hence, The elements r_{ij} of the matrix $\mathbf{R}$ are assigned as follows:

$$r_{ij} = \begin{cases} +1/\sqrt{r} & \text{with probability } 0.5 \\ -1/\sqrt{r} & \text{with probability } 0.5 \end{cases} \tag{4.2}$$

In general, the elements of a matrix that maps the data from high dimensional space to a low dimensional subspace and satisfies the JL lemma can be assigned as follows:

$$\begin{cases} +1/\sqrt{\text{reduced dimension}} & \text{with probability } 0.5 \\ -1/\sqrt{\text{reduced dimension}} & \text{with probability } 0.5 \end{cases} \tag{4.3}$$

4.3 High dimensional data classification and clustering problems

4.3.1 Classification of high dimensional data

In ELM network, the hidden layer maps the data from the input space R^d to the ELM feature space R^L. This ELM mapping performs either dimensionality reduction if $L < d$ or dimensionality expansion if $L > d$. The ELM algorithm (without combination with random projection) for the classification of high dimensional data is shown in Algorithm 1, in which $L < d$. The integration of RP with ELM (ELM^{rp}) [109] for high dimensional data classification is shown in Algorithm 7. In ELM^{rp} algorithm, the input weights and biases of the ELM hidden layer are initialized using Eq. (4.3) so that the mapping satisfies the JL lemma.

Algorithm 7: ELM^{rp} Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons ($L < d$)
$g(x)$: activation function

1 Initialize hidden layer parameters ($\mathbf{w}_i$, b_i), $i = 1, \ldots, L$ using Eq. (4.3);
2 Compute the hidden layer output matrix $\mathbf{H}$;
3 Compute the output weight $\boldsymbol{\beta}$;

4.3.2 Clustering of high dimensional data

Classical distance-based clustering algorithms, such as K-means algorithm, for low dimensional spaces cannot produce clusters that are meaningful in high dimensional data because several features are irrelevant and clusters are usually hidden in different low dimensional subspaces [113][114]. To enable conventional clustering algorithms to cluster high dimensional data efficiently and effectively, the data is first projected into a low dimensional space and then clustering is applied on the projected points. The ELM hidden layer perform this projection by setting the number of hidden neurons less than the number of input dimensions. In Algorithm 3, ELM is incorporated into K-means algorithm (ELM K-means) for data clustering. For clustering high dimensional data, we set $L < d$ in Algorithm 3. In Algorithm 8, ELM combined with RP is first used to reduce the dimensionality of the data while approximately preserving the distances

between the data points and then K-means algorithm performs clustering in the reduced space (ELM$^{\text{rp}}$ K-means).

Algorithm 8: ELM$^{\text{rp}}$ K-means Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons ($L < d$)
$g(x)$: activation function

1 Initialize hidden layer parameters ($\mathbf{w}_i$, b_i), $i = 1, \ldots, L$ using Eq. (4.3);
2 Compute the hidden layer output matrix $\mathbf{H}$;
3 Apply K-means algorithm on the matrix $\mathbf{H}$;

4.4 Low dimensional data classification and clustering problems

4.4.1 Classification of low dimensional data

For low dimensional data classification, ELM hidden layer maps the data from the d-dimensional input space to the L-dimensional ELM feature space, where $L > d$. By mapping the data to high dimensional ELM feature space, the linear separability of the data often increases within the transformed space. The ELM algorithm (without combination with RP) for the classification of low dimensional data is shown in Algorithm 1.

4.4.1.1 ELM combined with RP (ELM-RP).

The ELM projection of the low dimensional data into high dimensional ELM feature space has advantage of increasing the linear separability of the data points in the ELM space, but performing classification or clustering in this space is expensive due to the increase in dimensions. To tackle this problem, we propose to perform dimensionality reduction using RP on ELM feature space and then perform classification or clustering in the reduced space. The proposed approach has two advantages: first the linear separability of the data in ELM feature space is preserved in the reduced space, second the computational complexity of performing classification or clustering is reduced.

The network architecture of the proposed approach consists of two hidden layers instead of one as in the ELM network shown in Figure 1.1. The network architecture

of the proposed approach is similar to the ELM network shown in Figure 1.1 with an extra hidden layer. The first hidden layer contains L neurons, whereas the second hidden layer contains r neurons with $r < L$. The second hidden layer output matrix $\mathbf{H}'$ is calculated as follows: from Eq. (1.4) $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_N)]^T$ where $\mathbf{h}(\mathbf{x}) = [g(\mathbf{w}_1 \cdot \mathbf{x} + b_1), \ldots, g(\mathbf{w}_L \cdot \mathbf{x} + b_L)]$ then

$$\mathbf{H}' = \begin{bmatrix} g(\mathbf{w}'_1 \cdot \mathbf{h}(\mathbf{x}_1) + b'_1) & \ldots & g(\mathbf{w}'_r \cdot \mathbf{h}(\mathbf{x}_1) + b'_r) \\ \vdots & \ldots & \vdots \\ g(\mathbf{w}'_1 \cdot \mathbf{h}(\mathbf{x}_N) + b'_1) & \ldots & g(\mathbf{w}'_r \cdot \mathbf{h}(\mathbf{x}_N) + b'_r) \end{bmatrix}_{N \times r} \tag{4.4}$$

where $\mathbf{w}'_j = [w_{j1}, \ldots, w_{jL}]^T$ is the weight vector connecting the jth neuron in the second hidden layer and the neurons in the first hidden layer, and b'_j is the bias of the jth neuron in the second hidden layer.

The integration of RP with ELM (ELM-RP) for low dimensional data classification is shown in Algorithm 9.

Algorithm 9: ELM-RP Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of neurons in the first hidden layer ($L > d$)
r: number of neurons in the second hidden layer ($r < L$)
$g(x)$: activation function

1 Initialize first hidden layer parameters ($\mathbf{w}_i$, b_i), $i = 1, \ldots, L$ randomly;
2 Compute the first hidden layer output matrix $\mathbf{H}$;
3 Initialize second hidden layer parameters ($\mathbf{w}'_j$, b'_j), $j = 1, \ldots, r$ using Eq. (4.3);
4 Compute the second hidden layer output matrix $\mathbf{H}'$;
5 Compute the output weight $\boldsymbol{\beta}' : \boldsymbol{\beta}' = \mathbf{H}'^{\dagger}\mathbf{T}$;

4.4.2 Clustering of low dimensional data

The data points which are not linearly separable in the input space often become linearly separable after mapping the data to high dimensional ELM feature space, thereby improving the quality of clustering. The algorithm which integrates ELM method with K-means algorithm for clustering low dimensional data is the same as Algorithm 3.

4.4.2.1 ELM combined with RP (ELM-RP K-means).

As it has been mentioned in Section 4.4.1.1, the data points which are not linearly separable in the input space often become linearly separable after projecting them into high dimensional ELM feature space. To avoid the computational cost of performing clustering in the high dimensional ELM feature space, we use RP to reduce the dimensionality of data in ELM feature space and then we apply K-means algorithm in the reduced space.

In Algorithm 10, the low dimensional data is first projected to the high dimensional ELM feature space, then RP is used to reduce the dimensionality of ELM feature space while preserving the linear separability of the data and finally clustering using K-means is performed in the reduced space.

Algorithm 10: ELM-RP K-means Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of neurons in the first hidden layer ($L > d$)
r: number of neurons in the second hidden layer ($r < L$)
$g(x)$: activation function

1 Initialize first hidden layer parameters ($\mathbf{w}_i$, b_i), $i = 1, \ldots, L$ randomly;
2 Calculate the first hidden layer output matrix $\mathbf{H}$;
3 Initialize second hidden layer parameters ($\mathbf{w}'_j$, b'_j), $j = 1, \ldots, r$ using Eq. (4.3);
4 Compute the second hidden layer output matrix $\mathbf{H}'$;
5 Apply K-means algorithm on the matrix $\mathbf{H}'$;

4.5 Experimental study

In this chapter, 8 high dimensional data sets and 12 low dimensional data sets have been used to evaluate the performance of the ELM combined with RP approach for classification and clustering. Here it is pertinent to mention that there is no consistency in the literature about the minimum number of dimensions that renders a data set high dimensional. Data with as few as 10 dimensions are regarded as high dimensional in some works. On the other hand, numerous other works, specially those pertaining to image processing or bio-informatics have hundreds or thousands of dimensions [114].

In this chapter, we consider data sets with thousand or more dimensions as high dimensional data.

4.5.1 Data sets

The data sets used in this study are described as follows.

- *High dimensional data sets:* The Columbia Object Image Library (COIL20) data set contains gray-scale images of 20 objects, where each object has 72 images. Colon data set consists of 62 samples, where each sample has 2000 genes. The data set contains 22 normal and 40 tumor tissue samples. Global Cancer Map (GCM) data set consists of gene expression data for 190 tumor samples and 90 normal tissue samples. Leukemia data set contains 72 samples. Each sample has 7129 genes. The samples are either acute lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML). Lung cancer data set consists of gene expression data for 181 samples. The samples are either malignant pleural mesothe-lioma (MPM) or adenocarcinoma (ADCA). ORL data set contains face images of 40 distinct subjects, where each subject has 10 different images. Prostate cancer data set contains 136 tissue samples among which 77 samples are tumor and 59 are normal. Each sample is described by 12600 genes. Yale is a face recognition data set which contains face images of 15 individuals. Each individual has 11 images.

- *Low dimensional data sets:* The low dimensional data sets are described briefly in Section 2.4.1

The specifications of the high dimensional data sets and the low dimensional data sets have been provided in Table 4.1 and Table 4.2, respectively. It is to be noted that in this chapter, we consider data sets with thousand or more dimensions as high dimensional data.

4.5.2 Results and discussion

As mentioned already, the performance of ELM combined with RP approach for classification/clustering is evaluated on 8 high dimensional and 12 Low dimensional data sets. We have used the sigmoid function for nonlinear mapping, and the input weights

Table 4.1: Specifications of high dimensional data sets

Data sets	Patterns	Classification		Features	Classes
		Training	Testing		
COIL20	1440	1008	432	1024	20
Colon	62	30	32	2000	2
GCM	280	196	84	16063	2
Leukemia	72	38	34	7129	2
Lung	181	32	149	12533	2
ORL	400	280	120	1024	40
Prostate	136	102	34	12600	2
YALE	165	115	50	1024	15

Table 4.2: Specifications of low dimensional data sets

Data sets	Patterns	Classification		Features	Classes
		Training	Testing		
Balance	625	475	150	4	3
Cancer-Diagnostic	569	427	142	30	2
Cancer-Original	683	512	171	9	2
Cardiotocography-3	2126	1595	531	21	3
Cardiotocography-10	2126	1595	531	21	10
CNAE	1080	810	270	856	9
Dermatology	358	270	88	34	6
Glass	214	142	72	9	6
Iris	150	100	50	4	3
LIBRAS	360	270	90	90	15
Spam	1534	1150	384	57	2
USPST	2007	1505	502	256	10

and biases were generated either from a uniform distribution over [-1, 1] or using RP technique (i.e., Eq. (4.3)). All the simulations are carried out 20 times independently and the average performance is reported. For classification problems, the training and

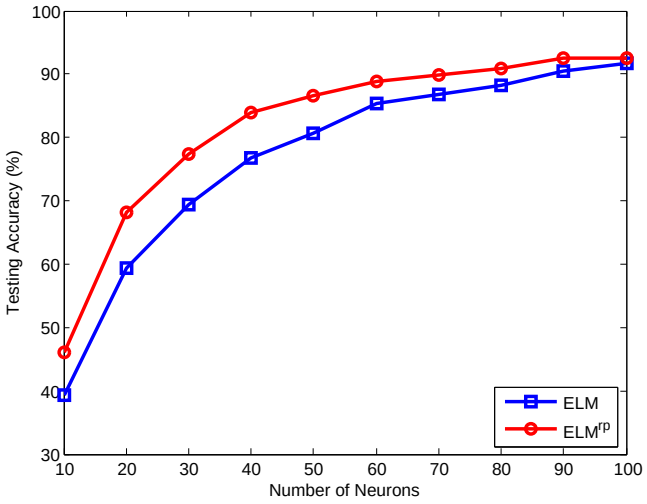

Figure 4.1: Classification performance on COIL20 with respect to different number of neurons

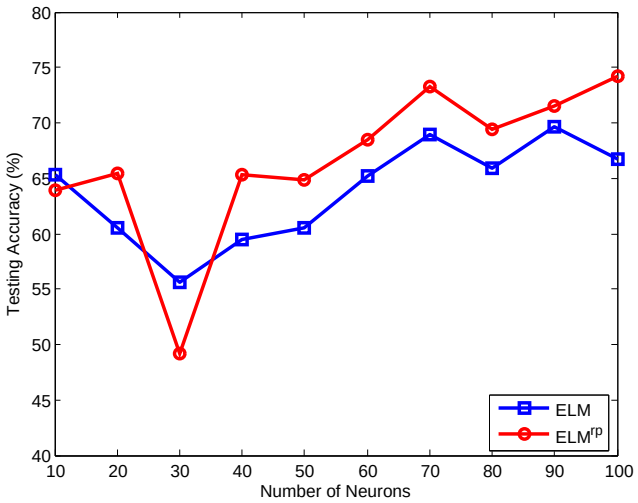

Figure 4.2: Classification performance on Colon with respect to different number of neurons

testing sets are randomly generated from the whole data set at each run according to Table 4.1 and Table 4.2.

Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8 show the classification performance of ELM and ELM$^{\text{rp}}$ algorithms on different high dimensional data sets with respect to different number of hidden neurons L. In these figures, the highest value of L is

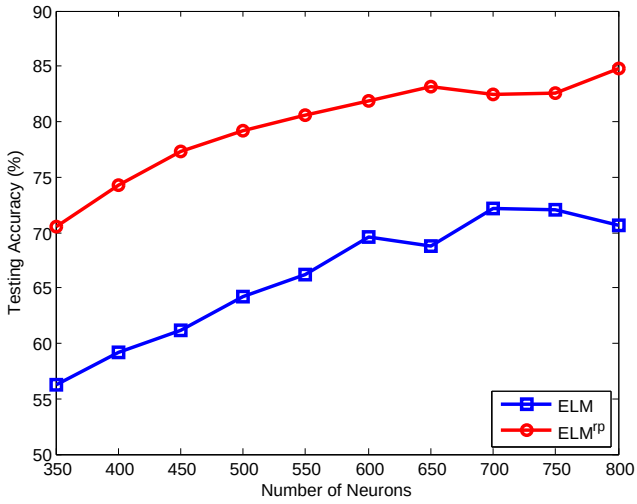

Figure 4.3: Classification performance on GCM with respect to different number of neurons

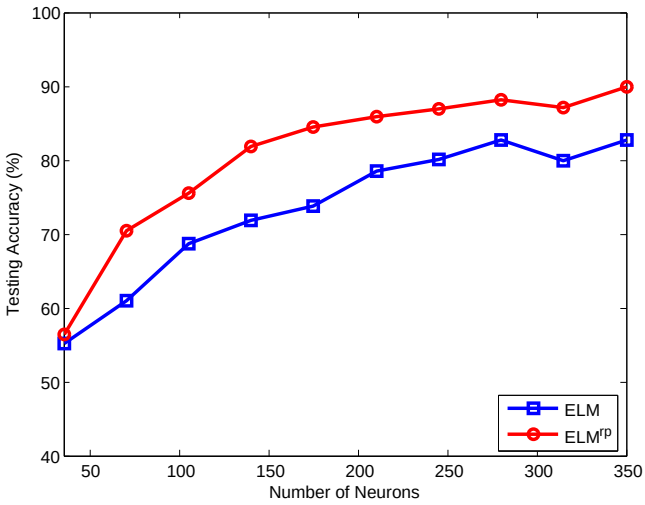

Figure 4.4: Classification performance on Leukemia with respect to different number of neurons

equivalent to the compression ratio 1:20 in the feature-projection stage $L < d$. From the results shown in these figures, it is obvious that ELM$^{\text{rp}}$ algorithm outperformed the ELM algorithm consistently on all data sets. It can also be observed that the classification accuracy of ELM$^{\text{rp}}$ in most cases increases gradually with the number of dimensions (i.e., neurons).

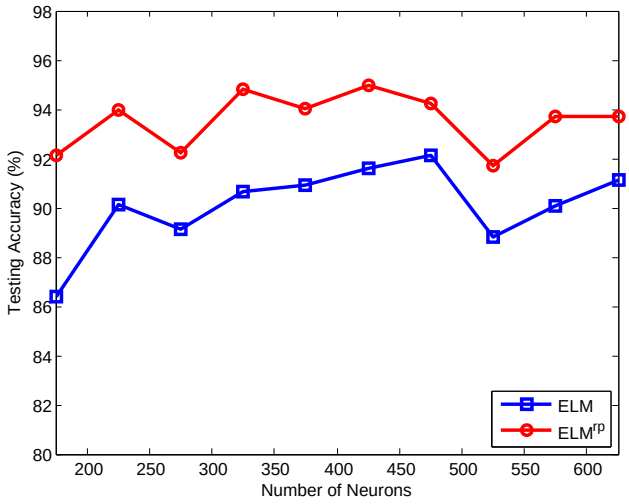

Figure 4.5: Classification performance on Lung with respect to different number of neurons

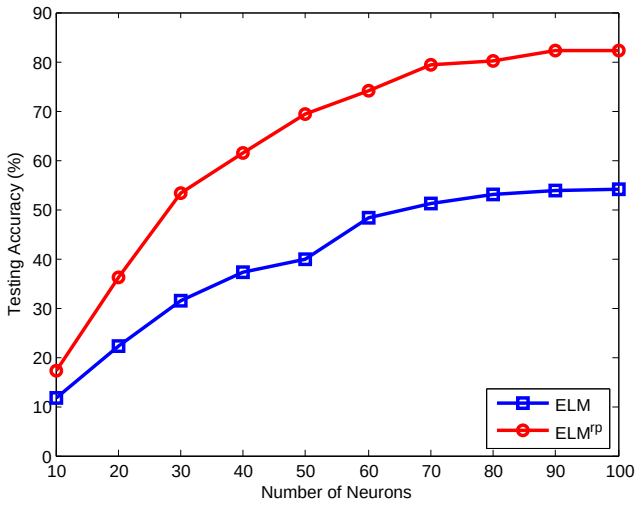

Figure 4.6: Classification performance on ORL with respect to different number of neurons

Figures 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 and 4.16 show the clustering performance of ELM K-means and ELM$^{\text{rp}}$ K-means on different high dimensional data sets with respect to different number of hidden neurons L. From these figures, we observe that ELM$^{\text{rp}}$ K-means algorithm has obtained satisfying results on all data sets and outperformed ELM K-means on six out of the eight data sets. Compared to K-means

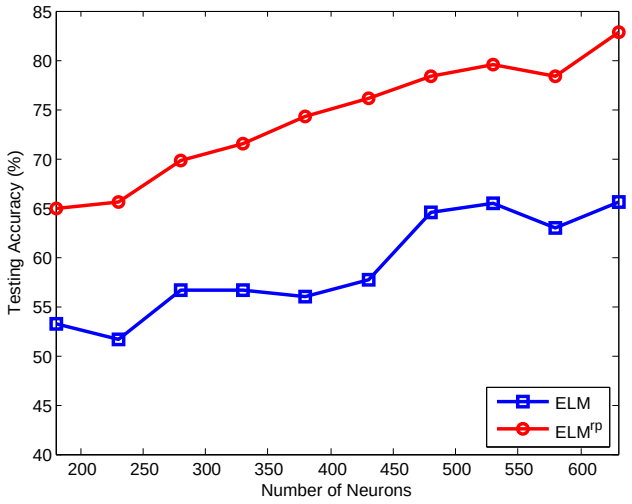

Figure 4.7: Classification performance on Prostate with respect to different number of neurons

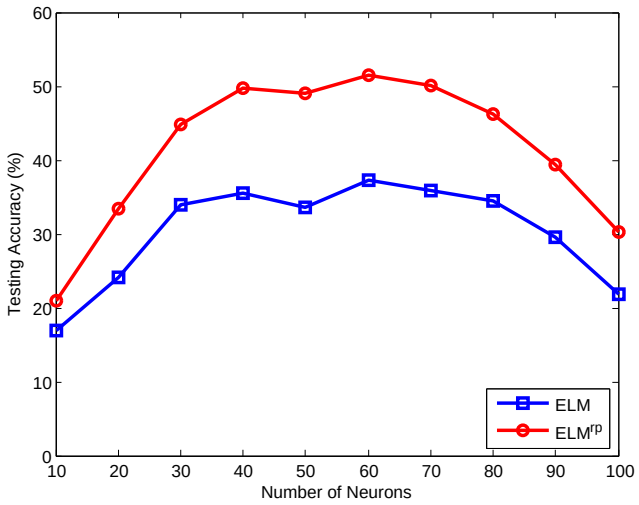

Figure 4.8: Classification performance on YALE with respect to different number of neurons

clustering in the input space, ELM$^{\text{rp}}$ K-means algorithm obtains similar or better results on four out of the eight data sets. The time required for ELM$^{\text{rp}}$ K-means to perform clustering is significantly reduced. For example, the time taken by K-means algorithm to perform clustering on COIL20 data set in the input space is 5.84 seconds, whereas ELM$^{\text{rp}}$ K-means algorithm with $L = 100$ takes 0.38 seconds. In addition, for

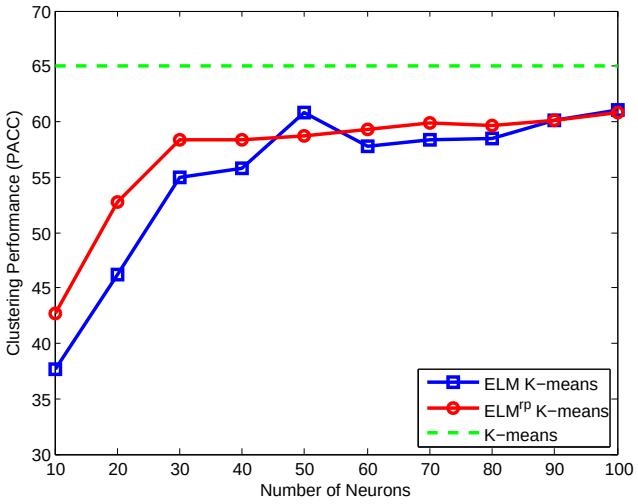

Figure 4.9: Clustering performance on COIL20 with respect to different number of neurons

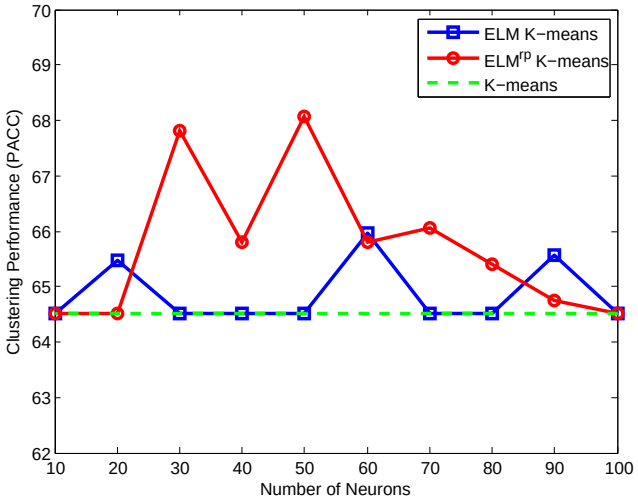

Figure 4.10: Clustering performance on Colon with respect to different number of neurons

RP-based clustering technique to obtain a high accuracy, a large number of dimensions is required. For example, ELM$^{\text{rp}}$ K-means algorithm obtains clustering accuracy of 62.95 and 72.22 on COIL20 and Leukemia data sets at $L = 200$ and $L = 700$, respectively. Thus, ELM$^{\text{rp}}$ K-means algorithm provides a trade-off between the clustering accuracy and computational complexity.

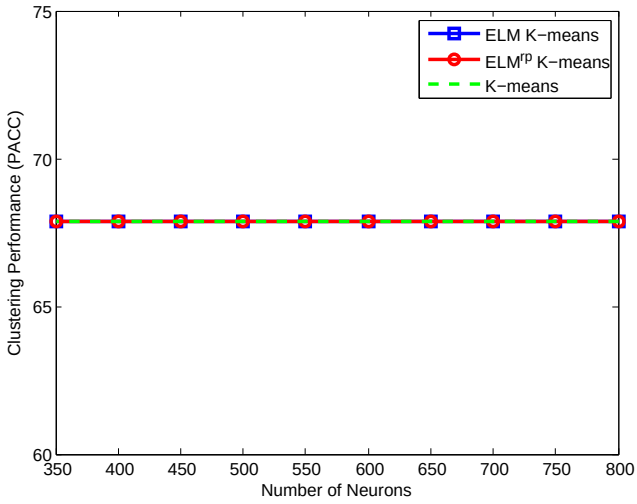

Figure 4.11: Clustering performance on GCM with respect to different number of neurons

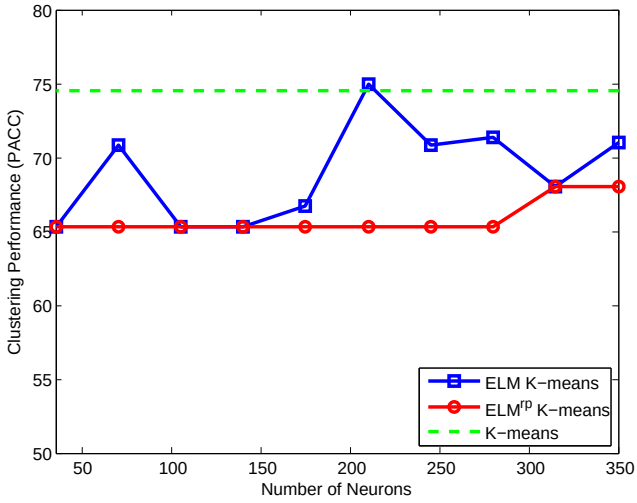

Figure 4.12: Clustering performance on Leukemia with respect to different number of neurons

Figures 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26, 4.27 and 4.28 show the classification performance of ELM and ELM-RP algorithms on different low dimensional data sets with respect to different number of neurons. In these figures, the ELM performs classification directly in the ELM feature space with different number of hidden neurons $L = [10, 20, \ldots, 100]$. While for ELM-RP, the data sets are first

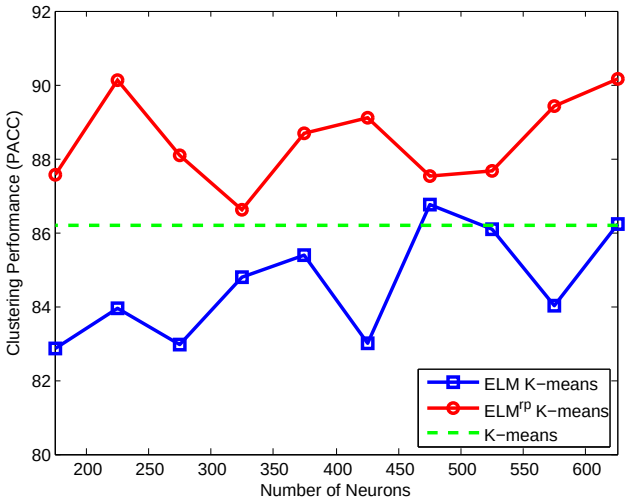

Figure 4.13: Clustering performance on Lung with respect to different number of neurons

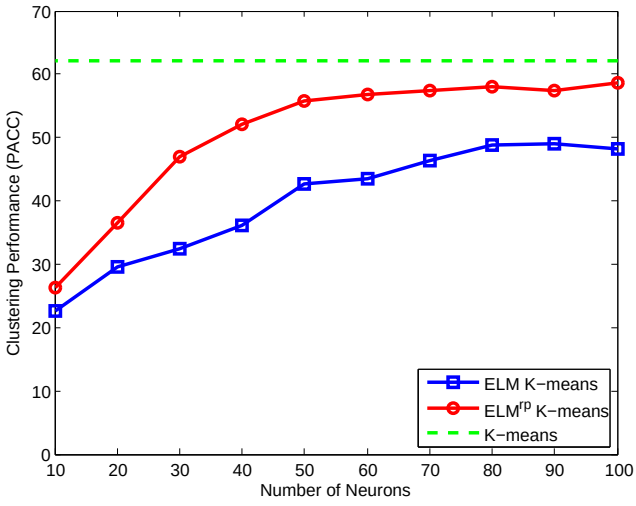

Figure 4.14: Clustering performance on ORL with respect to different number of neurons

projected to high dimensional space $L = 1000$ and then reduced using RP to different number of neurons $r = [10, 20, \ldots, 100]$. $\text{ELM}^{\text{L = 1000}}$ refers to ELM algorithm performing classification in ELM feature space with number of hidden neurons $L = 1000$. From these figures, we can observe that both ELM and ELM-RP algorithms have relatively similar performance on all data sets with different number of neurons $[10, 20, \ldots, 100]$.

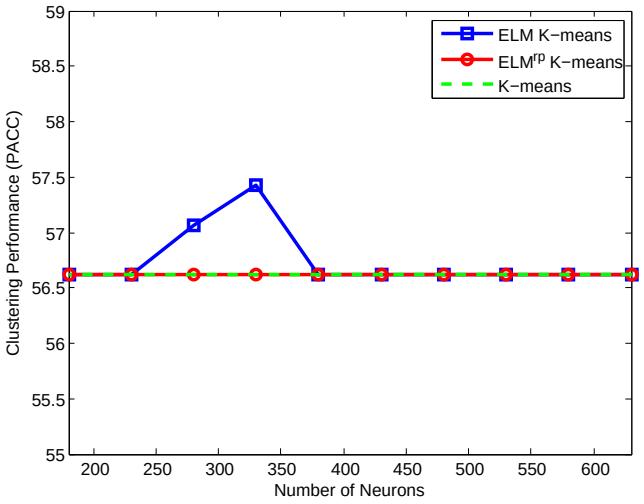

Figure 4.15: Clustering performance on Prostate with respect to different number of neurons

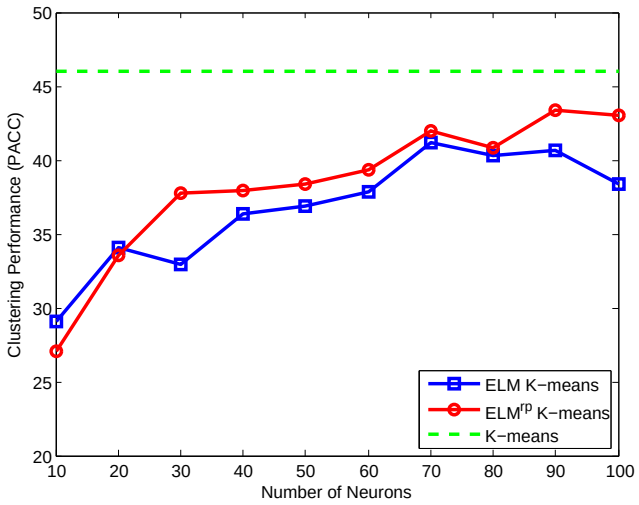

Figure 4.16: Clustering performance on YALE with respect to different number of neurons

Even though ELM-RP algorithm is performing classification in the reduced space of ELM feature space, it is capable of taking advantage of linear separability of data in the high dimensional ELM space, where $L = 1000$, and avoid over-fitting. It has been observed that $\text{ELM}^{\text{L = 1000}}$ algorithm obtains excellent training accuracy, but fails to generalize to unseen data. The performance of $\text{ELM}^{\text{L = 1000}}$ can be much improved by

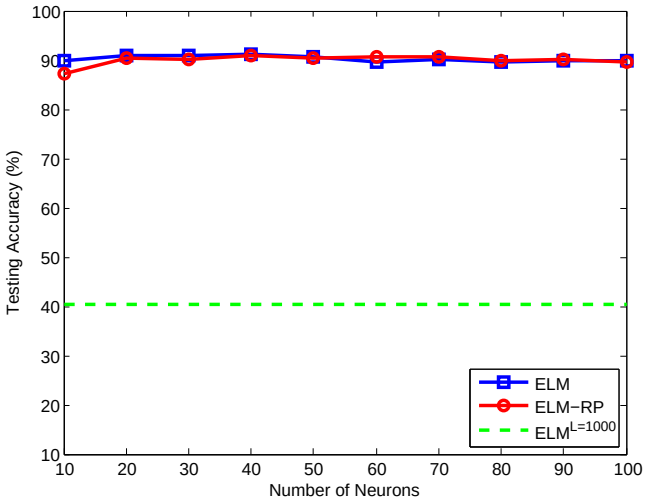

Figure 4.17: Classification performance on Balance with respect to different number of neurons

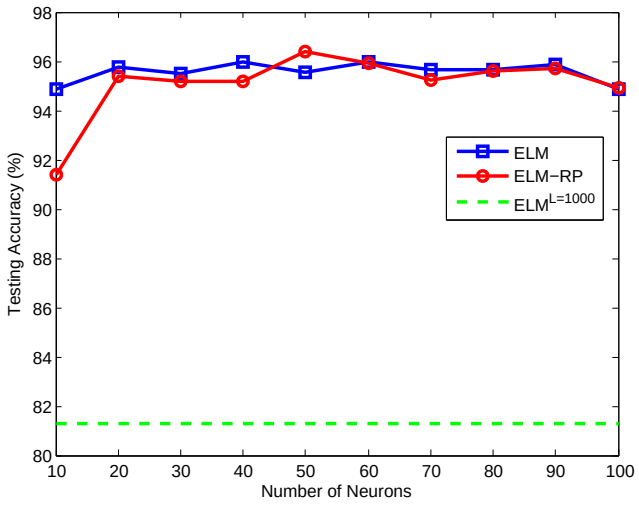

Figure 4.18: Classification performance on Cancer-Diagnostic with respect to different number of neurons

using regularization technique [7].

Figures 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37, 4.38, 4.39 and 4.40 show the clustering performance of ELM K-means and ELM-RP K-means on different low dimensional data sets with respect to different number of neurons. In these figures, ELM-RP K-means algorithm performs clustering in the reduced space of the high

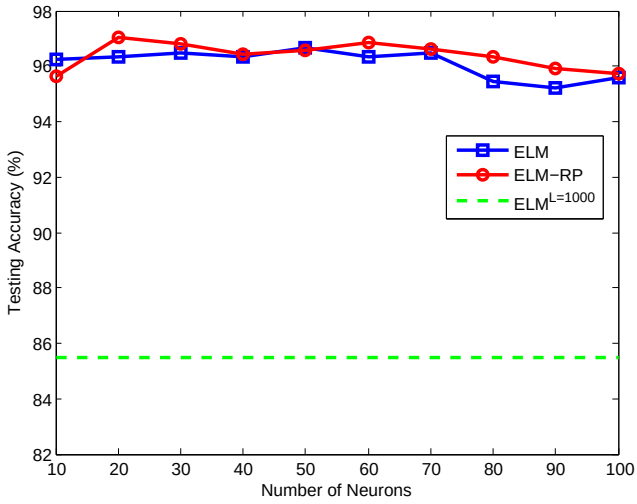

Figure 4.19: Classification performance on Cancer-Original with respect to different number of neurons

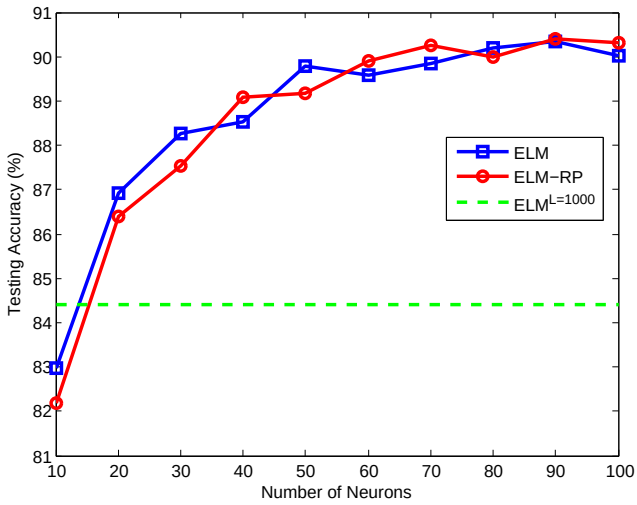

Figure 4.20: Classification performance on Cardiotocography-3 with respect to different number of neurons

dimensional ELM space, where $L = 1000$ in ELM space. ELM K-means performs clustering directly in the ELM feature space with different number of hidden neurons $L = [10, 20, \ldots, 100]$. ELM$^{\mathrm{L}\,=\,1000}$ K-means refers to K-means algorithm performing clustering in ELM feature space with number of hidden neurons $L = 1000$. These figures show that ELM-RP K-means algorithm obtains comparable results with ELM

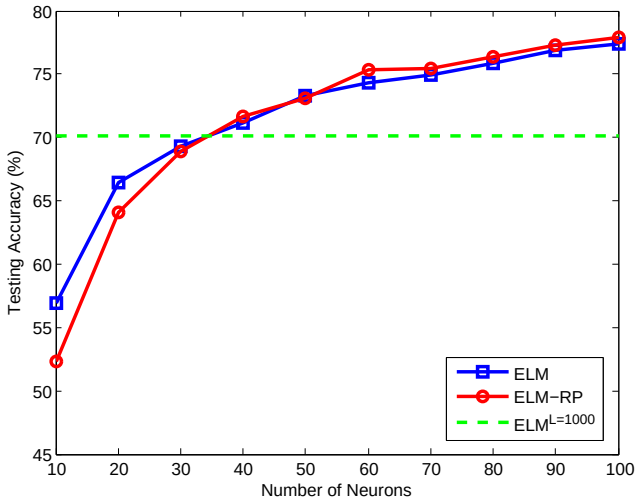

Figure 4.21: Classification performance on Cardiotocography-10 with respect to different number of neurons

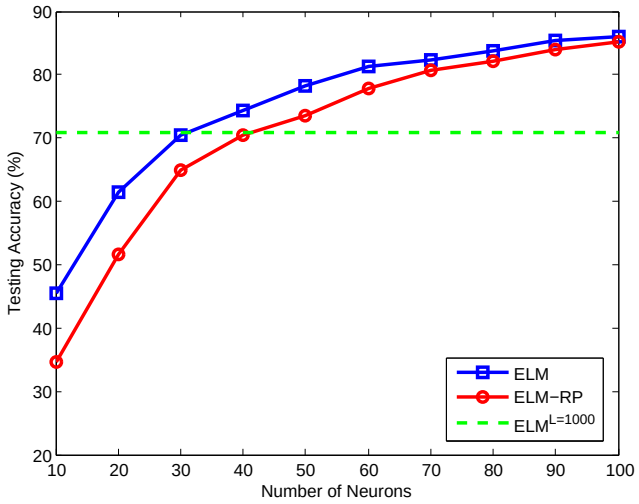

Figure 4.22: Classification performance on CNAE with respect to different number of neurons

K-means on most of the data sets and tends to obtain high accuracy with large number of neurons. It is also observed that ELM-RP K-means achieves satisfying results compared to $\text{ELM}^{\text{L}\,=\,1000}$ K-means which indicates that ELM-RP K-means algorithm has the ability to balance clustering quality and computational complexity.

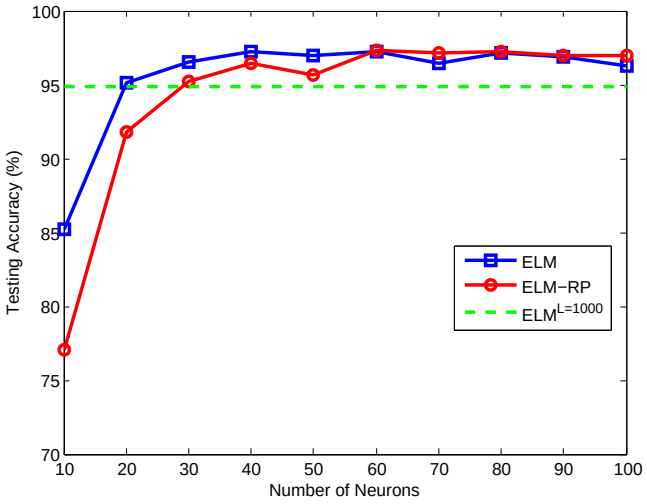

Figure 4.23: Classification performance on Dermatology with respect to different number of neurons

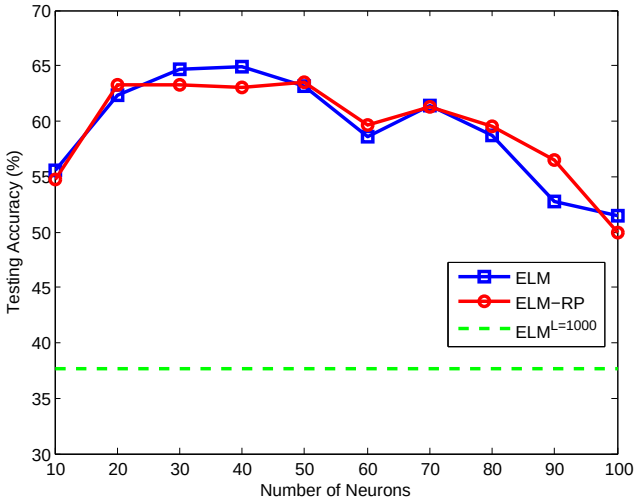

Figure 4.24: Classification performance on Glass with respect to different number of neurons

4.6 Conclusion

This chapter delivers a systematic investigation on the application of RP in conjunction with ELM for both low and high dimensional data classification and clustering. The capability of RP technique in performing dimensionality reduction without distorting

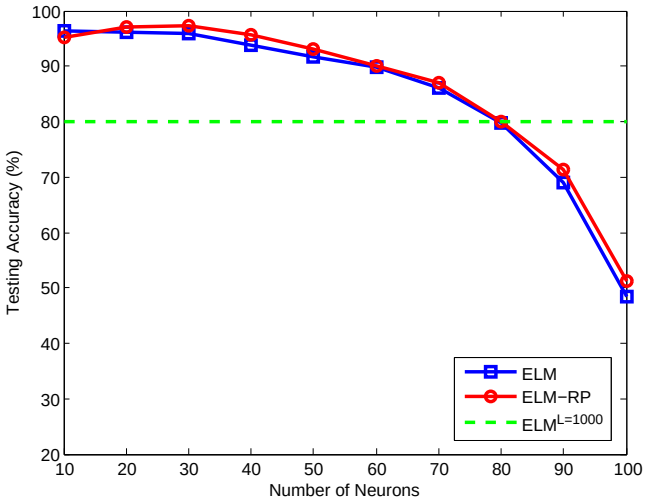

Figure 4.25: Classification performance on Iris with respect to different number of neurons

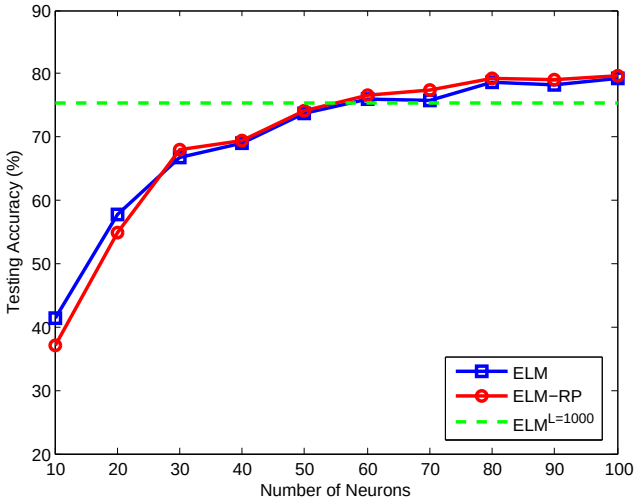

Figure 4.26: Classification performance on LIBRAS with respect to different number of neurons

significantly the structure of data has been investigated through different scenarios. The first scenario is for high dimensional data where RP is used to reduce the dimensionality by shrinking the number of neurons in the ELM hidden layer and then classification or clustering is performed in the reduced space. The second scenario is for low dimensional data in which the data is first projected to the high dimensional ELM feature space,

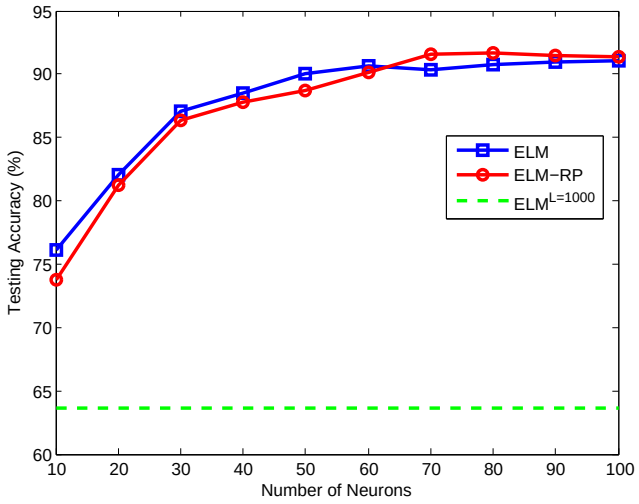

Figure 4.27: Classification performance on Spam with respect to different number of neurons

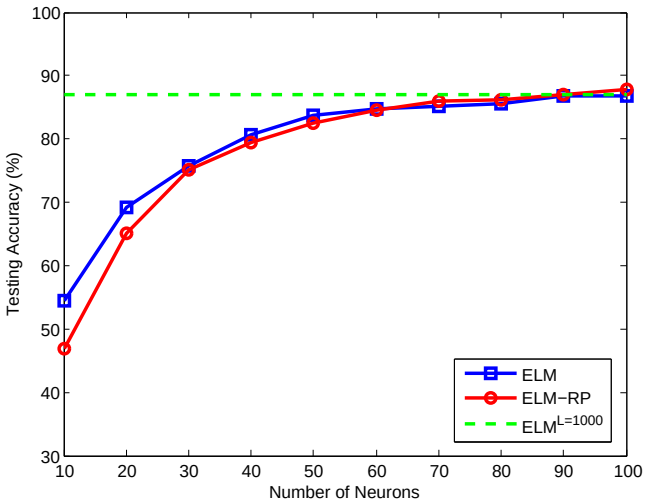

Figure 4.28: Classification performance on USPST with respect to different number of neurons

where the data is expected to be linearly separable and then RP is used to reduce the dimensions while preserving the linear separability of data. The experiments show that the integration of ELM with RP for data classification and clustering not only obtains satisfying results, but also provides a trade-off between generalization performance and computational complexity.

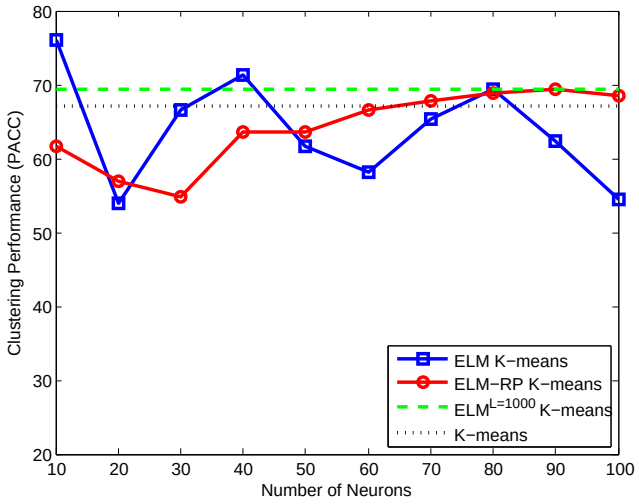

Figure 4.29: Clustering performance on Balance with respect to different number of neurons

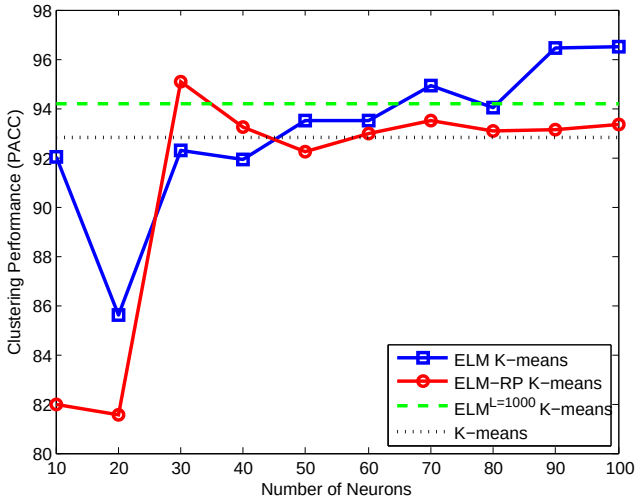

Figure 4.30: Clustering performance on Cancer-Diagnostic with respect to different number of neurons

.

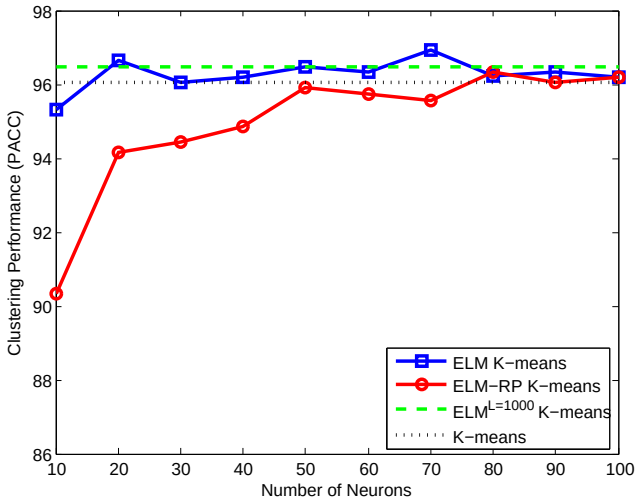

Figure 4.31: Clustering performance on Cancer-Original with respect to different number of neurons

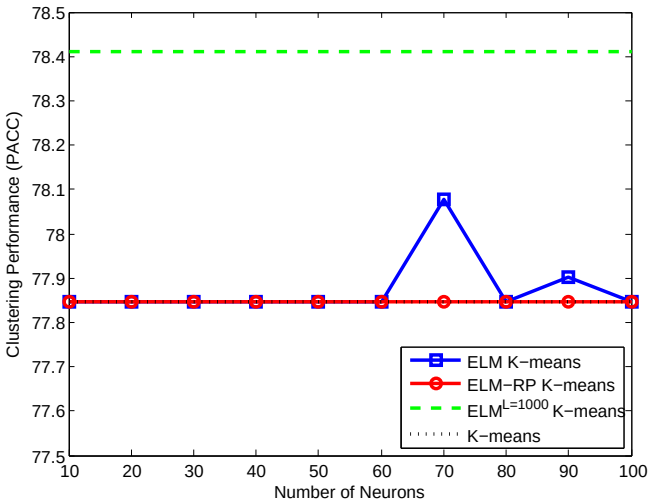

Figure 4.32: Clustering performance on Cardiotocography-3 with respect to different number of neurons

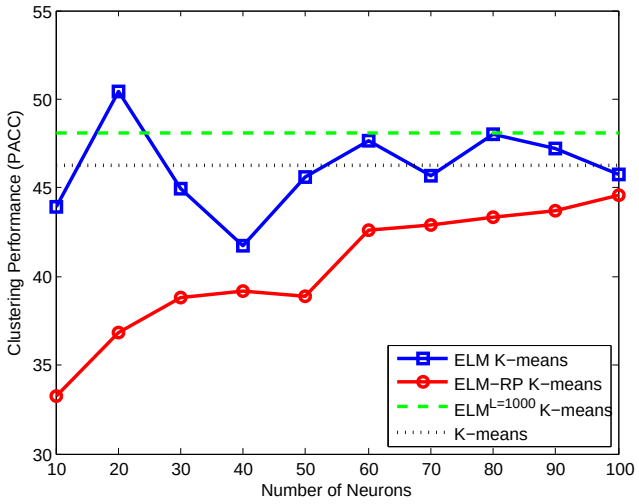

Figure 4.33: Clustering performance on Cardiotocography-10 with respect to different number of neurons

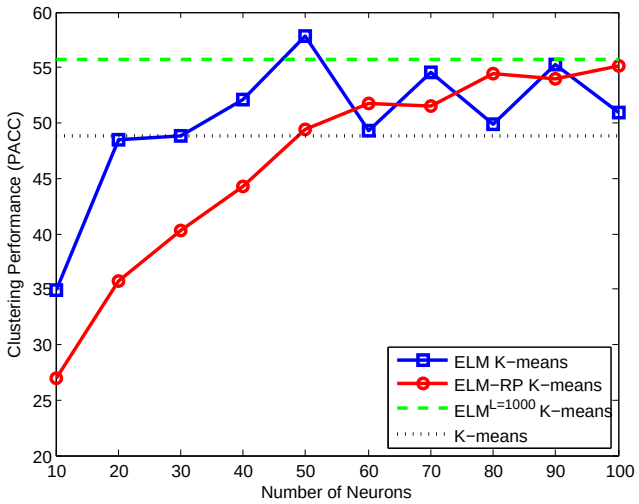

Figure 4.34: Clustering performance on CNAE with respect to different number of neurons

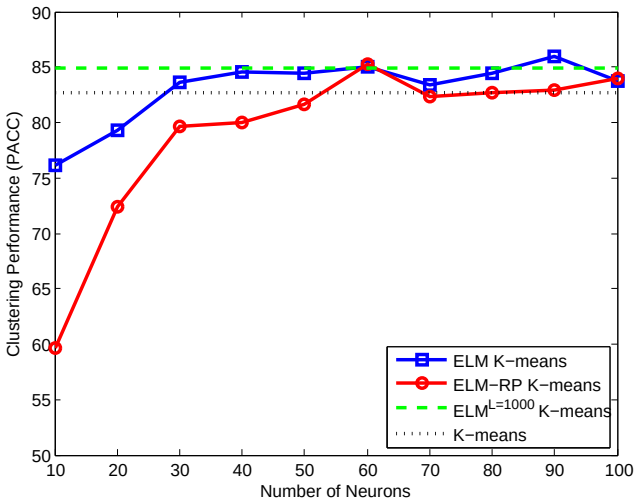

Figure 4.35: Clustering performance on Dermatology with respect to different number of neurons

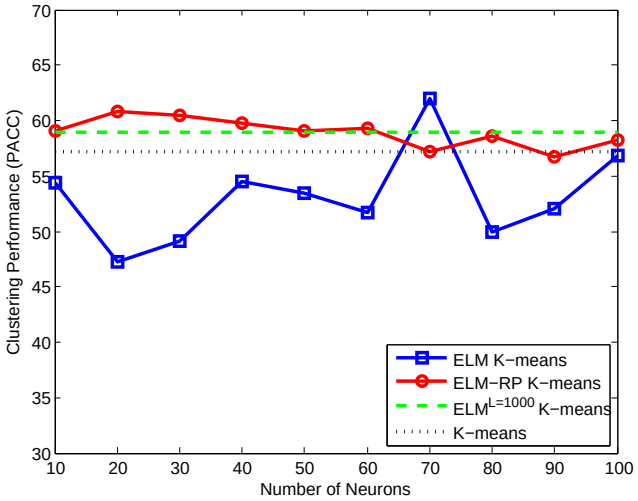

Figure 4.36: Clustering performance on Glass with respect to different number of neurons

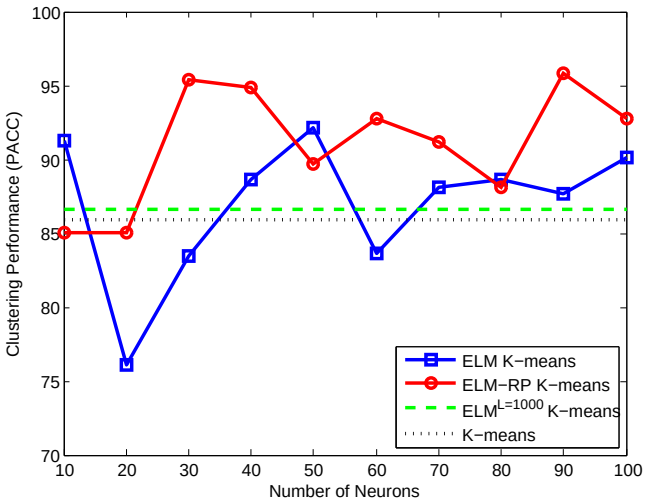

Figure 4.37: Clustering performance on Iris with respect to different number of neurons

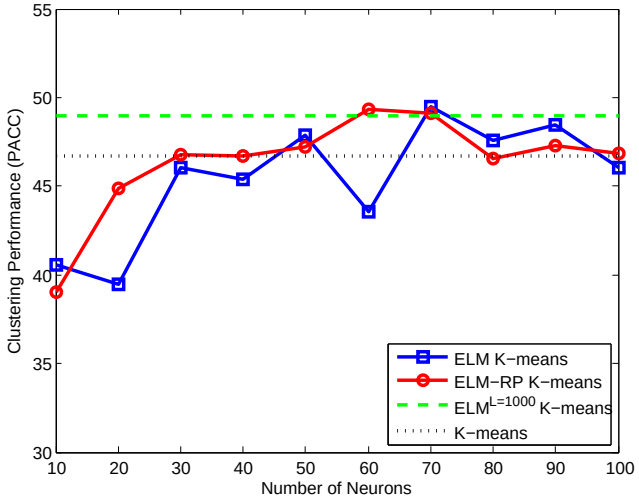

Figure 4.38: Clustering performance on LIBRAS with respect to different number of neurons

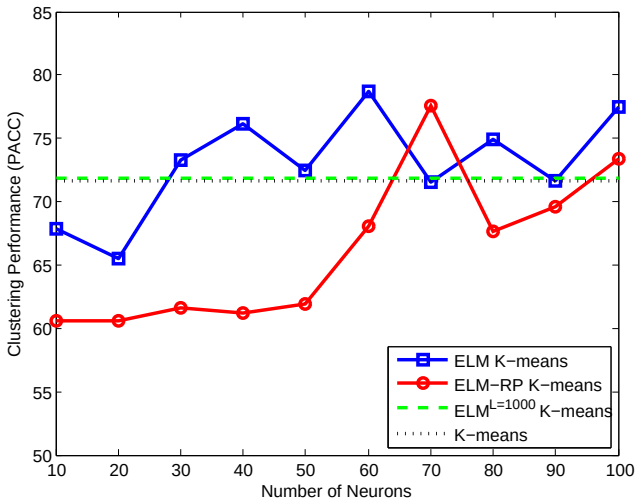

Figure 4.39: Clustering performance on Spam with respect to different number of neurons

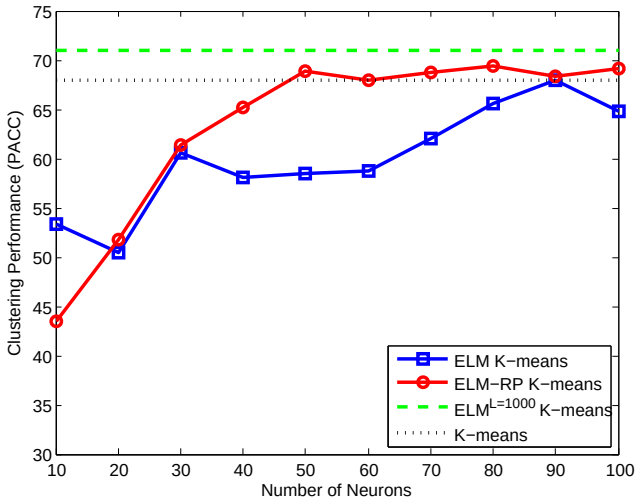

Figure 4.40: Clustering performance on USPST with respect to different number of neurons

Chapter 5

Two metaheuristic approaches for tuning extreme learning machine

5.1 Introduction

In ELM, since the input weights and hidden biases are randomly assigned and remain fixed during the learning process, the number of hidden neurons should be large enough in order to obtain good generalization performance [7]. Since ELM requires large number of hidden neurons, there may exist some suboptimal or unnecessary input weights and hidden biases, and, for some applications, the high number of neurons in the hidden layer renders the ELM respond slowly to unseen testing data [3]. The suitable number of neurons in ELM hidden layer is still an open problem. Several methods have been proposed to determine the suitable number of ELM hidden neurons. Rong et al. [72] proposed a pruned ELM (P-ELM) algorithm for pattern classification problems. The algorithm begins with large number of hidden neurons and then identifies the relevance of the hidden neurons to the class labels using some statistical measures. The hidden neurons which are irrelevant or have low relevance will be eliminated. Miche et al. [75] proposed optimally-pruned ELM (OP-ELM) for both classification and regression problems. The algorithm starts by constructing a large network using standard ELM algorithm. Then the multi-response sparse regression algorithm (MRSR) is used to rank the hidden neurons and finally the optimal number of hidden neurons is selected using leave-one-out (LOO) validation. Huang et al. [6] presented incremental extreme learning machine (I-ELM) algorithm in which the neurons are added one-by-one to the

hidden layer and the training rate is calculated after adding each neuron. The training will stop when the maximum number of neurons is reached or the training rate is less than the expected one. Lan et al. [10] proposed constructive method for selecting the ELM hidden neurons for regression problems. Their method begins with large number of hidden neurons and then each neuron is selected based on its significance and the selection stops when the unbiased risk estimation based criterion reaches its minimum. Zhu et al. [3] proposed evolutionary extreme learning machine (E-ELM) algorithm. E-ELM algorithm uses the differential evolutionary (DE) algorithm to tune the input weights and hidden biases while the output weights are calculated using Moore-Penrose (MP) generalized inverse. In E-ELM, the trial vector is produced by random generation method and the control parameters are manually defined by the user. To improve the performance of E-ELM, Cao et al. [2] proposed a new algorithm named self-adaptive evolutionary extreme learning machine (SaE-ELM) for SLFNs. In SaE-ELM, the input weights and hidden biases are optimized by the self-adaptive differential evolutionary algorithm and Moore-Penrose (MP) generalized inverse is utilized to analytically determine the output weights.

Optimizing the input weights and hidden neurons biases is a real-parameter optimization problem. Artificial bee colony (ABC) algorithm [52] and invasive weed optimization algorithm [61] have already proved their effectiveness for real-parameter optimization [115, 116]. This has motivated us to develop two approaches for SLFNs by incorporating the artificial bee colony algorithm and invasive weed optimization algorithm to tune the input weights and hidden neurons biases, and the extreme learning machine to compute the output weights. We refer to these two approaches as ABC-ELM and IWO-ELM. In ABC-ELM, the artificial bee colony algorithm is incorporated to tune the input weights and hidden neurons biases, and the extreme learning machine method to analytically determine the output weights. In IWO-ELM, the invasive weed optimization algorithm is used to optimize the input weights and hidden neurons biases, and the output weights are calculated using the extreme learning machine method.

The remaining part of this chapter is organized as follows: Section 5.2 and Section 5.3 present the proposed ABC-ELM algorithm and IWO-ELM algorithm respectively. Experimental results are presented and analysed in Section 5.4. Finally, Section 5.5 outlines some concluding remarks.

5.2 Proposed ABC-ELM algorithm

Due to the random initialization of input weights and hidden biases of ELM network, ELM requires a high number of neurons. Some of the input weights and hidden neurons biases could be unnecessary or suboptimal. And for some applications, the high number of hidden neurons may render ELM slow in responding to unknown testing data.

In this section, we have proposed a hybrid approach named ABC-ELM which uses ABC algorithm to tune the input weights and hidden neurons biases, and MP generalized inverse to determine the output weights.

5.2.1 Initialization

The ABC algorithm randomly generates the population of SN solutions (food source positions). Each solution $\mathbf{S}$ includes all input weights and hidden biases:

$$\mathbf{S} = [w_{11}, w_{12}, \ldots, w_{1L}, w_{21}, w_{22}, \ldots, w_{2L}, \ldots, w_{d1}, w_{d2}, \ldots, w_{dL}, b_1, b_2, \ldots, b_L] \quad (5.1)$$

where the dimension of each solution $\mathbf{S}$ is $D = (d+1) \times L$ and all w_{ij} and b_j are randomly generated within a specific range (e.g. [-1, 1]).

5.2.2 Solution fitness

After initializing all solutions, the corresponding output weights for each solution are calculated by using Eq. (1.6). For classification problems, the objective is to minimize the number of misclassified patterns. The fitness of the ith solution in the population is calculated as follows:

$$fit_i = \frac{1}{1 + MC_i} \quad (5.2)$$

where MC_i is the number of misclassified patterns obtained by using the solution i. The fitness function value needs to be maximized.

5.2.3 Generation of neighboring solution

In order to obtain a new solution $\mathbf{V}$ in the neighborhood of the current solution $\mathbf{S}$, we have utilized two different neighborhood procedures. The first procedure is fitness-based, whereas the second one is iteration-based. The first procedure is inspired by

spatial distribution concept in IWO in which the new solutions are generated in the neighborhood of the current solution with zero mean but varying variance. To produce a candidate solution $\mathbf{V}$ from the current solution $\mathbf{S}$, the first procedure uses the following expression:

$$\mathbf{V}_{ij} = \mathbf{S}_{ij} + rand(0, \sigma_{iter}) \tag{5.3}$$

where $i = 1, 2, \ldots, SN$ and $j = 1, 2, \ldots, D$. $rand$ is a function that returns normally distributed random number with zero mean and standard deviation σ_{iter}.

From Eq. (5.3), if we modify all elements of the solution $\mathbf{S}$ in order to produce a new solution $\mathbf{V}$, the algorithm may get stuck in local minima due to premature convergence. On the other hand, if we modify only one element, the algorithm will suffer from slow convergence. In the first procedure, we suggest that the amount of changes/modifications should be based on the fitness of the solutions and is calculated as follows:

$$\begin{aligned} Change &= \frac{D \times ChgPerct}{100} \\ ChgPerct &= \frac{fit_h - fit}{fit_h} \left(Chg_{max} - Chg_{min}\right) + Chg_{min} \end{aligned} \tag{5.4}$$

where fit is the fitness of the current solution. fit_h corresponds to the highest fitness of the population. Chg_{max} and Chg_{min} are respectively the maximum and the minimum percentage of the current solution elements that can be changed to produce a new solution.

Eq. (5.4) suggests that the fittest solutions in the population should not be perturbed much in order to obtain the neighboring solutions. The steps of generating neighboring solution using the first procedure are given in Algorithm 11.

The second procedure is based on the difference between the parameters/elements of the current solution and another solution chosen randomly from the population. To produce a candidate solution $\mathbf{V}$ from the current solution $\mathbf{S}$, the second procedure uses the following expression:

$$\mathbf{V}_{ij} = \mathbf{S}_{ij} + \phi_{ij}(\mathbf{S}_{ij} - \mathbf{S}_{kj}) \tag{5.5}$$

Algorithm 11: The first neighborhood procedure

input : Let $\mathbf{S}_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,D}\}$ be current solution and $\mathbf{V}_i$ is its neighbor

Set $\mathbf{V} = \mathbf{S}$;
Calculate the value of the variable $Change$ using Eq. (5.4);
for $k = 1$ *to* $Change$ **do**
    Generate an integer number j in the interval [1, D];
    $\mathbf{V}_{ij} = \mathbf{S}_{ij} + rand(0, \sigma_{iter})$;
end

where $i, k \in \{1, 2, \ldots, SN\}$, $i \neq k$ and $j = 1, 2, \ldots, D$. ϕ_{ij} is a random number between [-1, 1] which controls the production of the neighboring solution around the current solution.

In the second procedure, the elements of the current solution are copied to the neighboring solution with probability P_{copy}. The neighboring solutions will be generated by varying the value of P_{copy} according to the number of iterations. The P_{copy} varies from a previously defined initial value, $P_{initial}$ to a previously defined final value, P_{final}. The value of P_{copy} at iteration $iter$ is calculated as follows [103]:

$$P_{copy} = \frac{P_{final} - P_{initial}}{iter_{max}}(iter) + P_{initial} \tag{5.6}$$

where $iter_{max}$ is the maximum number of iterations. From Eq. (5.6), the probability of copying the elements of the current solution to the neighboring solution increases as the algorithm proceeds. The pseudo-code of generating neighboring solution using second procedure is given in Algorithm 12.

In both neighborhood procedures, if the produced parameter/element exceeds its predetermined limit, the value of the parameter will be set to the corresponding limit value.

5.2.4 ABC-ELM algorithm

The ABC-ELM algorithm starts by randomly generating the population of SN solutions according to the encoding specified in Eq. (5.1) and then allocating the employed bees to the initial solutions. After initializing all solutions, the corresponding output weights for each solution are first calculated by using Eq. (1.6) and then the fitness of each solution is evaluated using Eq. (5.2).

Algorithm 12: The second neighborhood procedure

input : Let $\mathbf{S}_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,D}\}$ be current solution and $\mathbf{V}_i$ is its neighbor

Calculate the value of the variable P_{copy} using Eq. (5.6);
Generate an integer number k in the interval [1, SN];
for $j = 1$ *to* D **do**
    Generate a random number r in the interval [0, 1];
    if $r < P_{copy}$ **then**
        Set $\mathbf{V}_{ij} = \mathbf{S}_{ij}$;
    else
        Set $\mathbf{V}_{ij} = \mathbf{S}_{ij} + \phi_{ij}(\mathbf{S}_{ij} - \mathbf{S}_{kj})$;
    end
end

After associating the employed bees with the initial solutions, every employed bee produces a new solution by using the steps in Algorithm 11 or Algorithm 12 and then evaluates its fitness. The new solution replaces the old one if its fitness is better than the fitness of the old solution. The employee bees share the quality information of their solutions with the onlooker bees. Each onlooker bee chooses a solution with a probability related to its fitness value. The probability of choosing a solution i is defined as in Eq. (3.3).

As in the case of an employed bee, each onlooker bee produces a new solution in the neighborhood of its selected solution and compares the fitness of the new solution with the fitness of the selected solution. The new solution replaces the selected solution in case new solution has better fitness. The solution whose fitness value has not improved over predetermined number of trials *limit* is abandoned, and the associated employed bee becomes a scout. The scout produces a new solution randomly using Eq. (5.1). The process of replacing not improving solutions enables the algorithm to avoid suboptimal solutions. The search processes of the employed, onlooker and scout bees are repeated until the termination condition is met. The pseudo-code of the ABC-ELM algorithm is given in Algorithm 13. The version of ABC-ELM algorithm which uses Algorithm 11 to generate the neighboring solution will be referred to as ABC(I)-ELM, whereas the version which uses Algorithm 12 will be referred to as ABC(II)-ELM.

Algorithm 13: The pseudo-code of the ABC-ELM algorithm

input : Given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$

Construct ELM network with L hidden neurons and activation function $g(x)$;
Initialize the population of SN solutions $\mathbf{S}_{ij}, i = 1, 2, \ldots, SN, j = 1, 2, \ldots, D$. Assign employed bees to the solutions;
Compute the output weights for each solution $\mathbf{S}_i$ using Eq. (1.6);
Evaluate the fitness $fit(\mathbf{S}_i)$ of the population $\mathbf{S}_i$ using Eq. (5.2);
Set $trial_i = 0$, $i = 1, 2, \ldots, SN$;
repeat
    /* Employed bees phase */;
    for $i = 1$ *to* SN **do**
        Produce a new solution $\mathbf{V}_i$ from $\mathbf{S}_i$ using Algorithm 11 or Algorithm 12;
        Evaluate the fitness of the new solution using Eq. (5.2);
        if $fit(\boldsymbol{V}_i) > fit(\boldsymbol{S}_i)$ **then**
            Replace $\mathbf{S}_i$ with $\mathbf{V}_i$ and set $trial_i = 0$;
        else
            $trial_i = trial_i + 1$;
        end
    end
    Calculate the probability values p_i for the solutions using Eq. (3.3);
    /* Onlooker bees phase */;
    for *each onlooker bee* **do**
        Select a solution $\mathbf{S}_i$ depending on p_i;
        Produce a new solution $\mathbf{V}_i$ from $\mathbf{S}_i$ using Algorithm 11 or Algorithm 12;
        Evaluate the fitness of the new solution using Eq. (5.2);
        if $fit(\boldsymbol{V}_i) > fit(\boldsymbol{S}_i)$ **then**
            Replace $\mathbf{S}_i$ with $\mathbf{V}_i$ and set $trial_i = 0$;
        else
            $trial_i = trial_i + 1$;
        end
    end
    /* Scout bees phase */;
    for *each solution* $\boldsymbol{S}_i$ **do**
        if $trial_i > limit$ **then**
            Replace $\mathbf{S}_i$ with a randomly produced solution;
        end
    end
    Memorize the best solution achieved so far;
until *termination condition is satisfied*;

5.3 Proposed IWO-ELM algorithm

Similar to ABC-ELM algorithm, the IWO-ELM algorithm uses the same solution encoding as specified in Eq. (5.1). The IWO-ELM algorithm starts with a set of initial weed solutions (WN_{init}) randomly generated and dispersed over the search space. It computes the output weights for all solutions by using Eq. (1.6) and then evaluates the fitness of each weed solution using Eq. (5.2). Each weed solution produces certain number of seed solutions depending on its own, the highest and lowest fitness of the colony. The number of seeds a weed can produce is calculated by using Eq. (1.7). Each seed will be produced in same way as the neighboring solution generated in ABC-ELM algorithm. In other words, the seed solution will be produced by using Algorithm 11 or Algorithm 12 where the generating weed represents the current solution and the seed represents the neighboring solution. The seed solutions will be included in the population and their fitnesses are evaluated. This process will continue in every iteration until the maximum number of weed solutions WN_{max} in the colony is reached. Once the maximum number of weed solutions is reached, a competitive exclusion process is performed to eliminate the solutions with poor fitness. In competitive exclusion process, the weeds and the newly generated seed solutions are together sorted according to their fitness. Next, only the fittest weeds are allowed to replicate and survive to the next generation and the weeds with poor fitness are eliminated to maintain the maximum allowable number of solutions (WN_{max}) in a colony. The pseudo-code of the IWO-ELM algorithm is given in Algorithm 14, where WN_c represents the total number of weeds in the colony at any instance of time. The version of IWO-ELM algorithm which uses Algorithm 11 to generate the neighboring solution will be referred to as IWO(I)-ELM, whereas the version which uses Algorithm 12 will be referred to as IWO(II)-ELM.

5.4 Experimental study

In this chapter, seven benchmark classification data sets have been used to evaluate the performance of the proposed ABC-ELM and IWO-ELM algorithms. These data sets can be downloaded from the UCI machine learning repository[1]. The specifications of these data sets are given in Table 5.1.

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Algorithm 14: The pseudo-code of the IWO-ELM algorithm

input : Given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$

Construct ELM network with L hidden neurons and activation function $g(x)$;
Initialize the population of WN_{init} weed solutions $\mathbf{S}_{ij}, i = 1, 2, \ldots, WN_{init}, j = 1, 2, \ldots, D$;
Compute the output weights for each solution $\mathbf{S}_i$ using Eq. (1.6);
Evaluate the fitness $fit(\mathbf{S}_i)$ of the population $\mathbf{S}_i$ using Eq. (5.2);
Set $WN_c = WN_{init}$;
Sort the weeds WN_c according to their fitness;
if $WN_c > WN_{max}$ **then**
    Set $WN_c = WN_{max}$;
end
repeat
    Set $WN_{colony} = WN_c$;
    for $i = 1$ *to* WN_{colony} **do**
        Calculate the number of seeds N_{seed} for solution $\mathbf{S}_i$ using Eq. (1.7);
        for $j = 1$ *to* N_{seed} **do**
            Set $WN_c = WN_c + 1$;
            Generate seed solution $\mathbf{S}_j$ from solution $\mathbf{S}_i$ using Algorithm 11 or Algorithm 12;
        end
    end
    Sort the weeds WN_c according to their fitness;
    Memorize the best solution achieved so far;
    if $WN_c > WN_{max}$ **then**
        Set $WN_c = WN_{max}$;
    end
until *termination condition is satisfied*;

Table 5.1: Specifications of benchmark classification data sets

Data sets	Data		Features	Classes
	Training	Testing		
Diabetes	500	268	8	2
Disease	100	170	13	2
Iris	100	50	4	3
Wine	100	78	13	3
Image segmentation	1900	410	18	7
Satellite image	5400	1000	36	7
Shuttle	50750	7250	9	7

5.4.1 Data sets

The data sets considered in this chapter can be described briefly as follows. The Pima Indians Diabetes data set (Diabetes) is used to diagnose whether the patient is diabetes positive or not. It consists of 768 patterns from two different classes. The heart Disease data set (Disease) consists of 13 attributes and two classes. The class refers to the presence of heart disease in the patient. The Fisher Iris data [92] is one of the most popular data sets to test the performance of novel methods in pattern recognition and machine learning. There are three classes in this data set, each having 50 patterns. One of the classes is linearly separable from the other two, while the remaining two are not linearly separable. Wine data is obtained from a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. The data set consists of 178 patterns belonging to three types of wines. Image segmentation data set consists of seven outdoor images. The images are brickface, sky, foliage, cement, window, path and grass. The images were hand-segmented to create a classification for every pixel. Satellite image data set is composed of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. Shuttle data set deals with the positioning of radiators in the Space Shuttle and consists of 58000 patterns belonging to seven different classes. Approximately 80% of the data belongs to one of the seven classes.

5.4.2 Results and discussion

The proposed ABC-ELM and IWO-ELM algorithms are tested on seven benchmark classification data sets and their performance is compared with the techniques given in [2] and [3]. All the simulations are carried out 50 independent times and the average performance is reported. The training and testing sets are randomly generated from the whole data set at each run according to the specifications given in Table 5.1. We have used the sigmoid function for nonlinear mapping. The initial solutions of ABC-ELM and IWO-ELM algorithms were generated from a uniform distribution over [-1, 1]. For ABC-ELM, the number of employed bees and the number of onlookers are set to be equal to the number of food sources (SN), which is 10, the $limit = 10$. For IWO-ELM, the number of initial solutions $WN_{init} = 5$, the maximum number of solutions allowed in the colony $WN_{max} = 10$, $S_{max} = 5$ and $S_{min} = 0$. For both ABC-ELM and IWO-ELM, we set $nmi = 3$, $\sigma_{initial} = 0.8$, $\sigma_{final} = 0.1$, $Chg_{max} = 20$, $Chg_{min} = 3$, $P_{initial} = 0$ and $P_{final} = 1$. Both ABC-ELM and IWO-ELM algorithms terminate if they have executed for $iter_{max} = 100$ iterations. All the simulations are carried out in MATLAB environment on a system with Core i5-2400, 3.10 GHZ CPU and 4 GB RAM. The performance of ABC-ELM and IWO-ELM algorithms on the first four classification data sets in Table 5.1 has been compared with the techniques given in [2], whereas the comparison on the remaining data sets is done with the methods given in [3]. Actually, first four classification data sets in Table 5.1 were used in [2] to test the performance of various approaches proposed. On the other hand, [3] used the remaining data sets. Table 5.2 and Table 5.3 show the classification performance of the proposed approaches along with the methods presented in [2] and [3] respectively. For ABC-ELM and IWO-ELM algorithms, similar to the simulations in [2] and [3], the number of hidden neurons is gradually increased and the one with the best testing accuracy is reported in Tables 5.2 and 5.3.

In Table 5.2, we report the best, average and worst performance of E-ELM and DE-LM over eight different combinations of generation strategies and their control parameters as specified in [2]. From Table 5.2, we can see that IWO-ELM obtains the best classification performance on Iris and Wine data sets while ABC-ELM is the second best on these two data sets. Both ABC-ELM and IWO-ELM achieve good generalization performance on Disease data set with less number of neurons. SaE-ELM

obtains the best testing accuracy on Diabetes and Disease data sets and both ABC-ELM and IWO-ELM achieve satisfying results on Diabetes data set with small number of neurons. Both ABC-ELM and IWO-ELM outperformed the average performance of E-ELM and DE-LM on all the data sets. We can observe that the training time of ABC-ELM and IWO-ELM is higher than that of the other approaches except for SVM and CFWNN-ELM is the fastest. From Table 5.2, we can also observe the standard deviation of ABC-ELM and IWO-ELM is small compared to other methods. IWO-ELM outperformed ABC-ELM on all the data sets, but ABC-ELM obtains comparable results on these data sets with less training time. Both ABC-ELM and IWO-ELM achieve better results with the second neighborhood procedure (i.e. Algorithm 12) than that with the first neighborhood procedure (i.e., Algorithm 11).

The data sets in Table 5.2 are relatively small and the number of classes at maximum equals to 3. The data sets used in [3] are relatively large and the number of classes is high. We have compared the performance of ABC-ELM and IWO-ELM on these data sets with the methods presented in [3] and the results are reported in Table 5.3. The results of our implementation of ELM on these data sets are reported in Table 5.3. From Table 5.3, we observe that both ABC-ELM and IWO-ELM outperformed the other algorithms on all the data sets. IWO-ELM obtains better results than ABC-ELM on Image segmentation and Shuttle data sets. We also observe that IWO-ELM achieves better results with the first neighborhood procedure than those with the second neighborhood procedure. Compared to ELM, the integration of ABC and IWO helps to relatively reduce the number of hidden neurons in ABC-ELM and IWO-ELM, respectively and also helps to improve the generalization performance. E-ELM obtains good generalization performance with less number of neurons.

5.5 Conclusion

In this chapter, we have proposed two metaheuristic approaches for training single-hidden layer feedforward neural networks. The two approaches (referred to as ABC-ELM and IWO-ELM) use artificial bee colony (ABC) algorithm and invasive weed optimization (IWO) algorithm to tune the network input weights and hidden biases, and extreme learning machine (ELM) to determine the output weights analytically. Both ABC and IWO algorithms are simple, robust and easy to understand and implement.

The proposed ABC-ELM and IWO-ELM have been tested on seven benchmark classification data sets and their performance is compared with other techniques which are widely used in the literature. Experimental results show that the proposed approaches achieve good generalization performance in comparison to other techniques.

Table 5.2: Comparison of classification performance of the techniques given in [2] and the proposed ABC-ELM and IWO-ELM algorithms

Data sets	Methods	Testing (%)		Training time (s)	Neurons/SVs
		Accuracy	Dev		
Diabetes	SaE-ELM	79.55	2.68	5.6422	20
	LM	74.73	3.2	2.8978	20
	CFWNN-ELM	78.02	2.56	0.0102	35
	SVM	77.31	2.73	364.6	62.28
	E-ELM^{best}	78.09	3.53	1.475	30
	E-ELM^{average}	77	3.4	1.2491	23
	E-ELM^{worst}	76.06	3.52	1.1754	22
	DE-LM^{best}	77.5	2.08	4.6297	2
	DE-LM^{average}	76.54	2.41	5.0405	2.25
	DE-LM^{worst}	75.94	2.97	4.3526	2
	ABC(I)-ELM	77.13	2.02	4.3326	10
	ABC(II)-ELM	77.19	1.89	4.4723	13
	IWO(I)-ELM	77.03	1.68	7.024	9
	IWO(II)-ELM	77.31	1.69	8.5576	13
Disease	SaE-ELM	82.53	3.86	0.7086	18
	LM	71.75	6.67	0.3066	20
	CFWNN-ELM	76.85	3.43	0.0016	30
	SVM	76.1	3.46	6.7094	81.6
	E-ELM^{best}	81.62	4.29	0.2247	16
	E-ELM^{average}	81.01	3.67	0.2425	15
	E-ELM^{worst}	80.05	3.44	0.2588	14
	DE-LM^{best}	79.06	2.28	2.8203	2
	DE-LM^{average}	77.99	3.06	2.8418	2
	DE-LM^{worst}	77.29	3.13	2.7469	2
	ABC(I)-ELM	81.35	2.39	1.1965	7
	ABC(II)-ELM	81.49	2.56	1.0568	8
	IWO(I)-ELM	81.61	2.81	2.6548	11
	IWO(II)-ELM	81.92	2.41	2.8829	15
Iris	SaE-ELM	97.2	4.12	0.2289	18
	LM	95.4	3.19	0.3641	10
	CFWNN-ELM	95.92	3.05	0.0009	20
	SVM	94.36	2.76	4.5488	23.3
	E-ELM^{best}	96.48	3.67	0.1522	16
	E-ELM^{average}	96.17	3.48	0.1596	16.5
	E-ELM^{worst}	95.8	3.78	0.15	16
	DE-LM^{best}	96.94	2.21	1.2546	6
	DE-LM^{average}	96.29	2.39	1.3177	5.25
	DE-LM^{worst}	95.87	2.2	1.3566	6
	ABC(I)-ELM	97.2	2.46	1.6041	17
	ABC(II)-ELM	97.08	2.39	1.2091	12
	IWO(I)-ELM	97.04	2.40	3.4738	18
	IWO(II)-ELM	97.6	2.21	3.1772	17
Wine	SaE-ELM	97.95	2.54	0.305	22
	LM	92.97	4.36	0.4988	20
	CFWNN-ELM	95.15	3.13	0.0028	30
	SVM	97.48	1.57	5.8941	47.3
	E-ELM^{best}	97.28	2.71	0.2319	22
	E-ELM^{average}	96.85	2.65	0.2608	23.25
	E-ELM^{worst}	96.09	2.88	0.3214	30
	DE-LM^{best}	97.11	1.65	1.9985	4
	DE-LM^{average}	96.59	1.96	1.8947	4.25
	DE-LM^{worst}	96.03	1.99	2.2023	6
	ABC(I)-ELM	97.79	1.72	1.8322	19
	ABC(II)-ELM	97.97	1.97	1.6536	20
	IWO(I)-ELM	98	2.07	3.3855	15
	IWO(II)-ELM	98.13	1.64	3.6714	20

Table 5.3: Comparison of classification performance of the techniques given in [3] and the proposed ABC-ELM and IWO-ELM algorithms

Data sets	Methods	Testing (%)		Training time (s)	Neurons
		Accuracy	Dev		
Image segmentation	E-ELM	95.27	1.56	154.1	70
	ELM	95.07	1.14	0.1024	200
	DE-LM	88.65	2.05	13196.6	80
	LM	86.27	1.8	4745.7	100
	GALS	94.27	1.95	3423.7	80
	ABC(I)-ELM	95.36	1.25	168.558	170
	ABC(II)-ELM	95.33	1.23	167.637	170
	IWO(I)-ELM	95.64	1.28	172.608	150
	IWO(II)-ELM	95.25	1.26	378.3	170
Satellite image	E-ELM	88.46	1.36	1569.27	90
	ELM	88.99	1.01	1.2065	500
	DE-LM	85.31	2.14	36924	80
	LM	82.34	1.25	12561	100
	GALS	86.5	2.06	29209.4	80
	ABC(I)-ELM	89.01	1.02	1658.5	410
	ABC(II)-ELM	89.17	0.98	1499.3	400
	IWO(I)-ELM	89.10	0.99	1470.2	400
	IWO(II)-ELM	89.02	1.03	3523.4	410
Shuttle	E-ELM	99.53	0.12	1336.9	20
	ELM	99.51	0.1	2.8206	200
	DE-LM	99.33	0.09	10892.3	20
	LM	99.27	0.13	6132.2	50
	GALS	Out of memory			
	ABC(I)-ELM	99.54	0.08	6486.8	160
	ABC(II)-ELM	99.53	0.08	5799.5	150
	IWO(I)-ELM	99.57	0.08	5716.6	150
	IWO(II)-ELM	99.54	0.08	14321	160

Chapter 6

Comparative analysis of ELM and No-Prop algorithms

6.1 Introduction

Feedforward Neural networks have been successfully applied in many fields and applications. Out of many different types of neural networks, single hidden layer feedforward networks (SLFNs) have been studied more thoroughly during the past decades. The back-propagation (BP) algorithm is the most widely used method for training single hidden layer feedforward networks which adopts gradient descent-based methods to tune and optimize the weights in the network. Traditional gradient descent-based algorithms, such as back-propagation (BP), for SLFNs require all the weights and biases of the networks to be tuned iteratively. These algorithms usually get stuck in local minima and suffer from slow convergence. To overcome the limitations of conventional learning algorithm, Huang et al. [4][5] proposed the extreme learning machine (ELM) for training SLFNs. In ELM, the hidden layer parameters (input weights and biases) are randomly initialized and the output weights are analytically determined using Moore-Penrose (MP) generalized inverse [9]. Due to its universal approximation capability and classification capability, ELM has been extensively used in classification and regression applications [5] [6] [7]. Compared to traditional algorithms such as back-propagation, ELM not only provides good generalization performance, but also provides learning at extremely fast speed [5]. Recently, Widrow et al. [43] proposed a new learning algorithm called No-Propagation (No-Prop) for training feedforward Neu-

ral networks. In No-Prop algorithm, the weights and biases of the hidden layer neurons are randomly assigned and remain fixed during the learning process, and the output weights are trained using least mean square error (LMS) algorithm. The difference between ELM and No-Prop lies in the method used for training the output weights. ELM uses Moore-Penrose (MP) generalized inverse to compute the output weights analytically. On the other hand, No-Prop uses LMS gradient algorithm to optimize the output weights iteratively.

In this chapter, we have investigated and compared the performance of ELM and No-Prop algorithms on different benchmark classification data sets. The computational experiments demonstrate that the No-Prop algorithm is stable and provides good generalization performance, but it is slow. These experiments also show that ELM is fast, but in some cases it provides poor generalization if it is used without regularization.

The remaining part of this chapter is organized as follows: Section 6.2 presents a brief overview of ELM. Section 6.3 introduces the No-Prop algorithm. Experimental results and their analysis are presented in Section 6.4. Finally, Section 6.5 outlines some concluding remarks.

6.2 Regularized extreme learning machine

In Section 1.1, the output weights $\boldsymbol{\beta}$ of ELM network are calculated using Eq. (1.6) in which $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$. Several methods, such as orthogonal projection method and singular value decomposition (SVD), can be used to compute the Moore-Penrose generalized inverse of a matrix [7] [117]. The orthogonal projection method can be used in two ways: if $\mathbf{H}\mathbf{H}^T$ is nonsingular, then $\mathbf{H}^{\dagger} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$ or if $\mathbf{H}^T\mathbf{H}$ is nonsingular, then $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$.

A small value can be added to the diagonal of $\mathbf{H}\mathbf{H}^T$ or $\mathbf{H}^T\mathbf{H}$ to obtain a stable solution with better generalization performance according to ridge regression theory [7][117]. That is, $\mathbf{H}^{\dagger} = \mathbf{H}^T(\frac{I}{RG} + \mathbf{H}\mathbf{H}^T)^{-1}$ or $\mathbf{H}^{\dagger} = (\frac{I}{RG} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, where I is identity matrix and RG is the regularization parameter [118]. We refer to ELM with regularization factor RG as ELM(Reg).

6.3 NO-Prop algorithm

No-Prop algorithm is recently proposed learning algorithm by Widrow et al. [43] for training feedforward neural networks. No-Prop algorithm randomly initializes the weights and biases of the hidden layer neurons and computes the output weights iteratively using least mean square error (LMS) algorithm. Suppose we are given a single hidden layer feedforward neural networks (SLFNs) with L hidden neurons and activation function $g(x)$ as shown in Figure 1.1. The No-Prop algorithm initializes the input weights and hidden biases with random values and uses LMS algorithm to train the output weights. LMS algorithm, introduced by Widrow and Hoff [119] is the most widely used learning algorithm due to its computational simplicity. LMS algorithm is gradient descent-based method which adjusts the output weights of the network to minimize mean square error. With the presentation of each input vector during training, an instantaneous gradient is obtained of the mean square error with respect to each of the weights. The output weights are iteratively changed in the direction of the negative gradient which leads to an approximate minimum mean square error solution. In No-Prop algorithm, the output weights are updated as follows:

$$\mathbf{E}(k) = \mathbf{T} - \mathbf{H}_i(k)\boldsymbol{\beta}(k) \tag{6.1}$$

and

$$\boldsymbol{\beta}(k+1) = \boldsymbol{\beta}(k) + 2\eta\mathbf{E}(k)\mathbf{H}_i^T(k) \tag{6.2}$$

where η is the step-size parameter which controls the convergence rate and stability of the LMS algorithm and $\mathbf{E}(k)$ is the error between the desired output and the network output. The pseudo-code of No-Prop algorithm is given in Algorithm 15.

6.4 Experimental study

To evaluate the performance of the standard ELM, regularized ELM and No-Prop algorithms, we have used five standard benchmark data sets, viz. Balance, Dermatology, Iris, LIBRAS and Wine. The specifications of these data sets are given in Table 6.1.

Algorithm 15: No-Prop Algorithm

input : $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m, i = 1, \ldots, N\}$: data set
L: number of hidden neurons
$g(x)$: activation function

Initialize hidden layer parameters ($\mathbf{w}_i$, b_i), $i = 1, \ldots, L$ randomly;
Initialize the output weights $\boldsymbol{\beta}$;
Compute the hidden layer output matrix $\mathbf{H}$;
repeat
  for $i = 1$ *to* N **do**
    Compute the error $\mathbf{E}(k)$ of the current input $\mathbf{H}_i$ using Eq. (6.1);
    Update the output weights $\boldsymbol{\beta}(k+1)$ using Eq. (6.2);
  end
until *termination condition is satisfied*;

Table 6.1: Specifications of benchmark classification data sets

Data sets	Data		Features	Classes
	Training	Testing		
Balance	475	150	4	3
Dermatology	270	88	34	6
Iris	100	50	4	3
LIBRAS	270	90	90	15
Wine	100	78	13	3

6.4.1 Results and discussion

As mentioned already, the performance of the standard ELM, ELM with regularization parameter ELM(Reg) and No-Prop algorithms are evaluated on five benchmark data sets. We have used the sigmoid function for nonlinear mapping, and the input weights and hidden biases were generated from a uniform distribution over [-1, 1]. For No-Prop algorithm, the training will stop if either the maximum number of iterations ($iter_{max} = 10000$) is reached or the training error is less than a predetermined threshold. The training and testing sets are randomly generated from the whole data set at each run according to the specifications given in Table 6.1. All the simulations are

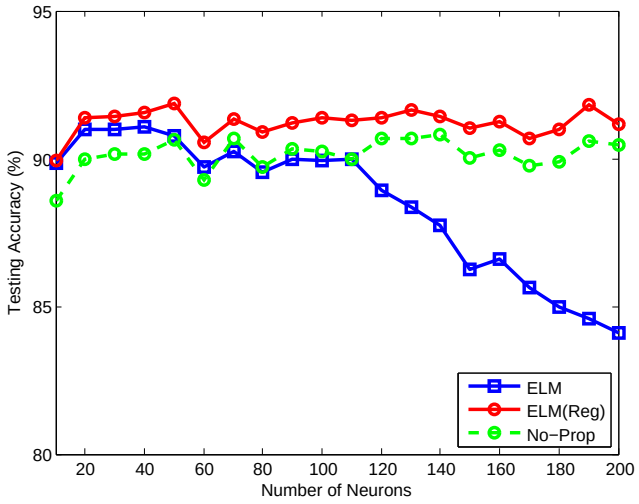

Figure 6.1: Classification performance on Balance with respect to different number of neurons

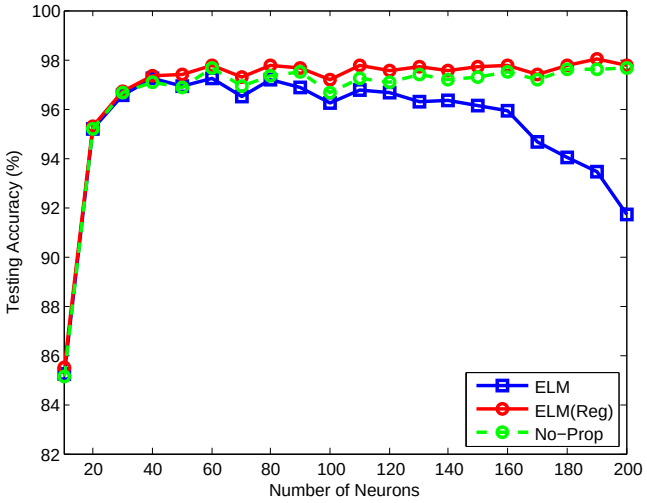

Figure 6.2: Classification performance on Dermatology with respect to different number of neurons

carried out 20 times independently and the average performance is reported. Figures 6.1, 6.2, 6.3, 6.4 and 6.5 show the classification performance of ELM, ELM(Reg) and No-Prop algorithms on different data sets with different number of hidden neurons $[10, 20, \ldots, 190, 200]$. For ELM(Reg), for each number of neurons the regularization value RG is chosen from the range $[2^{-24}, 2^{-23}, \ldots, 2^{23}, 2^{24}]$. Table 6.2 shows values of

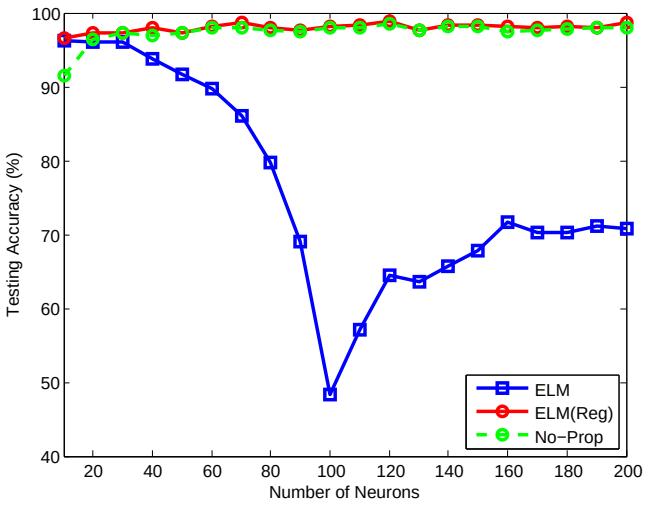

Figure 6.3: Classification performance on Iris with respect to different number of neurons

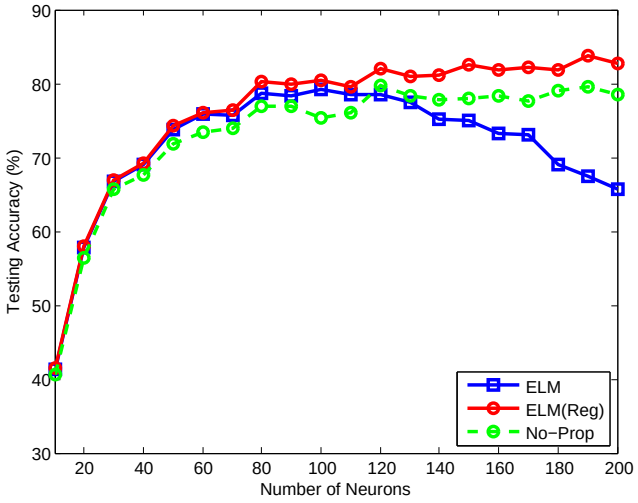

Figure 6.4: Classification performance on LIBRAS with respect to different number of neurons

the regularization parameter *RG* used in ELM(Reg) algorithm for different choices of number of hidden neurons. From Figures 6.1–6.5, we can observe that the standard ELM (i.e., without regularization parameter) is unstable and fails to generalize for different number of hidden neurons. The poor performance of standard ELM is attributed to the large condition number of the hidden layer correlation matrix which make the

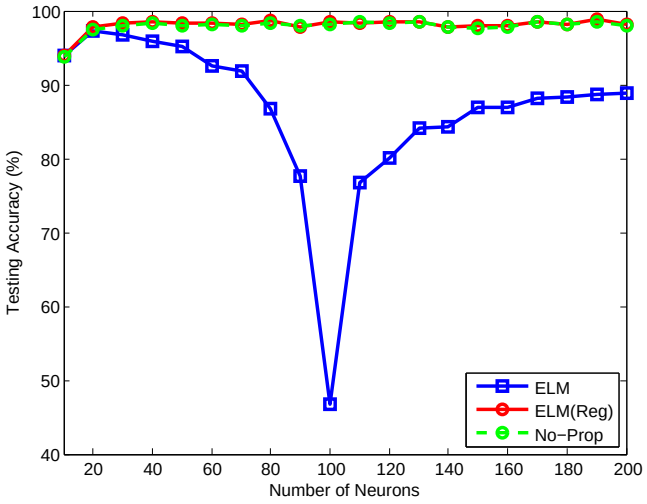

Figure 6.5: Classification performance on Wine with respect to different number of neurons

matrix inversion unstable or sensitive to the noise. The computational experiments show that No-Prop algorithm is stable and provides good generalization performance on all data sets, but it is slow. The classification performance of No-Prop algorithm increases with the increase in number of hidden neurons. ELM with regularization parameter outperformed No-Prop algorithm on all data sets, but there is a cost involved in searching for the best value of the regularization parameter.

6.5 Conclusion

This chapter provides a comparative analysis of the performance of ELM and No-Prop algorithms on different benchmark classification data sets. Both ELM and No-Prop algorithms are independently proposed for training feedforward neural networks. In both algorithms, the input weights and hidden biases are randomly initialized and remain unchanged during the learning process. While ELM determines the output weights analytically, No-Prop, on the other hand, uses LMS algorithm to optimize the output weights iteratively. In this chapter, we have evaluated and analysed the performance of ELM and No-Prop algorithms on different benchmark classification data sets. The simulations show that the No-Prop algorithm is stable and provides good generalization performance, but it suffers from slow convergence. The experiments

Table 6.2: The regularization parameter *RG* values used in ELM(Reg) algorithm

Number of Neurons	Balance	Dermatology	Iris	LIBRAS	Wine
10	2^{10}	2^{3}	2^{20}	2^{7}	2^{11}
20	2^{12}	2^{2}	2^{12}	2^{13}	2^{9}
30	2^{8}	2^{2}	2^{5}	2^{11}	2^{6}
40	2^{8}	2^{0}	2^{14}	2^{8}	2^{6}
50	2^{10}	2^{-1}	2^{7}	2^{10}	2^{4}
60	2^{9}	2^{3}	2^{8}	2^{13}	2^{4}
70	2^{9}	2^{-2}	2^{8}	2^{6}	2^{5}
80	2^{10}	2^{1}	2^{7}	2^{7}	2^{3}
90	2^{9}	2^{-1}	2^{5}	2^{8}	2^{1}
100	2^{9}	2^{-1}	2^{8}	2^{8}	2^{1}
110	2^{9}	2^{0}	2^{5}	2^{6}	2^{4}
120	2^{10}	2^{-2}	2^{6}	2^{6}	2^{-1}
130	2^{9}	2^{-2}	2^{7}	2^{4}	2^{4}
140	2^{13}	2^{0}	2^{7}	2^{6}	2^{0}
150	2^{9}	2^{-3}	2^{4}	2^{6}	2^{4}
160	2^{7}	2^{-2}	2^{10}	2^{6}	2^{4}
170	2^{9}	2^{3}	2^{6}	2^{6}	2^{0}
180	2^{13}	2^{-2}	2^{7}	2^{5}	2^{3}
190	2^{11}	2^{-4}	2^{3}	2^{5}	2^{2}
200	2^{8}	2^{-3}	2^{7}	2^{5}	2^{-1}

show that standard ELM has fast convergence, but it provides poor generalization performance if it is used without regularization. The experiments also show that ELM with regularization parameter has good generalization performance and outperformed No-Prop algorithm on all data sets.

.

Chapter 7

Conclusions and Directions for Future Research

7.1 Summary and Conclusions

For the last two decades, the research on neural networks with randomized hidden layer continues to attract attention of researchers from different domains and fields. Extreme learning machine is one of the randomized feedforward neural networks which gained more attention in the last decade, especially in the last few years and has shown its success and good performance in different domains of application. This thesis has provided extensive empirical study on the capability of ELMs for clustering and classification applications. There are three key ideas this thesis addresses.

The first key idea is concerned with the clustering in ELM feature space and it has been investigated in Chapter 2 and Chapter 3. Chapter 2 exploited the notion that performing a nonlinear transformation of nonlinearly separable input patterns into a higher dimensional feature space increases the possibility to separate these patterns linearly.

In Chapter 2, we have proposed a new approach (ELM K-means) which combines ELM method and K-means algorithm for unsupervised clustering. In this approach, the ELM method facilitates the projection of input data into ELM feature space and K-means algorithm performs clustering within this feature space. The new approach is tested on real world benchmark data sets and the simulations show that the projection of data into high dimensional ELM feature space enables K-means clustering algorithm

to achieve significantly better performance than when clustering was done in the original input space.

The proposed ELM K-means algorithm in Chapter 2 has obtained promising clustering performance but it has some limitations such as dependence on the initial cluster centers and convergence to local minima. To overcome such shortcomings, in Chapter 3 we have presented artificial bee colony (ABC) algorithm-based clustering approach for performing clustering in ELM feature space. The performance comparison of ELM K-means and ELM-ABC based clustering algorithms shows that ELM-ABC algorithm has achieved better performance than ELM K-means algorithm. In addition to comparison of clustering performance, we have compared the time-accuracy tradeoff of both algorithms and we found that ELM-ABC algorithm has a better time-accuracy tradeoff than ELM K-means algorithm. We have also conducted two-tailed t-test at 5% significance level for ELM-ABC against other algorithms. The t-test results show that the difference between the ELM-ABC and the other algorithms is statistically significant for most data sets which demonstrates the effectiveness of the proposed ELM-ABC algorithm.

The second key idea is focused on developing more principled approach for setting hidden neuron weights and hidden layer size selection either by tuning or by some sort of dimensionality reduction.

In Chapter 4, we have developed a new approach which combines ELM with random projection (RP) and investigated its application for low and high dimensional data classification and clustering problems. For high dimensional data, we have initialized the ELM hidden layer weights and biases using RP method where the size of ELM hidden layer is less than the dimension of the input data. This approach is evaluated on high dimensional data sets of different sizes for classification and clustering problems. The simulations suggested that the use of RP may help to shrink the dimension of the input data while preserving the distances between data points and reduces the computational complexity of performing classification and clustering. For low dimensional data, the ELM network is extended to contain two hidden layers. The parameters of the first hidden layer were assigned randomly whereas the second hidden layer weights and biases were initialized using RP method. The experiments on low dimensional data sets show that this approach obtains satisfying results and provides a tradeoff between generalization performance and computational complexity.

In Chapter 5, we have proposed two hybrid methods for optimizing the weights and biases of the ELM hidden layer in order to achieve good generalization performance with less number of neurons in the ELM hidden layer. In both approaches, the output weights are determined using Moore-Penrose (MP) generalized inverse. The first method ABC-ELM uses artificial bee colony (ABC) algorithm to tune the input weights and hidden biases whereas the second method IWO-ELM utilizes the invasive weed optimization (IWO) algorithm to optimize the hidden layer weights and biases. We also have proposed two procedures for generating neighboring solutions. The experiments on benchmark classification data sets show that the proposed ABC-ELM and IWO-ELM approaches were able to obtain good generalization performance.

The third key idea is concerned with the way output weights are trained. There are several ways to train the output layer of randomized SLFNs. In ELM, the output weights are optimized in batch mode using Moore-Penrose (MP) generalized inverse. In No-Prop algorithm, the output weights are trained iteratively using least mean square error (LMS) method.

In Chapter 6, we have applied ELM and No-Prop methods for classification problem and their performance has been analyzed. The simulations show that the standard ELM suffers from overfitting and fails to generalize whereas the regularized ELM was able to generalize well and outperformed No-Prop algorithm on all cases. The simulations also show that the No-Prop algorithm suffers from slow convergence but it provides good generalization performance.

7.2 Future Directions

This thesis has investigated the clustering in ELM feature space using K-means and artificial bee colony algorithms (see Chapter 2 and Chapter 3). However, there is a room for further improvement and investigation. For example, clustering methods which work in feature space obtained by Mercer kernel function based feature mapping such as RBF kernel can also work on feature space obtained by kernels inspired from extreme learning machine [120].

It is also possible to incorporate the ELM method into some other metaheuristic techniques such as genetic algorithm and particle swarm optimization, and compare the performance of the resulting methods with the ELM-ABC algorithm. There is

also scope for integrating ABC based clustering algorithm with some kernel methods such as RBF kernel or kernels inspired from extreme learning machine and compare the clustering performance of ABC algorithm in ELM and other kernel feature spaces. There is also possibility to improve the clustering performance of ELM-ABC algorithm by using local search method [103].

In this thesis, we have combined random projection (RP) with classification and clustering (see Chapter 4). We have already pointed out that random vector functional link (RVFL) network [46, 47] is another successful technique for SLFNs. The recent works on RVFL networks [46, 47] show that the direct links from the input layer to the output layer in RVFL have improved the performance of the SLFN with randomized input weights and hidden biases, suggesting that the direct connections regularize the effects of randomization. The strategy of projection of data using random projection (RP) can also be combined with RVFL networks for classification problem and investigating how direct links in RVLF can improve the classification performance.

There are several issues remaining in our work on tuning ELM using ABC and IWO algorithms (see Chapter 5) which provide scope for future studies such as developing an efficient way to reduce the training time required by ABC-ELM and IWO-ELM. There is also scope for developing additional neighborhood procedures in such a way that they can be used in a self-adaptive manner on the lines of SaE-ELM [2] and develop what can be called self-adaptive ABC-ELM and self-adaptive IWO-ELM.

While in this thesis we have worked with clustering and classification problems, regression is a problem that we have not addressed and ELM for regression is already proposed [7]. However, similar issues that we have raised in this thesis can also be taken up in the domain of regression in future.

References

[1] G.-B. Huang, X. Ding, and H. Zhou. **Optimization method based extreme learning machine for classification**. *Neurocomputing*, **74**:155–163, 2010. (viii, 2, 3, 10)

[2] J. Cao, Z. Lin, and G.-B. Huang. **Self-adaptive evolutionary extreme learning machine**. *Neural Processing Letters*, **36**:285–305, 2012. (xiii, 74, 83, 86, 99)

[3] Q.-Y. Zhu, A.K. Qin, P.N. Suganthan, and G.-B. Huang. **Evolutionary extreme learning machine**. *Pattern Recognition*, **38**:1759–1763, 2005. (xiii, 4, 10, 11, 73, 74, 83, 84, 87)

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. **Extreme learning machine: a new learning scheme of feedforward neural networks**. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, **2**, pages 985–990, Budapest, Hungary, 2004. (1, 3, 88)

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. **Extreme learning machine: theory and applications**. *Neurocomputing*, **70**:489–501, 2006. (1, 2, 3, 88)

[6] G.-B. Huang, L. Chen, and C.-K. Siew. **Universal approximation using incremental constructive feedforward networks with random hidden nodes**. *IEEE Transactions on Neural Networks*, **17**(4):879–892, 2006. (1, 3, 11, 73, 88)

[7] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. **Extreme learning machine for regression and multiclass classification**. *IEEE Transactions on*

Systems, Man, and Cybernetics B, Cybernetics, **42**(2):513–529, 2012. (1, 4, 61, 73, 88, 89, 99)

[8] G.-B. Huang. **Learning capability and storage capacity of two-hidden-layer feedforward networks**. *IEEE Transactions on Neural Networks*, **14**(2):274–281, 2003. (2)

[9] D. Serre. *Matrices: theory and applications*. Springer-VerlagInc, NewYork, 2002. (2, 88)

[10] Y. Lan, Y.C. Soh, and G.-B. Huang. **Constructive hidden nodes selection of extreme learning machine for regression**. *Neurocomputing*, **73**:3191–3199, 2010. (2, 10, 74)

[11] D.S. Broomhead and D. Lowe. **Multivariable functional interpolation and adaptive networks**. *Complex Systems*, **2**(3):321–355, 1988. (3)

[12] W.F. Schmidt, M.A. Kraaijveld, and R.P.W. Duin. **Feedforward neural networks with random weights**. In *Proceedings of the 11th IAPR international conference on pattern recognition methodology and systems*, pages 1–4, Hague, Netherlands, 1992. (3)

[13] Y.-H. Pao and Y. Takefuji. **Functional-link net computing: theory, system architecture, and functionalities**. *Computer*, **25**(5):76–79, 1992. (3)

[14] Y.-H. Pao, G.-H. Park, and D.J. Sobajic. **Learning and generalization characteristics of random vector functional-link net**. *Neurocomputing*, **6**:163–180, 1994. (3, 4)

[15] G.-B. Huang, Q.-Y. Zhu, K.Z. Mao, C.K. Siew, P. Saratchandran, and N. Sundararajan. **Can threshold networks be trained directly?** *IEEE Transactions on Circuits and Systems II: Express Briefs*, **53**(3):187–191, 2006. (3)

[16] G.-B. Huang, D.H. Wang, and Y. Lan. **Extreme learning machines: a survey**. *International Journal of Machine Learning and Cybernetics*, **2**(2):107–122, 2011. (3)

[17] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu. **Universal approximation of extreme learning machine with adaptive growth of hidden nodes**. *IEEE Transactions on Neural Networks and Learning Systems*, **23**(2):365–371, 2012. (3)

[18] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, and Y.C. Soh. **Dynamic extreme learning machine and its approximation capability**. *IEEE Transactions on Cybernetics*, **43**(6):2054–2065, 2013. (3)

[19] G.-B. Huang. **An insight into extreme learning machines: random neurons, random features and kernels**. *Cognitive Computation*, **6**(3):376–390, 2014. (3)

[20] G. Huang, G.-B. Huang, S. Song, and K. You. **Trends in extreme learning machines: a review**. *Neural Networks*, **61**:32–48, 2015. (3)

[21] Q. He, X. Jin, C. Du, F. Zhuang, and Z. Shi. **Clustering in extreme learning machine feature space**. *Neurocomputing*, **128**:88–95, 2014. (4, 11)

[22] A.K. Alshamiri, A. Singh, and B.R. Surampudi. **A novel ELM K-means algorithm for clustering**. In *Proceedings of the 5th International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO)*, **8947**, pages 212–222, Odisha, India, 2015. (4, 11)

[23] A.K. Alshamiri, A. Singh, and B.R. Surampudi. **Artificial bee colony algorithm for clustering: an extreme learning approach**. *Soft Computing*, `http://dx.doi.org/10.1007/s00500-015-1686-5`, 2015. (4)

[24] L.L.C. Kasun, H. Zhou, G.-B. Huang, and C.M. Vong. **Representational learning with ELMs for big data**. *IEEE intelligent systems*, **28**(6):31–34, 2013. (4)

[25] L. Feng, Y.-S. Ong, and M.-H. Lim. **ELM-guided memetic computation for vehicle routing**. *IEEE intelligent systems*, **28**(6):38–41, 2013. (4)

[26] B.-S. Oh, J. Jeon, K.-A. Toh, A.B.J. Teoh, and J. Kim. **A system for signature verification based on horizontal and vertical components in hand gestures**. *IEEE intelligent systems*, **28**(6):52–55, 2013. (4)

[27] A. Baradarani, Q.M.J. Wu, and M. Ahmadi. **An efficient illumination invariant face recognition framework via illumination enhancement and DD-DTCCWT filtering**. *Pattern Recognition*, **46**(1):57–72, 2013. (4)

[28] B. Lu, X. Duan, and Y. Yuan. **Facial expression recognition based on ensemble extreme learning machine with eye movements information**. In *Proceedings of ELM-2015*, **2**, pages 295–306, Hangzhou, China, 2015. (4)

[29] Z.-H. You, Y.-K. Lei, L. Zhu, J. Xia, and B. Wang. **Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis**. *BMC Bioinformatics*, **14**, 2013. (4)

[30] L. An and B. Bhanu. **Image super-resolution by extreme learning machine**. In *Proceedings of the 19th IEEE International Conference on Image Processing*, pages 2209–2212, Orlando, FL, USA, 2012. (4)

[31] S. Decherchi, P. Gastaldo, R.S. Dahiya, M. Valle, and R. Zunino. **Tactile-data classification of contact materials using computational intelligence**. *IEEE Transactions on Robotics*, **27**(3):635–639, 2011. (4)

[32] Q. Chen, J. Ding, J. Cai, and J. Zhao. **Rapid measurement of total acid content (TAC) in vinegar using near infrared spectroscopy based on efficient variables selection algorithm and nonlinear regression tools**. *Food Chemistry*, **135**(2):590–595, 2012. (4)

[33] X. Chen, Z.Y. Dong, K. Meng, Y. Xu, K.P. Wong, and H.W. Ngan. **Electricity price forecasting with extreme learning machine and bootstrapping**. *IEEE Transactions on Power Systems*, **27**(4):2055–2062, 2012. (4)

[34] M. Pal, A.E. Maxwell, and T.A. Warner. **Kernel-based extreme learning machine for remote-sensing image classification**. *Remote Sensing Letters*, **4**(9):853–862, 2013. (4)

[35] X. Lu, H. Zou, H. Zhou, L. Xie, and G.-B. Huang. **Robust extreme learning machine with its application to indoor positioning**. *IEEE Transactions on Cybernetics*, **46**(1):194–205, 2016. (4)

[36] L.P. Wang and C.R. Wan. **Comments on the extreme learning machine**. *IEEE Transactions on Neural Networks*, **19**(8):1494–1495, 2008. (4)

[37] G.-B. Huang. **Reply to Comments on the extreme learning machine**. *IEEE Transactions on Neural Networks*, **19**(8):1495–1496, 2008. (4)

[38] G.-B. Huang. **What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle**. *Cognitive Computation*, **7**(3):263–278, 2015. (4)

[39] P.J. Werbos. **Beyond regression: new tools for prediction and analysis in the behavioral sciences**, 1974. PhD thesis, Harvard University. (4)

[40] D.B. Parker. **Learning Logic**, 1982. Invention Report. S8l-64. File 1. Office of Technology Licensing, Stanford University. (4)

[41] Y. LeCun. **A learning scheme for asymmetric threshold networks**. In *Proceedings of Cognitiva*, **85**, pages 599–604, Paris, France, 1985. (4)

[42] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. **Learning representations by back-propagating errors**. *Nature*, **323**:533–536, 1986. (4)

[43] B. Widrow, A. Greenblatt, Y. Kim, and D. Park. **The No-Prop algorithm: a new learning algorithm for multilayer neural networks**. *Neural Networks*, **37**:182–188, 2013. (4, 12, 14, 88, 90)

[44] M.-H. Lim. **Comments on the No-Prop algorithm**. *Neural Networks*, **48**:59–60, 2013. (4)

[45] B. Widrow. **Reply to the Comments on the No-Prop algorithm**. *Neural Networks*, **48**:204, 2013. (4)

[46] L. Zhang and P.N. Suganthan. **A comprehensive evaluation of random vector functional link networks**. *Information Sciences*, `http://dx.doi.org/10.1016/j.ins.2015.09.025`, 2015. (4, 99)

[47] Y. Ren, P.N. Suganthan, N. Srikanth, and G. Amaratunga. **Random vector functional link network for short-term electricity load demand**

forecasting. *Information Sciences*, http://dx.doi.org/10.1016/j.ins.2015.11.039, 2015. (4, 99)

[48] W. Li, D. Wang, and T. Chai. **Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes**. *IEEE Transactions on Systems, Man, and Cybernetics, Systems*, **45**(2):303–314, 2015. (4)

[49] S. Scardapane, D. Wang, M. Panella, and A. Uncini. **Distributed learning for random vector functional link networks**. *Information Sciences*, **301**:271–284, 2015. (4)

[50] G. Beni and J. Wang. **Swarm intelligence in cellular robotic systems, proceed**, 1989. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy. (5)

[51] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York, 1999. (5)

[52] D. Karaboga. **An idea based on honey bee swarm for numerical optimization. Technical Report TR06**, 2005. Computer Engineering Department, Erciyes University, Turkey. (6, 74)

[53] B. Basturk and D. Karaboga. **An artificial bee colony (ABC) algorithm for numeric function optimization**. In *Swarm Intelligence Symposium 2006*, pages 12–14. IEEE, 2006. (6)

[54] D. Karaboga and B. Basturk. **A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm**. *Journal of Global Optimization*, **39**(3):459–471, 2007. (6)

[55] D. Karaboga and B. Basturk. **Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems**. In *IFSA 2007, Lecture notes in Artificial Intelligence*, **4529**, pages 789–798. Springer, 2007. (6)

[56] D. Karaboga and B. Basturk. **On the performance of artificial bee colony (ABC) algorithm**. *Applied Soft Computing*, **8**:687–697, 2008. (6)

[57] D. Karaboga and B. Akay. **A modified artificial bee colony (ABC) algorithm for constrained optimization problems**. *Applied Soft Computing*, **11**:3021–3031, 2011. (6)

[58] A. Singh. **An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem**. *Applied Soft Computing*, **9**:625–631, 2009. (6)

[59] W. Gao and S. Liu. **Improved artificial bee colony algorithm for global optimization**. *Information Processing Letters*, **111**:871–882, 2011. (6)

[60] W. Gao and S. Liu. **A modified artificial bee colony algorithm**. *Computers and Operations Research*, **39**:687–697, 2012. (6)

[61] A.R. Mehrabiana and C. Lucas. **A novel numerical optimization algorithm inspired from weed colonization**. *Ecological Informatics*, **1**:355–366, 2006. (9, 74)

[62] Y. Zhou, Q. Luo, H. Chen, A. He, and J. Wu. **A discrete invasive weed optimization algorithm for solving traveling salesman problem**. *Neurocomputing*, **151**:1227–1236, 2015. (9)

[63] R. Giri, A. Chowdhury, A. Ghosh, S. Das, A. Abraham, and V. Snasel. **A modified invasive weed optimization algorithm for training of feed-forward neural networks**. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3166–3173, Istanbul, Turkey, 2010. (9)

[64] X. Zhang, Y. Niu, G. Cui, and Y. Wang. **A modified invasive weed optimization with crossover operation**. In *Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA)*, pages 11–14, Jinan, China, 2010. (9)

[65] D. Kundu, K. Suresh, S. Ghosh, S. Das, B.K. Panigrahi, and S. Das. **Multi-objective optimization with artificial weed colonies**. *Information Sciences*, **181**(12):2441–2454, 2011. (9)

[66] A. Basak, D. Maity, and S. Das. **A differential invasive weed optimization algorithm for improved global numerical optimization**. *Applied Mathematics and Computation*, **219**(12):6645–6668, 2013. (9)

[67] S. Roy, Sk.M. Islam, S. Das, and S. Ghosh. **Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers**. *Applied Soft Computing*, **13**(1):27–46, 2013. (9)

[68] P. Ramezani, M. Ahangaran, and X.-S. Yang. **Constrained optimisation and robust function optimisation with EIWO**. *International Journal of Bio-Inspired Computation*, **5**(2):84–98, 2013. (9)

[69] X. Zhang, Y. Wang, G. Cui, Y. Niu, and J. Xu. **Application of a novel IWO to the design of encoding sequences for DNA computing**. *Computers & Mathematics with Applications*, **57**(11-12):2001–2008, 2009. (9)

[70] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran. **Online sequential fuzzy extreme learning machine for function approximation and classification problems**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **39**(4):1067–1072, 2009. (10)

[71] N. Liu and H. Wang. **Ensemble based extreme learning machine**. *IEEE Signal Processing Letters*, **17**(8):754–757, 2010. (10)

[72] H.-J. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu. **A fast pruned-extreme learning machine for classification problem**. *Neurocomputing*, **72**:359–366, 2008. (10, 11, 73)

[73] G. Huang, S. Song, J.N.D. Gupta, and C. Wu. **Semi-supervised and unsupervised extreme learning machines**. *IEEE Transactions on Cybernetics*, **44**(12):2405–2417, 2014. (11)

[74] L.L.C. Kasun, T. Liu, Y. Yang, Z. Lin, and G.-B. Huang. **Extreme learning machine for clustering**. In *Proceedings of ELM-2014*, **1**, pages 435–444, Hangzhou, China, 2014. (11)

[75] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. **OP-ELM: Optimally pruned extreme learning machine**. *IEEE Transactions on Neural Networks*, **21**(1):158–162, 2010. (11, 73)

[76] G.-B. Huang and L. Chen. **Convex incremental extreme learning machine**. *Neurocomputing*, **70**:3056–3062, 2007. (11)

[77] G.-B. Huang and L. Chen. **Enhanced random search based incremental extreme learning machine**. *Neurocomputing*, **71**:3460–3468, 2008. (11)

[78] J. Han and M. Kamber. *Data mining: concepts and techniques*. Academic Press, 2001. (16)

[79] A.K. Jain, M.N. Murty, and P.J. Flynn. **Data clustering: a review**. *ACM Computing Surveys*, **31**(3):264–323, 1999. (16)

[80] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. **A survey of kernel and spectral methods for clustering**. *Pattern Recognition*, **41**:176–190, 2008. (16, 19)

[81] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. PEnglewood Cliffs, NJ: Prentice-Hall, 1989. (17)

[82] R. Xu and II D. Wunsch. **Survey of clustering algorithms**. *IEEE Transactions on Neural Networks*, **16**(3):645–678, 2005. (17)

[83] M.K. Ng. **A note on constrained K-means algorithms**. *Pattern Recognition*, **33**:515–519, 2000. (17, 27)

[84] L. Zhang and Q. Cao. **A novel ant-based clustering algorithm using the kernel method**. *Information Sciences*, **181**:4658–4672, 2011. (17, 19, 22, 27)

[85] M. Girolami. **Mercer kernel based clustering in feature space**. *IEEE Transactions on Neural Networks*, **13**(3):780–784, 2002. (17, 18, 19, 31)

[86] B. Scholkopf, A. Smola, and K.R. Muller. **Nonlinear component analysis as a kernel eigenvalue problem**. *Neural Computation*, **10**(5):1299–1319, 1998. (18)

[87] G.F. Tzortzis and A.C. Likas. **The global kernel K-means algorithm for clustering in feature space**. *IEEE Transactions on Neural Networksr*, **20**(7):1181–1194, 2009. (18)

[88] R. Zhang and A.I. Rudnicky. **A large scale clustering scheme for kernel K-means**. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR), Quebec, Canada*, **4**, pages 289–292, 2002. (18)

[89] R. Chitta, R. Jin, T.C. Havens, and A.K. Jain. **Approximate kernel K-means: solution to large scale kernel clustering**. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD), New York, USA*, pages 895–903, 2011. (18)

[90] F. Camastra and A. Verri. **A novel kernel method for clustering**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(5):801–805, 2005. (18)

[91] A.K. Jain. **Data clustering: 50 years beyond K-means**. *Pattern Recognition*, **31**:651–666, 2010. (20)

[92] R.A. Fisher. **The use of multiple measurements in taxonomic problems**. *Annals of Eugenics*, **7**:179–188, 1936. (24, 82)

[93] K. Krishna and M.N. Murty. **Genetic K-means algorithm**. *IEEE Transactions on Systems, Man, and Cybernetics B, Cybernetics*, **29**(3):433–439, 1999. (27)

[94] S.Z. Selim and K. Al-sultan. **A simulated annealing algorithm for the clustering problems**. *Pattern Recognition*, **24**(10):1003–1008, 1991. (27)

[95] D.W. van der Merwe and A.P. Engelhrecht. **Data clustering using particle swarm optimization**. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 03)*, pages 215–220, Canbella, Australia, 2003. (27)

[96] P.S. Shelokar, V.K. Jayaraman, and B.D. Kulkarni. **An ant colony approach for clustering**. *Analytica Chimica Acta*, **509**:187–195, 2004. (27)

[97] D. Karaboga and C. Ozturk. **A novel clustering approach: artificial bee colony (ABC) algorithm**. *Applied Soft Computing*, **11**:652–657, 2010. (27, 30)

[98] C. Zhang, D. Ouyang, and J. Ning. **An artificial bee colony approach for clustering**. *Expert Systems with Applications*, **37**:4761–4767, 2010. (27)

[99] X. Yan, Y. Zhu, W. Zou, and L. Wang. **A new approach for data clustering using hybrid artificial bee colony algorithm**. *Neurocomputing*, **97**:241–250, 2012. (28)

[100] E. Falkenauer. *Genetic algorithm and grouping problems*. Wiley, New York, 1998. (28)

[101] S. Sundar and A. Singh. **A swarm intelligence approach to the quadratic multiple Knapsack problem**. In *Proceedings of the 17th International Conference on Neural Information Processing (ICONIP 2010), Lecture Notes in Computer Science*, **6443**, pages 626–633, 2010. (28)

[102] S. Sundar and A. Singh. **Metaheuristic approaches for the blockmodel problem**. *IEEE Systems Journal*, **9**(4):1237–1247, 2014. (28)

[103] V. Pandiri and A. Singh. **Two metaheuristic approaches for the multiple traveling salesperson problem**. *Applied Soft Computing*, **26**:74–89, 2015. (28, 77, 99)

[104] S.N. Chaurasia and A. Singh. **A hybrid swarm intelligence approach to the registration area planning problem**. *Information Sciences*, **302**:50–69, 2015. (28)

[105] D. Fradkin and D. Madigan. **Experiments with random projections for machine learning**. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, 2003. (44)

[106] S. Deegalla and H. Bostrom. **Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification**. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA)*, pages 245–250, Orlando, Fl, 2006. (45)

[107] X.Z. Fern and C.E. Brodley. **Random projection for high dimensional data clustering: A cluster ensemble approach**. In *Proceedings of the 20th International Conference of Machine Learning (ICML)*, pages 186–193, Washington DC, 2003. (45)

[108] A. Cardoso and A. Wichert. **Iterative random projections for high dimensional data clustering**. *Pattern Recognition Letters*, **33**:1749–1755, 2012. (45)

[109] P. Gastaldo, R. Zunino, E. Cambria, and S. Decherchi. **Combining ELM with random projections**. *IEEE Intelligent Systems*, **28**(6):18–20, 2013. (45, 47)

[110] W.B. Johnson and J. Lindenstrauss. **Extensions of Lipschitz mappings into a Hilbert space**. In *Proceedings of the Conference in modern analysis and probability*, pages 189–206, 1984. (46)

[111] D. Achlioptas. **Database-friendly random projectionse**. In *Proceedings of the 20th Annual Symposium on the Principles of Database Systems*, pages 274–281, 2001. (46)

[112] R. Baraniuk, M. Davenport, R. DeVore, and M.B. Wakin. **A simple proof of the restricted isometry property for random matrices**. *Constructive Approximation*, **28**:253–263, 2008. (46)

[113] C. Bouveyron, S. Girard, and C. Schmid. **High dimensional data clustering**. *Computational Statistics and Data Analysis*, **52**:502–519, 2007. (47)

[114] I. Assent. **Clustering high dimensional data.** *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **2**(4):340–350, 2012. (47, 50)

[115] B. Akay and D. Karaboga. **A modified artificial bee colony algorithm for real-parameter optimization**. *Information Sciences*, **192**:120–142, 2012. (74)

[116] A. Basak, S. Pal, S. Das, A. Abraham, and V. Snasel. **A modified invasive weed optimization algorithm for time-modulated linear antenna array synthesis**. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, Barcelona, 2010. (74)

[117] C.R. Rao and S.K. Mitra. *Generalized inverse of matrices and its applications*. Wiley, NewYork, 1971. (89)

[118] W. Deng, Q. Zheng, and L. Chen. **Regularized extreme learning machine**. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, pages 389–395, 2009. (89)

[119] B. Widrow and M.E. Hoff. **Adaptive switching circuits**. *IRE WESCON Convention Record*, **4**:96–104, 1960. (90)

[120] B. Frenay and M. Verleysen. **Parameter-insensitive kernel in extreme learning for non-linear support vector regression**. *Neurocomputing*, **74**:2526–2531, 2011. (98)

List of Publications

[1] A.K. Alshamiri, B.R. Surampudi and A. Singh. **A novel ELM K-means algorithm for clustering**. *In Proceedings of the 5th International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO)*, LNCS, **8947**, pages 212-222, Springer, 2015.

[2] A.K. Alshamiri, A. Singh and B.R. Surampudi. **Artificial bee colony algorithm for clustering: an extreme learning approach**. *Soft Computing*, `http://dx.doi.org/10.1007/s00500-015-1686-5`, 2015.

[3] A.K. Alshamiri, A. Singh and B.R. Surampudi. **Combining ELM with random projections for low and high dimensional data classification and clustering**. *In Proceedings of the 5th International Conference on Fuzzy and Neural Computing (FANCCO)*, **415**, AISC, pages 89-107, Springer, 2015.

[4] A.K. Alshamiri, A. Singh and B.R. Surampudi. **Two swarm intelligence approaches for tuning extreme learning machine**. Communicated to *the International Journal of Machine Learning and Cybernetics*.

[5] A.K. Alshamiri, A. Singh and B.R. Surampudi. **Comparative analysis of ELM and No-Prop algorithms**. Communicated to *the 9th International Conference on Contemporary Computing (IC3)*.