Evolutionary parallel extreme learning machines for the data classification problem

Tansel Dokeroglu[1]*, Ender Sevinc[2]

[1] TED University, Computer Engineering Department, Ankara, TURKEY
[2] University of Turkish Aeronautical Association, Computer Engineering Department, Ankara, TURKEY

Abstract

This study proposes an Island Parallel Evolutionary Extreme Learning Machine algorithm (IPE-ELM) for the well-known data classification problem. The ELM is a fast and efficient machine learning technique with its single-hidden layer feed-forward neural network (SLFN). High prediction accuracy and learning speed of the ELM make it an elegant tool for the fitness calculation process of the evolutionary algorithms. The IPE-ELM algorithm combines the evolutionary genetic algorithms (for feature selection), ELM machine learning technique (for prediction accuracy calculation), parallel computation (for faster fitness evaluation), and parameter tuning (activation function selection and the number of hidden neurons) for the solution of this important problem. Each ELM that runs at a different processor selects one of four different activation functions (*Sine*, *Cosine*, *Sigmoid* and *Hyperbolic Tangent*) and uses a randomized number of hidden neurons to achieve higher prediction accuracy. The proposed algorithm provides high quality results with its (near)-linear scalability behavior. The IPE-ELM algorithm is compared with state-of-the-art data classification algorithms by using UCI benchmark datasets and significant improvements are reported in terms of prediction accuracy with reasonable execution times. The scalable IPE-ELM algorithm can be reported as the first island parallel evolutionary classification algorithm with its high prediction accuracy results that outperforms state-of-the-art algorithms in literature.

Keywords

Extreme Learning Machine; Data classification; Feature selection; Distributed evolutionary algorithm.

1 Introduction

Data classification is a crucial mining technique with its many applications in our daily life [1]. Scientists can identify, acquire knowledge and derive statistical/predictive models by making use of data classification techniques. The accuracy and the execution speed are important issues of the data classification process. One of the best means of extracting interesting and valuable patterns is making use of recent machine learning techniques. Dealing with large datasets (having many features and rows) requires advanced supervised machine learning techniques [2][3]. Supervised machine learning techniques build a model to predict the class labels of data by using a set of training data. Recently, it is very common to have a large amount of data with several attributes/features and it is not an easy process to properly clean and classify such a large amount of data correctly and obtain the distilled information. In literature, there have been many supervised techniques proposed for the solution of the data classification problem. However, the demand for fast algorithms that work with high prediction accuracy is still a valuable research area.

A high quality classifier is a crucial part of a data classification process. The classifier should have a good prediction accuracy and a good generalization ability. The training speed of a classifier is another important point that should be considered. Extreme Learning Machine (ELM) is a recent and fast supervised machine learning technique with its high performance for the data classification problem [4]. The learning speed of feed-forward neural networks is generally slow and it has been a major drawback in machine learning applications. Slow gradient-based learning algorithms are used to train neural networks and the parameters of the networks are tuned by using such learning techniques. These are the main reasons of the slow learning process. However, the ELM is a different machine learning technique for single-hidden layer feed-forward neural networks (SLFNs) that randomly chooses the number of hidden nodes and determines the output weights of SLFNs [5]. This property of the ELM makes it a suitable technique for intensive fitness chromosome evaluation of evolutionary genetic algorithms that select the best feature subset. The ELM has been applied to a lot of important problems and many studies are still under progress for improving the performance of this valuable machine learning technique [6].

With recent developments in computer science, the need for real-time processing of large datasets presents big challenges to traditional ways of data processing [7]. For this reason, Feature Subset Selection (FSS) has attracted the attention of scientists to filter out unnecessary data and greatly reduce processing time [8]. Ensemble-based wrapper methods (they use an exploration method for efficient FSS and use a machine learning method to measure the accuracy level) applied with FSS are providing good results for the data classification problem [9]. The wrapper

methods are computationally expensive tools since they need to compute many fitness values for the explored subsets. However, they are still the best performing methods.

We use a parallel computation environment to select the features of a dataset by using a genetic algorithm and apply ELM to the selected features to evaluate the prediction accuracy through fast evaluation of the instances. Genetic and ELM machine learning have been used before for solving the data classification problem. However, it is the first application of these methods with a parallel island genetic algorithm approach. We tune the parameters of the ELM dynamically and experimentally show that our method outperforms state-of-the-art metaheuristics. There have been earlier works that try to parallelize the ELM. Our method differs from the fact that we don't parallelize the matrix multiplication phase of ELM, which is a trivial method that enables faster execution using parallel processing. This process requires intensive communication between processors and not scalable. Instead, we propose an island parallel method and execute as many ELM as the number of processors in the environment simultaneously. This approach provides an effective diversification mechanism for improving the population quality of the genetic algorithm.

Considering all the issues mentioned above, we propose a novel evolutionary island parallel ELM-based classifier for the data classification problem. To the best of our knowledge, the IPE-ELM algorithm is the first island parallel machine learning algorithm in literature that has been applied to the data classification problem [10]. The IPE-ELM generates diversified populations at each processor's memory and improves the population's fitness qualities independently. Our approach provides a very effective diversification mechanism for increasing the population quality of the genetic algorithm by initializing random number generator of each processor with a different seed. At the termination phase of the processes at each processor, the best results of the slave nodes are collected by the master node and the overall best solution is reported.

Parallel machine learning is a new developing research area and designing scalable parallel algorithms in this field is challenging. In our opinion, the IPE-ELM algorithm is a unique algorithm with its efficient features when compared with other algorithms in this domain. Four different activation functions are used during the classification process, namely, *Sine, Cosine, Sigmoid* and *Hyperbolic Tangent*. It is not always possible to choose a single type of activation function to be the best one for every possible dataset. Different activation functions can do better on varied datasets. Each processor randomly selects one of these activation functions and continues its optimization process. The number of hidden neurons is another criterion to be considered during the classification. The tuning of this parameter greatly affects performance. This issue is also observed and the results obtained in our experiments are reported. Therefore, each processor decides a different number of hidden neurons (in the range of [2-10] % of the instance size) and runs the ELM. The size of the population has been chosen as 70 after comprehensive tests. The best performing convergence, truncation, crossover and mutation ratios are applied in all processors of the parallel distributed memory environment by using Message Passing Interface (MPI) libraries. The parallel and diversified populations of the IPE-ELM algorithm provide a stagnation prevention mechanism. At each processor, we select different seeds for the randomization of all parameters in the FSS and the ELM phases of the algorithm, which prevents the genetic algorithm from exploring the same areas of the search space repeatedly. Comprehensive experiments comparing our scalable algorithm with state-of-the-art classification algorithms show that the IPE-ELM algorithm outperforms them in terms of prediction accuracy values with reasonable execution times.

Some of the state-of-the-art metaheuritics that have been applied to the data classification problem are Particle Swarm Optimization (PSO) [3], Attribute Bagging (AB), (a technique for improving the accuracy and stability of classifier ensembles induced using random subsets of features) [11], Multi-View Adaboost (MVA) [12], Random Subspace Method for constructing decision forests (RSE) [13], Correlation based Feature Selection (CFS-SFS) [14], C4.5 [15], Hybrid Genetic Algorithm and ELM-based feature selection algorithm (HGEFS) [9], Advanced Binary Ant Colony Optimization (ABACO) [16], and ACO-based feature selection algorithm (ACOFS) [17]. The algorithms mentioned here need to calculate the fitness of each next possible better solution, which is the most time-consuming part of these algorithms. In our experiments, we compare our solutions with the results of these algorithms.

In section 2, related studies for the state-of-the-art ELM techniques and data classification algorithms are given. The details of the ELM are presented in section 3. The proposed IPE-ELM algorithm is introduced in section 4. The setup of the experimental environment, obtained results of the experiments, and comparison with state-of-the-art methods are reported in section 5. Concluding remarks are provided in the last section.

2 Related work

In this section, we give information about the ELM, FSS, state-of-the-art evolutionary data classification techniques and parallel implementations of the ELM. Huang et al. introduced the ELM in 2004 [4]. Huang et al. propose the ELM for the classification of the standard optimization method and enhances the ELM to a SLFNs support vector network [18]. Huang et al. [19] show that least square SVM (LS-SVM) and proximal SVM (PSVM) can be simplified and a unified learning framework of LS-SVM, PSVM, and other algorithms of regularization with ELM can be built.

Dash et al. [20] provide a comprehensive survey on FSS. Another comprehensive guideline on alternative FSS approaches is presented by Xue et al. [9]. They present a survey of the state-of-the-art work on evolutionary computation for FSS, which identifies the contributions of the present algorithms. Yang & Honavar [21] offer an attractive approach to find near-optimal solutions. They present an approach to FSS using a genetic algorithm. Xue et al. [9] propose a novel hybrid genetic algorithm and ELM based FSS algorithm (HGEFS). The experiments show that HGEFS outperforms other algorithms in literature. Deniz et al. [22] propose a multi-objective genetic algorithm combined with supervised machine learning techniques for the FSS in the binary classification problem. The performance of their algorithm is compared with state-of-the-art algorithms, Greedy Search, Particle Swarm Optimization, Tabu Search, and Scatter Search. The proposed algorithm is robust and it performs better than the existing methods on most of the UCI datasets. Unler & Murat [3] investigate the FSS problem for the binary classification using logistic regression model. They develop a discrete particle swarm optimization (PSO) algorithm for the FSS problem. Their experiments report that this new discrete PSO algorithm is efficient in terms of both computational performance and classification accuracy . The ELM may need higher number of hidden neurons because of its random determination of the input weights and hidden biases. Zhu et al. [23] propose a new hybrid differential evolutionary algorithm to select the input weights and MoorePenrose (MP) generalized inverse to decide the output weights. Their results show that this new approach can have a good generalization performance with compact networks. In a comprehensive review by Huang et al. [24], the current state of the theoretical research and practical advances on ELM are reported. They give an overview of the ELM as theoretically from the perspective of universal approximation capability, generalization ability, and the interpolation theory. They explain various improvements of ELM that improve the stability, sparsity and accuracy under general or specific conditions. Kiziloz et al. [25] consider the minimum number of features as a multi-objective optimization problem while not compromising the accuracy of the results in FSS. They develop a set of novel multiobjective TLBO algorithms combined with supervised machine learning techniques for the solution of FSS in Binary Classification Problems.

Alexandre et al. [26] solve the problem of selecting sound-description features for improving a vehicle classifier work better. The purpose of the study is to measure the feasibility of a novel FSS method based on a special class of a hybrid evolutionary algorithm based ELM. The method helps the ELM classifier to improve its performance from a mean probability of correct classification of 74.83% (when no feature is eliminated) up to 93.74% (with a subset of selected features). Garcia-Nieto et al. [27] perform the selection of genes to analyze the sensitivity and the specificity, used as quality indicators of the classification tests. The classification is performed by SVMs. The results show that their method is suitable to solve this problem. Bryll et al. [11] present a new technique to improve the prediction accuracy of classifiers by using random subsets of features. The AB is a wrapper technique that can be combined with learning algorithms. It finds a good feature subset size and selects subsets of features randomly. Ho [13] proposes a new method to build a decision tree based classifier that keeps the highest accuracy on training data and improves the prediction accuracy as it grows in complexity. Xu & Papageorgiou [28] propose a mixed integer linear programming (MILP) model for the multi-class data classification problem by using a hyper-box representation. Kartal et al. [29] develop a hybrid method that combines machine learning techniques with multicriteria decision-making techniques to manage multi-attribute inventory analysis. Ou et al. [30] propose a method based on ELM to predict the major raw material price in steel plants and focus on the integration of Grey Relation Analysis (GRA) with a hybrid forecasting model to predict the cost of iron ore and coking coal.

Sun et al. [31] propose an OS-ELM based ensemble classification framework for distributed classification in a hierarchical P2P network. They apply the incremental learning principle to produce an ensemble classifier. The results of the experiments show the efficiency of the proposed algorithms. Li et al. [32] propose an algorithm (TL-ELM) based on the ELM. The algorithm uses a small amount of target domain tag data and a large number of source domain old data to build a high-quality classification model. The algorithm inherits the best sides of ELM and makes up for the defects that traditional ELM cannot transfer knowledge. Xin et al. [33] propose a novel distributed ELM based on MapReduce framework. The proposed algorithm covers the shortage of traditional ELM whose learning ability is weak to huge dataset.

The parallel implementations of the ELM are as follows: Heeswijk et al. [34] present an approach for performing regression on large datasets rapidly, using an ensemble of ELMs. They search for how the evaluation of ELMs can be improved. The experiments show that the performance is good on the regression tasks, and the GPU-accelerated ELM gains speedups over using a core. The approach can be applied with ELMs. He et al. [35] propose a parallel

ELM on a MapReduce framework for regression. The results show that the parallel ELM for regression can work well on very large datasets with commodity hardware. Lin et al. [36] present a practical ELM in cloud computing to cut the training time. The cloud environment calculates the Moore-Penrose generalized inverse, the heaviest computation load of the process. He et al. [37] propose a novel algorithm for face recognition with ELM and sparse coding. The common feature hypothesis is used to extract the basis function from the local universal images and the SLFN simulates the sparse coding process for face images with the ELM.

Parallel genetic algorithms are easy to develop and promise significant performance improvements for numerous optimization problems either with single or multi-objective goals. Many NP-Hard problems have been solved by using parallel genetic algorithms efficiently [38]. In a recent study by Tosun et al., [39] well-known quadratic assignment problem instances are solved successfully. Dokeroglu & Cosar [40] solve the one-dimensional bin packing problem by using island parallel genetic algorithms. Kucukyilmaz & Kiziloz [41] propose a novel parallel grouping genetic algorithm for the one dimensional bin packing problem. To the best of our knowledge, there is no island parallel evolutionary machine learning algorithm in literature like our IPE-ELM algorithm that is proposed for the data classification problem.

3 Extreme Learning Machines

In this section, we give information about the ELM used by the IPE-ELM algorithm [4], [5]. The ELM uses an SLFN with a learning speed faster than traditional feed-forward network learning algorithms (e.g. back-propagation (BP)) (see Figure 1 for SLFN). Due to its simplicity, remarkable efficiency, and impressive performance on generalization, the ELM has been applied in a variety of domains, such as computer vision, bioinformatics, data classification, system identification, and control and robotics [24].

Figure 1: Single-hidden layer feed-forward network.

The output of SLFN having L number of hidden nodes can be represented with the Equation 1;

$$f_L(x) = \sum_{i=1}^{L} \beta_i . G(a_i, b_i, x) \qquad x \epsilon \mathbf{R}^n, a_i, b_i \epsilon \mathbf{R} \tag{1}$$

where a_i and b_i are the learning parameters of hidden nodes β_i the weight connecting the i^{th} hidden node to the output node. $G(a_i, b_i, x)$ is the output of the hidden node with respect to the input x. In general, the additive hidden node with activation function is $g(x) : R \rightarrow R$. At time, $G(a_i, b_i, x)$ is given by;

$$G(a_i, b_i, x_j) = g(a_i . x_j + b_i) \qquad b_i \epsilon \mathbf{R}, j = 1, ..., N \tag{2}$$

Equation 3 can be written as;

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

where,

$$\mathbf{H}(a_1, ..., a_L, b_1, ..., b_L, \mathbf{x}_1, ..., \mathbf{x}_N) = \begin{bmatrix} g(a_1 . \mathbf{x}_1 + b_1) & \cdots & g(a_L . \mathbf{x}_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(a_1 . \mathbf{x}_N + b_1) & \cdots & g(a_L . \mathbf{x}_N + b_L) \end{bmatrix}_{NxL} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{Lxm} \quad and \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{Nxm} \tag{5}$$

H is called the hidden layer output matrix of the SLFN; the i^{-th} column of H is the j^{th} hidden node output with respect to the input $x_1, x_2, ..., x_N$. $h(x) = G(a_1, b_1, x), ..., g(a_L, b_L, x)$ is called the hidden layer feature mapping. The i^{th} row of H is the hidden layer feature mapping with respect to the i^{th} input $x_i : h(x_i)$. It has been proved that from the interpolation capability point of view, if the activation function g is infinitely differentiable in any interval the hidden layer parameters can be randomly generated.

Activation functions: The ELM finds a solution on a unified learning framework for SLFNs. Activation function, also called as Transfer Function, defines the output of a node due to a given input or set of inputs. In other words, activation functions are used to restrict and limit the output value to a certain finite value range. In terms of this

approximation, activation function has an important role. Here, we examine the efficiencies of activation functions and perform experiments. In our algorithm, we use four different activation functions. Each processor in the parallel computation environment uses/selects one of these activation functions randomly during the optimization process. These activation functions used by the IPE-ELM algorithm are:

Sigmoid function: is a mathematical function having a characteristic *"S"-shaped* curve or sigmoid curve. In this context, sigmoidal function refers to the special case of the logistic function, defined by the formula:

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

where n is the weighted sum of the inputs. Its range is between zero and one. It is easy to understand and apply but it has major problems. First, it has *vanishing gradient problem*, which means in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. Secondly, its output is not zero centered. It could make the gradient updates go too far in different directions.

Hyperbolic Tangent function: Its mathematical formula is:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{7}$$

Its output is zero centered because its range is between -1 to 1, i.e. $-1 < output < 1$. Hence the optimization is easier in this method in practice. It is sometimes preferred over *Sigmoid* function. But it also suffers from *vanishing gradient problem*.

Sine function: Though most activation functions used in SLFN or deep neural networks are non-periodic, we can also use periodic functions such as *sine* and *cosine*.

$$sin(x) = -1 \leq output \leq 1 \tag{8}$$

In case of a neural network system with *sine* activation, the entire solution will be repeated periodically and the system will be trained for similar output classes. A neural network with one hidden layer can approximate any function, given the activation function is increasing and is finite (with a min and a max) where The *sine* function is not an increasing function and an input to *sine* function that is very low and very high might produce the same output.

Cosine function: is used for a comparison with *sine* function though it is not a commonly used activation function. Its results are observed due to its periodic nature w.r.t. *sine* function.

$$cos(x) = -1 \leq output \leq 1 \tag{9}$$

4 Island Parallel Evolutionary Extreme Learning Machine Algorithm (IPE-ELM)

In this section, we introduce our proposed island parallel evolutionary algorithm, IPE-ELM. The main goal of the algorithm is to discover the best subset of features that will produce the highest prediction accuracy for the data classification problem. The IPE-ELM algorithm has two main components (phases), evolutionary computation (for selecting feature subsets) and the ELM (for finding the prediction accuracy of the selected features).

Island parallel genetic algorithms are novel implementations of classical evolutionary genetic algorithms on parallel/distributed computation environments. In classical approach there exits a single population on the memory of the processor and the quality of the population is improved by using crossover and mutation operators through generations [38]. Getting stuck into local optima is a common problem in this approach. However, island parallel version of the genetic algorithms provide many diverse populations on the memory of processors and optimize the solution of the problem. A communication can be provided between the populations in the environment or not. It depends on the implementation approach of the developers. In previous studies, it is observed that island parallel algorithms provide better results than their single-core versions [41]. Our proposed algorithm IPE-ELM is a typical island parallel genetic algorithm that is applied to the solution of this problem for the first time.

The chromosome structure of the IPE-ELM algorithm consists of genes that represent the attributes/features of a given dataset. The values of the genes are binary (either one or zero). Value one means that the gene is selected, whereas zero means that the corresponding feature is eliminated. In Figure 2, chromosomes in the crossover process are given in binary. The crossover operation happens between two selected chromosomes and produces two new children chromosomes. The crossover operator of the IPE-ELM algorithm prevents the searching process

from being stagnated at local optima by continuously diversifying the individuals in the population. The proposed mutation operator works on a single chromosome that is randomly selected from the population and one of its genes is randomly selected to be mutated. If the value of selected gene is one, it is changed to zero or vice versa. Figure 3 presents how the mutation operator works on a chromosome.

Figure 2: Crossover operator that works on two chromosomes. A new child is generated from the left part of first chromosome and right part of the second chromosome.

Figure 3: Mutation operator changes the value of the fourth gene to generate a new one.

Each processor (slave node) in the parallel computation environment has its own diverse population that is initialized/generated randomly with respect to the rank of the processor. Thus, the population produced in each processor becomes different from the other populations, resulting in very efficient use of the parallel processing power. The random number generator of the IPE-ELM algorithm depends on the rank identifier of the processor defined by MPI. This property provides distinct random number sequences and used to prevent populations to get stuck into local optima.

The number of hidden neurons is observed to be a crucial parameter for the performance of the ELM. In our experiments, we randomly select the number of hidden neurons in the range 2-10% of the number of instances of the dataset. This value results in a well-tuning of the number of hidden neurons that optimizes the quality of the solutions with good prediction accuracy while keeping reasonable execution times. Using higher number of hidden neurons takes a longer time and there is no guarantee that it will improve the quality of the solution.

Four different activation functions are used at the ELM phase of the IPE-ELM algorithm. The activation functions are *Sigmoid, Hyperbolic Tangent, Sine* and *Cosine*. During the fitness value evaluation of each chromosome, one of the activation functions mentioned above is randomly selected along with a random of number of hidden neurons. The main reason of doing this is that during our experiments we obtain varying fitness values for the same selected subset of features when we apply varying activation functions. There is no single activation function type that fits for all datasets. *Sigmoid* activation function can perform good for dataset 1, whereas *Cosine* activation function can work better with dataset 2. The IPE-ELM algorithm considers these possibilities and uses different activation functions during the learning process.

The IPE-ELM algorithm executes as many instances of ELM as the number of processors in the computation environment during the fitness calculation of the chromosomes. Although it is reported to be a fast method, the ELM execution part is the most time-consuming phase of the algorithm. Stagnation at some local optima is an important issue of evolutionary algorithms that must be considered carefully. It can generate serious problems and prevent the optimization process from attaining the optimal solution. The IPE-ELM algorithm is aware of this drawback of the evolutionary computation techniques. Therefore, diversified initial populations are generated at each processor. The possibility of getting better solutions and evaluating many different chromosomes is ensured with this way. This technique is similar to the restarting of the populations from scratch when working with a single-CPU. In a parallel computation environment, we build this mechanism by using several processors. This is provided by selecting a different seeding mechanism for the randomization of all parameters in the FSS and the ELM phases of the algorithm. This technique prevents the genetic algorithm from exploring already searched areas repeatedly.

Table 1 presents the parameters that are used in the evolutionary computation phase of the IPE-ELM algorithm. At all the processors, we use the same parameters that are near-optimal. The flowchart of the IPE-ELM algorithm is presented in Figure 4. The master and slave topology of the IPE-ELM algorithm is presented in Figure 5. The master node starts and controls the execution of the slave nodes. Each slave generates solutions with different starting pseudorandom numbers and executes the optimization individually. At the end of the optimization periods of slaves, the slaves send their results to the master node and the master nodes decides the best solution after collecting all the solutions. There is no deadlock possibility of the processes between the slaves in this topology. Therefore, it is easy to implements and efficient while optimizing the solution of data classification problems. Algorithm 1 gives the details of the IPE-ELM algorithm.

Table 1: Parameter settings for the IPE-ELM algorithm.

Figure 4: Flowchart of the IPE-ELM algorithm.

Figure 5: Master and slave communication topology of the IPE-ELM algorithm.

Selection of parent chromosomes: Truncation is applied after the crossover and mutation operations. The solution

population is divided into two equal parts with respect to the fitness values of the chromosomes and the better half is used for evolving. Later, the better half is divided into two equal segments and starting from the topmost part to the chromosome with the best fitness value, each chromosome is applied crossover and mutation operators by other corresponding chromosomes in the second half. With this operation, 50% more individuals are generated with respect to the original population size. The population size becomes 50% more than its original size. All the chromosomes in this new population are sorted and the worst half of the population is removed.

Termination condition: is decided to be an echelon value that any good solution cannot be found any more. It may change with respect to the execution order of the genetic algorithm. Convergence ratio, 95%, is used during our experiments as the termination condition.

Fitness value calculation: is simply calculated by ELM as an accuracy value in terms of percentage. The objective of a genetic algorithm is to evolve through better solutions and improve the quality of prediction accuracy of the chromosomes in the population.

Algorithm 1: Pseudocode of the IPE-ELM algorithm

```
if (Master processor) then
    Receive results from slaves();
    Find the best result();
    Report the overall best result();
else
    // Code of slave processors;
    p: population;
    generate initial population(p);
    while (termination criterion is not met) do
        par_1, par_2 ← Select parents(p);
        offspring ←Crossover(par_1, par_2);
        offspring ← Mutation(offspring);
        // Extreme Learning Machine Phase
        Decide # hidden neurons (w.r.t. the rank of the processor);
        Decide the activation function (w.r.t. the rank of the processor);
        Calculate hidden-layer output matrix H;
        Calculate (β and T);
        Evaluate H, i.e. the MoorePenrose generalized inverse of matrix H;
        Evaluate the fitness value of the chromosome (selected set of features);
        Insert the offspring chromosome into the population ;
    Send the best result of this processor to the master node;
```

5 Performance Evaluation of the IPE-ELM Algorithm

In this section, we present the experimental setup, the results of a series of experiments carried out to evaluate the prediction accuracy of the IPE-ELM algorithm, the parameter sensitivity of the algorithm and comparison with state-of-the-art data classification algorithms in literature. We carry out experiments on 8 core 64-bit CPU. It is possible to create 8 threads at each core (providing 64 possible cores simultaneously). The server uses 256 GB RAM and 1.5 TB. hard-disk storage.

The IPE-ELM algorithm is developed by using Java on an Open MPI [1] environment. The Open MPI is an open source MPI version that is written and maintained by a collaboration of research, academic, and industry groups. Open MPI combines the technologies, expertise, and resources from all across the HPC community to build the best MPI library. Open MPI provides facilities for system and software administrators, application developers and computer science researchers. Open MPI is developed on Linux and OS X, so it is fairly POSIX-neutral, it runs without modifications on most POSIX-like systems. Open MPI is layered on top of the Open Run-Time Environment (ORTE), which started as a small portion of the Open MPI code base. ORTE is a modular system that is constructed to abstract away the back-end run-time environment system, providing a neutral API to the upper-level Open MPI layer.

1. https://www.open-mpi.org/

Benchmark datasets used in the experiments: are obtained from the University of California (UCI) Machine Learning Repository. The number of features, the number of instances and the actual number of classes of the datasets are given in Table 2. There are 11 different datasets. In order to analyze the performance of the IPE-ELM algorithm, datasets with a small and large number of features/instances are used. *Iris* dataset has 4 features, whereas *musk* has 160 features. *Sonar* has the smallest number of instances (208) and *spam* has the largest number of instances (4,601). These datasets are the most used sets by state-of-the-art algorithms. Therefore, they provide a fair environment during the comparisons.

Table 2: Descriptive statistics of the datasets.

The effect of activation function and the number of hidden neurons: Selecting the most appropriate activation function increases the performance of the ELM significantly. Therefore, we carry out some experiments on this part of our study to detect the best performing activation function. *Sigmoid, Hyperbolic Tangent, Sine,* and *Cosine* are the activation functions that we are used for the ELM phase. At each processor, the ELM selects an activation function randomly and evaluates the accuracy achieved by using the selected features. We carry out our experiments on the *musk, sonar, ionosphere,* and *waveform* datasets. We execute the IPE-ELM algorithm with a single processor (10 times and report the average values) and with increasing number of hidden neurons. The number of hidden neurons is started at 10% of the number of rows (instances) of the datasets *(musk, sonar, ionosphere*) up to 200% and from 1% to 20% for *waveform* dataset (because as the number of neurons is increased the execution time also increases, which can be a big workload during the data classification process). All the features of the datasets are selected during these experiments. No feature subset selection technique is applied. Figures 6, 7, 8 and 9 present the results of our experiments.

Sigmoid is found to be the best performing activation function with datasets, *sonar, ionosphere,* and *waveform*. However, *Cosine* function outperforms others as the best activation function for the *sonar* dataset. In *waveform* dataset experiments, *Sigmoid* function performs better than the other functions. From the observations we make in the activation function experiments, sometimes the performance of the activation function can change depending on the number of hidden neurons. For example, during *ionosphere* dataset experiments (see Figure 8), *Sine* function performed better than all the other activation functions. The number of hidden neurons is nearly 100 with this experiments. Therefore, to make use of the performance of different activation functions under different conditions, we use a random mechanism while selecting an activation function for the IPE-ELM algorithm.

In Figure 10, the execution time of the activation functions are reported for the *waveform* dataset with a different number of hidden neuron numbers starting from 1% to 20% of the number of the instances in the dataset. No big difference is observed between the execution times of the activation functions. Therefore, there is no advantage of any activation function over others in terms of evaluation speed.

Figure 6: Experiments with increasing number of hidden neurons on musk dataset.

Figure 7: Experiments with increasing number of hidden neurons on sonar dataset.

Figure 8: Experiments with increasing number of hidden neurons on ionosphere dataset.

Figure 9: Experiments with increasing number of hidden neurons on waveform dataset.

Figure 10: Execution time of the activation functions in seconds.

Setting the population size: Selecting the parameters of a genetic algorithm to converge to a global optimum quickly is a serious tuning process that can significantly impact the performance of the optimization process. Therefore, we perform some experiments with varying population sizes. Our experiments are executed on *sonar* and *ionosphere* datasets. The population size is selected as 10 at the first step of the experiments. Later, this value is increased by steps of 10 up to 100 individuals. Figures 11 and 13 give the details of our prediction accuracy results with respect to the increasing number of population size for the *sonar* and *ionosphere* datasets respectively. All the population sizes are observed to increase their accuracy levels as the new generations are produced by genetic algorithm. However, small populations are observed to get stuck into local optima while populations larger than 80 individuals tend to spend unnecessarily long time while optimizing the prediction accuracy level. Populations having 70 to 80 individuals are experienced to be among the best performing population sizes for the datasets. Figures 12 and 14 give the execution times of the *sonar* and *ionosphere* datasets respectively for population sizes 10 to 100 until they achieve their termination condition. Populations with a larger number of individuals (more than 70) tend to execute longer times than smaller populations. As a result of our population experiments, we

decide the best performing population to be 70, because its execution time is reasonable and reports near-optimal solutions during the data classification process.

Figure 11: The prediction accuracy results of sonar dataset for populations 10 to 100 until they reach to their termination condition.

Figure 12: Execution times of the sonar dataset for populations 10 to 100 until they reach to their termination condition.

Figure 13: The prediction accuracy results of ionosphere dataset for populations 10 to 100 until they reach to their termination condition.

Figure 14: Execution times of the ionosphere dataset for populations 10 to 100 until they reach to their termination condition.

Testing the prediction accuracy with different number of features: We observe the prediction accuracy of the ELM with different number of selected features. The number of hidden neurons is selected as 20 and activation function is selected as *Sigmoid* during the experiments. The first feature is selected as a starting point, then new features are added one at a time to generate another. The number of features is increased with this way during the experiments. Datasets *spambase* and *musk* are analyzed (see Figures 15 and 16). As it can be seen from the experiments, the number of selected features does not present a stable and predictable performance. Therefore, a good exploration and exploitation algorithm is required for the selection of the best possible set of features. We do these experiments to explain the necessity of selecting the best possible subset for the classification process. Selecting all the features of a dataset does not give the best prediction accuracy results. Therefore, the selection of the best subset of features is an interesting area of research for the data classification, which can improve the accuracy results significantly while also obtaining a faster executing light-weight classifier since it needs to calculate and process less features.

Figure 15: Prediction accuracy results with increasing number of features for the spambase dataset (features are selected from 1 to 57).

Figure 16: Prediction accuracy results with increasing number of features for the sonar dataset (features are selected from 1 to 60).

The prediction accuracy performance deviation of the ELM: is analyzed in this part of our experiments. Since the ELM is using a randomized process, it can have some deviations in its prediction accuracy performance. Figures 17 and 18 give the prediction accuracy results of the ELM for the datasets *sonar* and *ionosphere* respectively. The deviation of the ELM is 7.1% and 2.6% for the *sonar and ionosphere* datasets, respectively. Activation function is *Sigmoid* and the number of hidden neurons is 22 for sonar and 37 for ionosphere respectively. 50 runs are executed during these experiments (x-axis of the graph shows the iteration numbers). Increasing the number of hidden neurons is not observed to have any positive effect on reducing the deviation of the performance of the ELM. Instead, the deviation is observed to change depending on the type (structure) of the dataset that is being classified.

Figure 17: The deviation of prediction accuracy performance of the ELM for the sonar dataset.

Figure 18: The deviation of prediction accuracy performance of the ELM for the ionosphere dataset.

Increasing the number of processors: We experimentally measure the performance of the IPE-ELM algorithm with increasing number of processors. Since the FSS is an intractable problem due to its exponential number of subsets that needs to be evaluated, increasing the number of processors and working on diversified populations is reported to be an efficient way for the solution of the problem [42]. The IPE-ELM algorithm is scalable, which means that as more processors are added to the computation then more fitness values can be evaluated concurrently, enabling the investigation of more and larger subsets of features to be investigated in a short time. Island parallel behavior of the IPE-ELM algorithm provides a very efficient way to calculate the fitness of several chromosomes concurrently. MPI is a powerful library that can support thousands of processors to work in coordination simultaneously. Although we perform experiments with only up to 64 processors, the IPE-ELM algorithm has a potential to getter better results by increasing number of processors. In Figure 19, the performance of the IPE-ELM algorithm with increasing number of processors on the *sonar* dataset is presented. Experiments are performed with increasing number of

processors starting from 5 up to 64. Each test is executed 10 times and the average of the results is presented. 83.3% of the instances are classified correctly with five processors and the accuracy is increased up to 87.6% in the average when the number of processors is 64. This shows that the IPE-ELM algorithm improves its solution quality with increasing number of processors. In Table 3, detailed information for the #processors, time (sec.), accuracy (%), #hidden neurons, activation function type, and #generations are provided. The activation functions *Sigmoid, Cosine,* and *Sine* are observed to perform well on the classification of dataset *sonar*. As we analyse the execution time of the IPE-ELM algorithm, it is observed that the execution time of the last processor that terminates (converges) its generations decides the overall execution time of the algorithm. Some of the processors terminate earlier because of the fast convergence, the others can spend more time on improving their quality of the population. Also, there can be differences in the execution time of the processors due to the varying #hidden neurons. This skewness (slowness) of the algorithm can be prevented by using the same number of generations for each processor. We prefer each processor to operate in its own convergence ratios. We are able to achieve better results with this method.

Figure 19: The performance of the IPE-ELM algorithm with increasing number of processors on the sonar dataset.

Table 3: Detailed information about the increasing number of processor experiments on the sonar dataset.

Prediction accuracy performance of the IPE-ELM algorithm: We observe whether we will have any improvement on the prediction accuracy of the results with the selected features instead of working with all features of a dataset. Table 4 gives the details of the experiments in terms of prediction accuracy improvements when a subset of features is selected intelligently. Improvements ranging from 3.1 to 231.5% are observed for *iris* and *vehicle* datasets, respectively, during the experiments. These results show that selecting the best subset of features can significantly improve the prediction accuracy level of the classification process. Datasets with a small number of attributes are more likely to reach higher prediction accuracy levels with the selected features. The performance increase in *iris* dataset is due to better tuning of the parameters and not from selecting a subset of features.

All the results are the average values of 10 runs with tenfold cross-validation. We use this method to minimize the impact of random factors. In the experiments, the dataset is first partitioned into 10 equal size sets and 9 out of these 10 subsets are used for training while the remaining one is used as a test dataset. Then, the average of these 10 runs is used as *accuracy value*. This is the most common way of evaluating the prediction accuracy performance of the classifiers in literature. Table 5 gives the features of the datasets that are selected by the IPE-ELM algorithm in the experiments. The last column shows the selected features of the datasets.

Table 6 presents the execution time, prediction accuracy, #hidden neurons, activation function type and #generations that give the best solutions after executing with 64 processors. *Sine* and *Sigmoid* functions are observed to be the best performing activation functions. With largest feature numbers, *Sigmoid* outperforms the other activation functions. *Spam* and *Waveform* datasets are observed to spend the longest execution times during the classification process.

Table 4: Comparison of prediction accuracy values when all features and the best performing feature set are selected.

Table 5: The selected features by the IPE-ELM algorithm to provide the best prediction accuracy results.

Table 6: Detailed information about the performance of the IPE-ELM algorithm with all datasets.

The execution time of the IPE-ELM algorithm: is compared with the algorithms that use the same kind of population-based optimization approach. It is observed that the IPE-ELM is a fast polynomial-time algorithm when compared with other state-of-the-art algorithms [5], [18], [19]. The PSO-SVM [27], GA-ELM [26] and HGEFS [9] are the algorithms that we use for comparison on *Arrythmia* dataset (having 279 features and 452 instances). The IPE-ELM algorithm can be decided as one of the fastest algorithms in its class due to the high-speed learning capability achieved by the ELM technique. The execution time of the IPE-ELM algorithm depends on the processor that runs the largest number of generations, which means that some of the processors can get stuck into the local optima and converge earlier than the others so that the master node has to wait for the last processor to finish its final generation. The number of hidden neurons also affects the running time. If the ELM uses larger #hidden neurons, the IPE-ELM executes longer (see Figure 10). The execution time of the algorithms PSO-SVM [27], GA-ELM [26] and HGEFS [9], and the IPE-ELM are 50,493, 4,373, 4,936 and 15,142 seconds, respectively. The IPE-ELM algorithm has a reasonable execution time with respect to the other state-of-the-art algorithms. In Figure 20, the execution time of the algorithms are presented.

Figure 20: Execution time of state-of-the-art algorithms (in seconds).

Scalability and speed-up The IPE-ELM algorithm generates initial populations at the memory of each processor and explores the fitness value of the local populations independently. There is no communication between the processors during their optimization processes. This way of processing the populations provides scalability for the IPE-ELM algorithm. The computation skewness (slowness) is prevented with this way, which means that processors do not have any dependency on the jobs that must be completed by the other processors. The independent population and exploration strategy of the IPE-ELM algorithm provides a scalable performance if and when the new processors can be added to the computation environment. In Figure 19, the performance of the IPE-ELM algorithm is presented for increasing number of processors. The scalability of the IPE-ELM algorithm can be observed easily when the number of processors is increased from 5 to 64 for *sonar* dataset. The total execution time is not affected significantly. The most important issue that decides the execution time of the algorithm is the convergence time of the optimization. If the algorithm gets stuck in local optima it terminates earlier.

The IPE-ELM algorithm provides almost a linear speed-up during the executions. The only computation skewness is due to the selection of a different number of hidden neurons. As the percentage of the number of hidden neurons are increased, the execution time is observed to increase.

Comparison with state-of-the-art algorithms: We compare the IPE-ELM algorithm with state-of-the-art algorithms in literature (Namely, PSO [3], Attribute Bagging (AB) [11], Multi-View Adaboost (MVA) [12], Random Subspacing Ensemble (RSE) [13], (CFS-SFS) [14], C4.5 [15], HGEFS [9], ABACO and ACOFS [16]). Table 7 gives the details of the comparison in terms of prediction accuracy (these are the only results we can be found from the earlier studies which are comparable with our proposed algorithm and our selected set of datasets). The underlined results are the best ones obtained by the state-of-the-art algorithms. The IPE-ELM algorithm produces the best results for seven datasets. These results are the best solutions that have been reported by any algorithm so far. *spam*, *musk*, and *wave* datasets are best solved by ACOFS, HGEFS and PSO algorithms respectively. The *musk* dataset has the largest number of features. The overall prediction accuracy performance of the IPE-ELM algorithm is observed to be 89.65% for the datasets in the experiments. These values are the best-reported results in literature. It is still possible to add new processors and improve the solution quality of the IPE-ELM algorithm.

Table 7: Comparison of prediction accuracy results with state-of-the-art algorithms.

6 Conclusions and future work

In this study, we present a novel Island Parallel Evolutionary Extreme Learning Machine algorithm (IPE-ELM) for the data classification problem. The ELM is an efficient machine learning technique with fast speed learning capability and high prediction accuracy. In addition, it is well known that effective feature selection methods can improve the quality of the ELM. We combine ELM with a parallel evolutionary algorithm and propose a robust data classification algorithm. Activation functions together with the number of hidden neurons are tuned for the first time in this study by using a scalable algorithm. Our results show that the IPE-ELM is among the best performing algorithms with its high prediction accuracy rate (89.65% in the average for the experimented datasets). The search space of the data classification problem has many parameters that must be well tuned for a better optimization. For this reason, a more powerful and intelligent tool such as parallel computation is required to calculate multiple alternatives intelligently and improve the solution quality. Because the IPE-ELM algorithm is scalable, it is possible to increase the performance of the IPE-ELM algorithm by adding more processors to the MPI computation environment. The IPE-ELM algorithm is scalable and can increase the prediction accuracy of the results. FSS problem is an open research area due to its intractable nature.

As future work, the data classification problem can be solved by using GPU supported hybrid metaheuristic algorithms. Multi-objective data classification with parallel computation can be another direction of research. Artificial Bee Colony (ABC) is a recent and efficient metaheuristic that can be applied to this important problem. Hyper heuristics is a novel area that uses the best practices of the heuristics to obtain near-optimal results

References

[1] M. A. H. Ian H. Witten, Eibe Frank, Implementations, Data Mining: Practical Machine Learning Tools and Techniques (2011) 191–304.
[2] N. M. Nasrabadi, Pattern recognition and machine learning, Journal of electronic imaging 16.
[3] A. Unler, A. Murat, A discrete particle swarm optimization method for feature selection in binary classification problems, European Journal of Operational Research 206 (2010) 528–539.
[4] C.-K. S. Guang-Bin Huang, Qin-Yu Zhu, Extreme learning machine: a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), IEEE, 2004.
[5] Q.-Y. Z. Guang-Bin Huang, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.
[6] G.-B. Huang, D. H. Wang, Y. Lan, Extreme learning machines: a survey, International journal of machine learning and cybernetics 2 (2) (2011) 107–122.
[7] S.-M. Boln-Canedo, Alonso-Betanzos, A review of feature selection methods on synthetic data, Knowledge and Information Systems 34 (2012) 483–519.
[8] L. W. Mingkui Tan, Ivor W. Tsang, Towards ultrahigh dimensional feature selection for big data, Journal of Machine Learning Research 15 (2014) 1371–1429.
[9] M. Y. Xiaowei Xue, Z. Wu, A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm, Knowledge and Information Systems 57 (2017) 389–412.
[10] C. C. Aggarwal, Algorithmic Graph Theory and Perfect Graphs, 1st Edition, CRC Press, 2014.
[11] O. R.Bryll, R.Gutierrez, Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets, Pattern Recognition 36 (2003) 1291–1302.
[12] S. S. Z. Xu, An algorithm on multi-view adaboost, Neural Information Processing. Theory and Algorithms Lecture Notes (2010) 355–362.
[13] T.K.Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 832–844.
[14] M. A. Hall, Correlation-based feature selection for machine learning, Ph.D. thesis, The University of Waikato (1998).
[15] J. Quinlan, Improved use of continuous attributes in c4.5, Journal of Artificial Intelligence Research 4 (1996) 77–90.
[16] H. N.-P. Shima Kashef, An advanced aco algorithm for feature subset selection, Neurocomputing 147 (2015) 271–279.
[17] B. Chen, L. Chen, Y. Chen, Efficient ant colony optimization for image feature selection, Signal processing 93 (6) (2013) 1566–1576.
[18] X. D. Guang-Bin Huang, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (2010) 757–754.
[19] H. Z. Guang-Bin Huang, X. Ding, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (2012) 513–529.
[20] M. Dash, H. Liu, Feature selection for classification. intelligent data analysis, Intelligent Data Analysis 1 (1997) 131–156.
[21] V. J.Yang, Feature subset selection using a genetic algorithm, Feature Extraction, Construction and Selection (1998) 117–136.
[22] T. D. A. Deniz, H. Kiziloz, A. Cosar, Robust multiobjective evolutionary feature subset selection algorithm for binary classification using machine learning techniques, Neurocomputing 241 (2017) 128–146.
[23] P. Q.Y.Zhu, A.k. Qin, G.B.Huang, Evolutionary extreme learning machine, Pattern recognition 38 (2005) 1759–1763.
[24] S. G.Huang, G.B.Huang, K.You, Trends in extreme learning machines: A review, Neural Networks 61 (2015) 32–48.
[25] T. H.E.Kiziloz, A.Deniz, A.Cosar, Novel multiobjective tlbo algorithms for the feature subset selection problem, Neurocomputing 306 (2018) 94–107.
[26] S. S.-S. E. Alexandre, L. Cuadra, A. Pastor-Snchez, C. Casanova-Mateo, Hybridizing extreme learning machines and genetic algorithms to select acoustic features in vehicle classification applications, Neurocomputing 152 (2015) 58–68.
[27] L. J. J. Garca-Nieto, E. Alba, E. Talbi, Sensitivity and specificity based multiobjective approach for feature selection: Application to cancer diagnosis, Information Processing Letters 109 (2009) 887–896.
[28] G.Xu-L.Papageorgiou, A mixed integer optimisation model for data classification, Computers & Industrial Engineering 56 (2009) 1205–1215.
[29] A. H.Kartal, A.Oztekin, F.Cebi, An integrated decision analytic framework of machine learning with multicriteria decision making for multi-attribute inventory classification, Computers & Industrial Engineering 101 (2016) 599–613.
[30] P. T.Y.Ou, C.Y.Cheng, C.Perng, Dynamic cost forecasting model based on extreme learning machine-a case study in steel plant, Computers & Industrial Engineering 101 (2016) 544–553.
[31] Y. Sun, Y. Yuan, G. Wang, An os-elm based distributed ensemble classification framework in p2p networks, Neurocomputing 74 (16) (2011) 2438–2443.
[32] X. Li, W. Mao, W. Jiang, Extreme learning machine based transfer learning for data classification, Neurocomputing 174 (2016) 203–210.
[33] J. Xin, Z. Wang, C. Chen, L. Ding, G. Wang, Y. Zhao, Elm*: distributed extreme learning machine with mapreduce, World Wide Web 17 (5) (2014) 1189–1204.
[34] E. M.Heeswijk, Y.Miche-Erkki, A.Lendasse, Gpu-accelerated and parallelized elm ensembles for large-scale regression, Neurocomputing 74 (2011) 2430–2437.
[35] F. Q.He, T.Shang, Z.Shi, Parallel extreme learning machine for regression based on mapreduce, Neurocomputing 102 (2013) 52–58.
[36] Z. Jiarun Lin, J.Yin, Q.Liu, K.Li, V.Leung, A secure and practical mechanism for outsourcing elms in cloud computing, IEEE Intelligent Systems 28 (2013) 35–38.
[37] R. N. Bo He, Dongxun Xu, M. Heeswijk, Q. Yu, Y. Miche, A. Lendasse, Fast face recognition via sparse coding and extreme learning machine, Cognitive Computation 6 (2013) 264–277.
[38] E. Cant-Paz, A survey of parallel genetic algorithms, Calculateurs paralleles, reseaux et systems repartis 10 (1998) 141–171.
[39] T. D. Umut Tosun, A. Cosar, A robust island parallel genetic algorithm for the quadratic assignment problem, International Journal of Production Research 51 (2013) 4117–4133.
[40] T. Dokeroglu, A. Cosar, Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms, Computers & Industrial Engineering 75 (2014) 178–186.
[41] T. Kucukyilmaz, H. E. Kiziloz, Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem, Computers & Industrial Engineering 125 (2018) 157–170.
[42] H. E. Kiziloz, T. Dokeroglu, A robust and cooperative parallel tabu search algorithm for the maximum vertex weight clique problem, Computers & Industrial Engineering 118 (2018) 54–66.

7 Figures and Tables

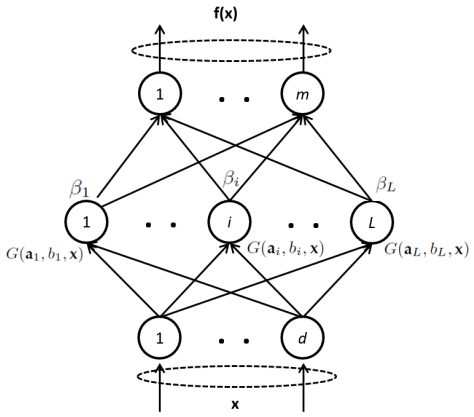

Fig. 1: Single-hidden layer feed-forward network.

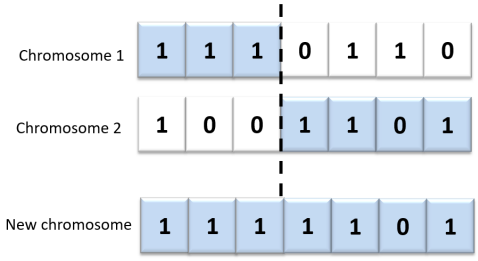

Fig. 2: The crossover operator that works on two chromosomes. A new child is generated from the left part of first chromosome and right part of the second chromosome.

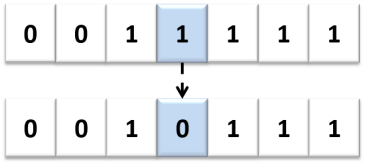

Fig. 3: The mutation operator changes the value of the fourth gene to generate a new chromosome

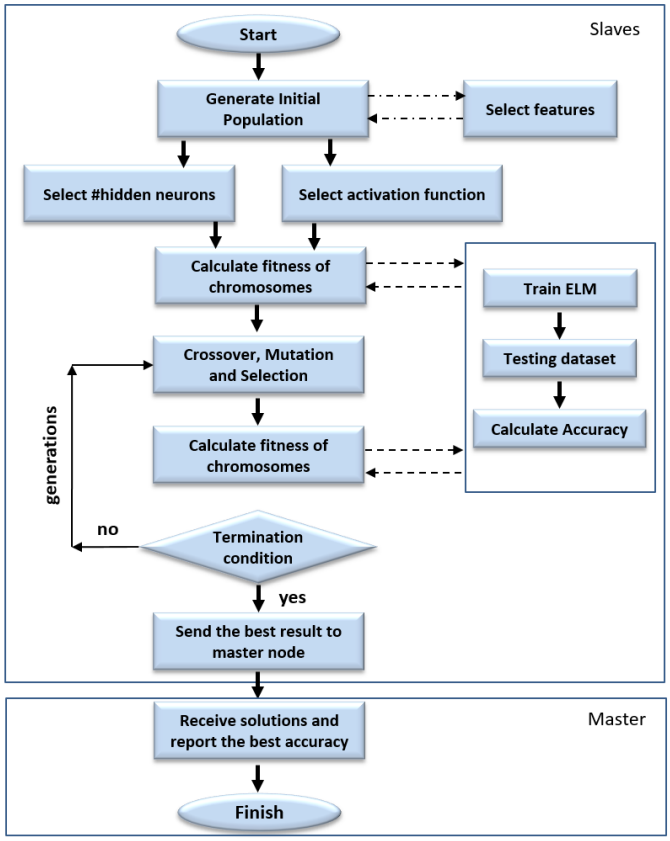

Fig. 4: The flowchart of the IPE-ELM algorithm.

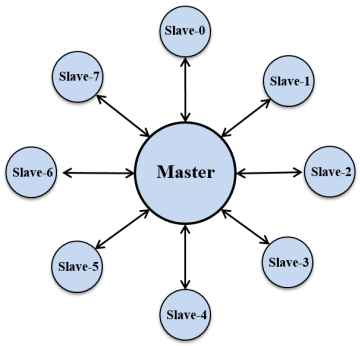

Fig. 5: Master and slave communication topology of the IPE-ELM algorithm.

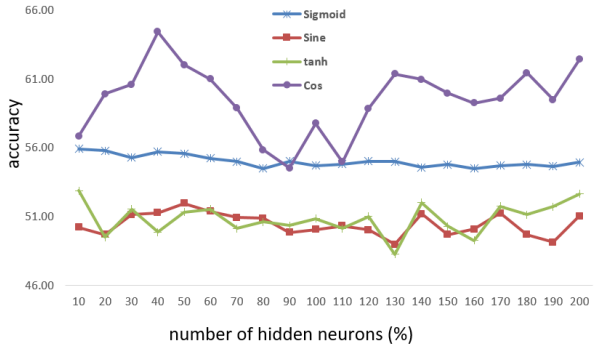

Fig. 6: Experiments with increasing number of hidden neurons on *musk* dataset

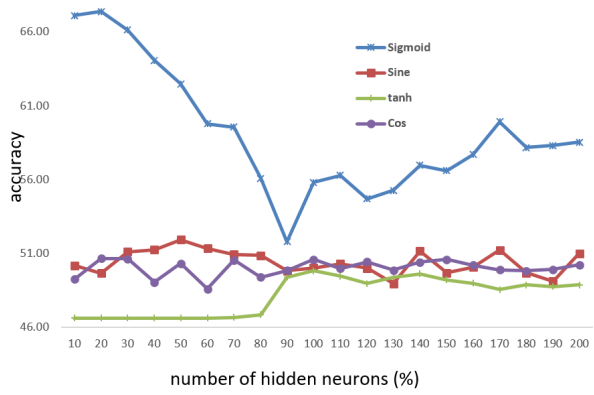

Fig. 7: Experiments with increasing number of hidden neurons on *sonar* dataset.

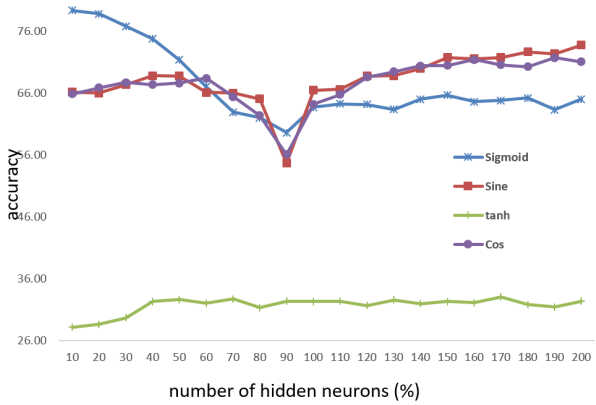

Fig. 8: Experiments with increasing number of hidden neurons on *ionosphere* dataset.

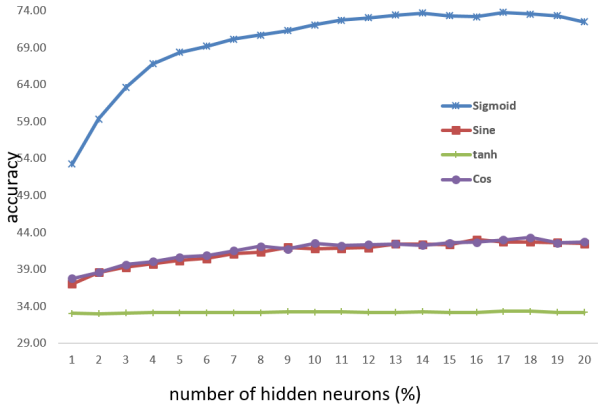

Fig. 9: Experiments with increasing number of hidden neurons on *waveform* dataset.

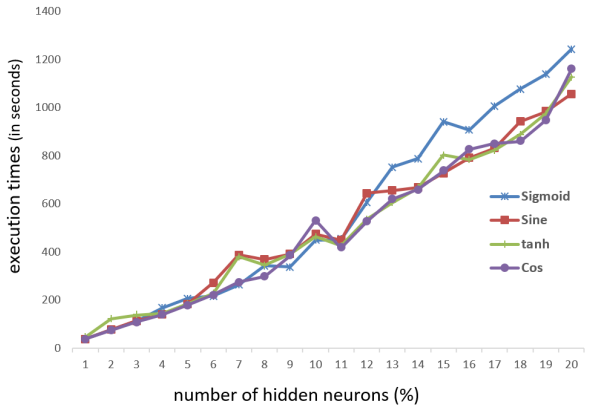

Fig. 10: Execution time of the activation functions in seconds.

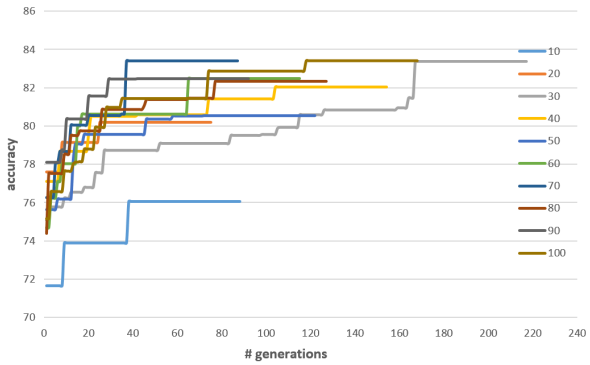

Fig. 11: The prediction accuracy results of *sonar* dataset for populations 10 to 100 until they achieve their termination condition.

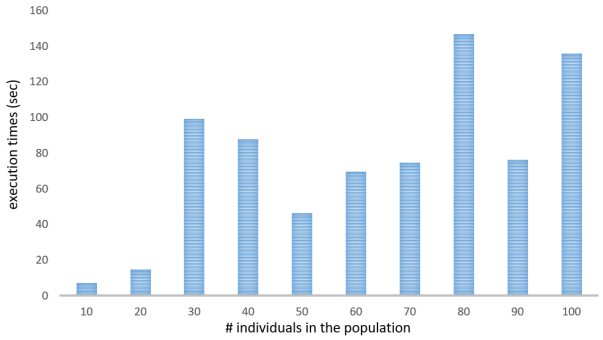

Fig. 12: Execution times of the *sonar* dataset for populations 10 to 100 until they achieve their termination condition.

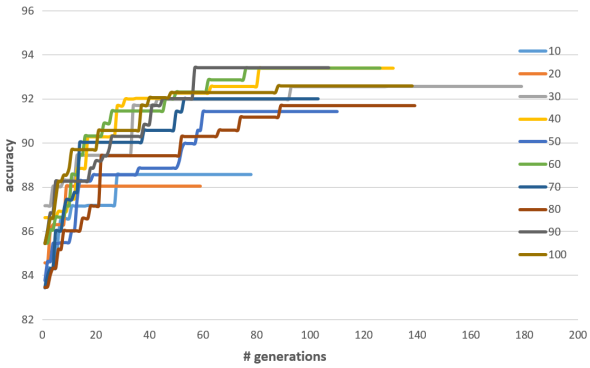

Fig. 13: The prediction accuracy results of *ionosphere* dataset for populations 10 to 100 until they achieve their termination condition.

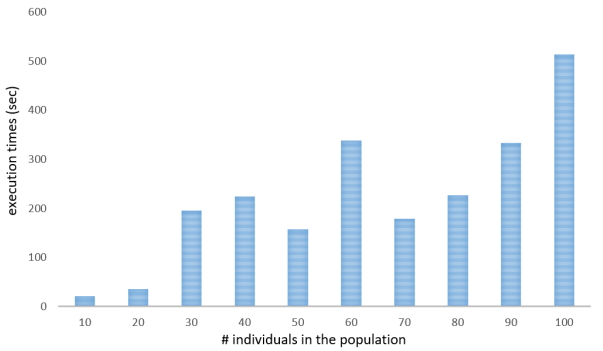

Fig. 14: Execution times of the ionosphere dataset for populations 10 to 100 until they achieve their termination condition.

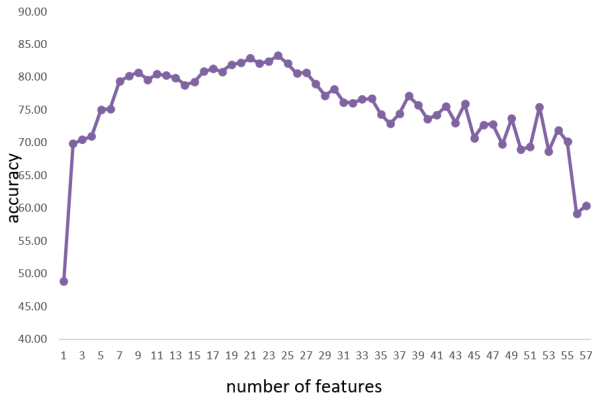

Fig. 15: Prediction accuracy results with increasing number of features for the *spambase* dataset (features are selected from 1 to 57).

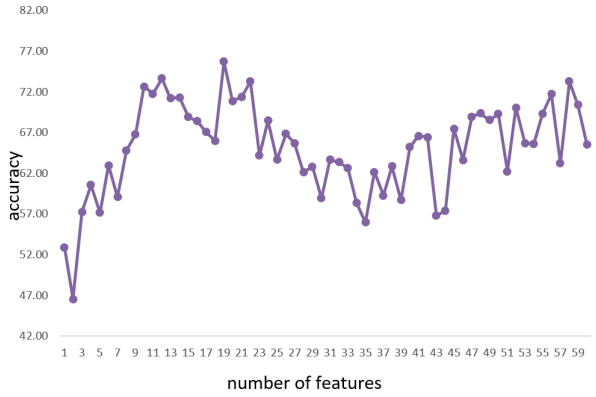

Fig. 16: Prediction accuracy results with increasing number of features for the *sonar* dataset (features are selected from 1 to 60).

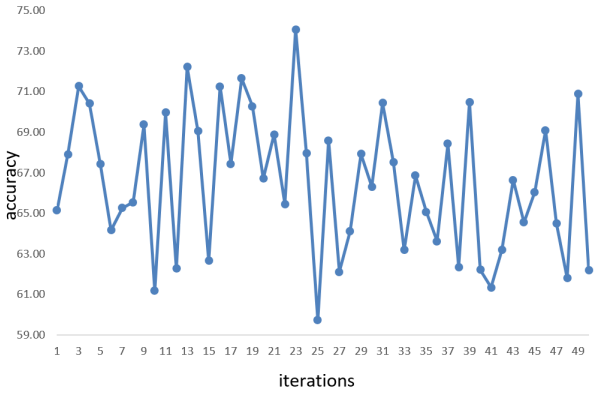

Fig. 17: The deviation of prediction accuracy performance of the ELM for the *sonar* dataset.

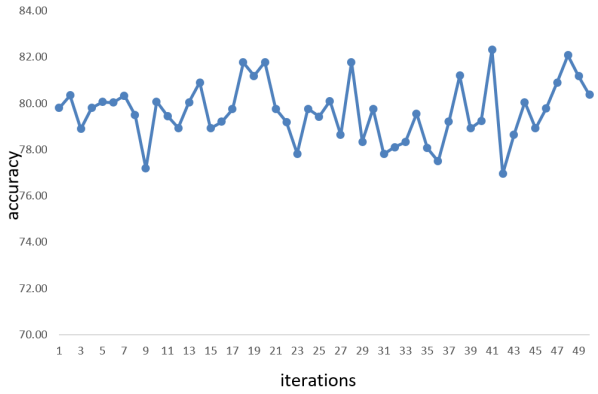

Fig. 18: The deviation of prediction accuracy performance of the ELM for the *ionosphere* dataset.

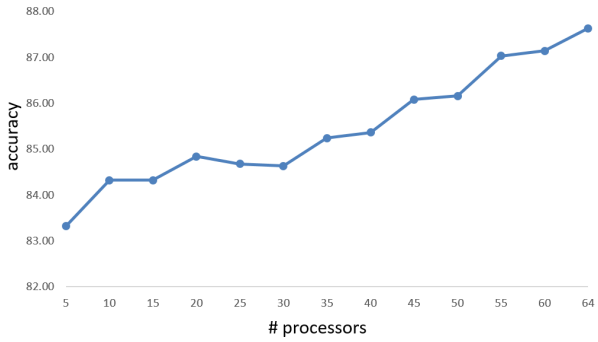

Fig. 19: The performance of the IPE-ELM algorithm with increasing number of processors on the *sonar* dataset.

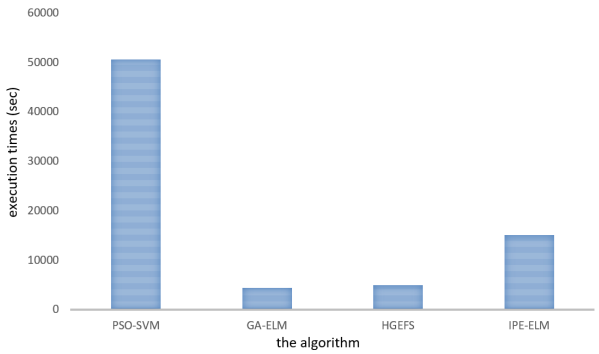

Fig. 20: Execution time of state-of-the-art algorithms (in seconds).

TABLE 1: The parameter settings for the IPE-ELM algorithm

parameter	setting
Population size	70
Convergence ratio	95%
Truncate ratio	50%
Crossover ratio	30%
Mutation ratio	1%

TABLE 2: The descriptive statistics of the datasets

ID	Dataset	#features	#instances	#classes
ION	IONOSPHERE	34	351	2
SON	SONAR	60	208	2
VEH	VEHICLE	18	846	4
MUS	MUSK	166	476	2
IRI	IRIS	4	150	3
PID	Pima-Indian Diabetes	8	768	2
WDB	WDBC	32	569	2
WIS	Wisconsin B.C.(Or.)	10	699	2
CHS	CHESS	36	3,196	2
SPM	SPAM	57	4,601	2
WAV	WAVEFORM	21	569	3

TABLE 3: Detailed information about the increasing number of processor experiments on the *sonar* dataset.

# processors	time (sec.)	accuracy (%)	#hidden neurons	activation func.	#generations
5	203	83.32	15	Sigmoid	192
10	116	84.32	10	Sigmoid	150
15	225	84.32	23	Cosine	147
20	58	84.84	7	Sigmoid	93
25	157	84.68	9	Cosine	232
30	263	84.63	17	Cosine	202
35	108	85.24	12	Sigmoid	102
40	200	85.36	11	Sigmoid	195
45	222	86.08	22	Sine	120
50	405	86.16	19	Sine	252
55	100	87.03	9	Sigmoid	98
60	247	87.14	20	Cosine	142
64	263	87.63	17	Cosine	202

TABLE 4: Comparison of prediction accuracy values when all features and the best performing feature set are selected.

ID	all features selected	selected features	improvement(%)
ION	79.6	93.44	17.3
SON	67.3	87.63	30.2
VEH	25.5	84.55	231.5
MUS	55.9	81.91	46.5
IRI	96.3	99.33	3.1
PID	52.3	78.13	41.3
WDB	55.4	97.53	76.0
WIS	93.4	97.81	4.7
CHS	58.6	94.56	61.3
SPM	59.4	90.76	52.7
WAV	54.7	80.54	47.2

TABLE 5: The selected features by the IPE-ELM algorithm to provide the best prediction accuracy values.

ID	accuracy	selected features
ION	93.44	0-1-0-0-1-0-0-0-0-0-0-0-1-0-0-0-0-0-0-0-0-0-0-0-1-0-1-0-0-0-0-0
SONAR	84.86	0-0-1-1-1-1-1-0-0-0-1-1-0-0-0-0-1-0-0-0-0-1-0-0-0-0-1-1-0-1-1-0-0-0-0-0-1-0-0-1-0-0-0-0-1-1-1-0-1-1-1-1-0-1-0-0-0-0-1-0
VEH	84.55	1-0-1-1-1-1-1-1-0-1-0-0-0-1-0-0-1-1
MUS	81.91	1-0-1-1-0-1-1-1-1-1-0-1-0-1-0-0-1-1-1-1-1-0-0-1-0-0-1-1-0-0-0-0-0-0-0-0-1-0-1-0-0-0-0-0-1-1-1-0-0-0-1-1-1-0-1-1-0-1-1-0-1-0-0-0-1-1-0-0-0-0-1-1-1-1-0-0-0-1-0-0-1-1-0-0-0-1-0-1-0-1-1-1-0-1-0-0-0-1-0-1-0-1-0-0-0-0-0-1-0-1-1-1-0-1-0-1-0-1-0-0-0-1-1-0-0-1-0-1-0-1-0-0-0-1-1-0-0-1-0-1-0-1-0-0-0-0-0-1-0-1-1-0-0-1
IRI	93.33	1-1-1-1
PID	78.13	0-1-0-0-0-1-1-1
WDB	97.53	1-1-0-0-0-1-1-1-0-0-0-0-0-0-1-0-1-0-0-1-1-1-0-0-1-0-1-1-1-1
WIS	97.80	1-0-1-0-0-1-0-1-0-1
CHS	94.55	1-0-0-0-0-1-0-0-1-1-0-0-0-1-1-0-1-0-1-0-1-1-0-0-0-0-0-0-0-0-0-1-1-0-1-0
SPM	90.76	0-0-0-0-1-0-1-0-1-0-0-0-0-0-0-1-1-0-0-0-1-0-0-1-1-1-0-1-0-0-0-0-1-0-0-0-0-0-1-1-0-1-1-1-0-0-0-1-1-0-0-0-0-0-1-1-1-0-0-1-1-1-1-0-1-1-0-1-0-1-1-0-0-0-1-0-1-0-1-0-0-0-0-0-1-0-0-1-1-1-0-0-1-1-1-0-0-0-0-0-1-0-1-0-1-0-1-1-1-0-1-0-1-0-1-0-0-0-0-1-0-1-1-1-0-0-0-1-0-1-0-1-0-0-0-1-1-0-0-1-0-1-0-1-0-0-0-0-0-1-0-1-1-0-0-1
WAV	80.53	0-0-0-0-1-1-0-0-0-1-1-0-1-0-1-0-1-0-0-0-0

TABLE 6: Detailed information about the performance of the IPE-ELM algorithm with all datasets.

ID	time (sec.)	accuracy (%)	#hidden neurons	activation func.	#generations
ION	750.9	93.44	24	Sine	233
SON	324	87.63	23	Sine	156
VEH	2047,442	84.55	70	Sigmoid	109
MUS	2,182,795	81.91	36	Sigmoid	159
IRI	5707	99.33	151	Sine	48
PID	974,236	78.13	84	Sigmoid	71
WDB	249,885	97.53	58	Sigmoid	49
WIS	894,762	97.81	40	Sigmoid	97
CHS	7,106,789	94.56	20	Sigmoid	91
SPM	18,564,449	90.76	20	Sigmoid	96
WAV	156,046,222	80.54	210	Sigmoid	66

TABLE 7: The comparison of prediction accuracy results with state-of-the-art algorithms.

ID	**PSO** [3]	**AB** [11]	**MVA** [12]	**RSE** [13]	**CFS-SFS** [14]	C4.5 [15]	**HGEFS** [9]	**ABACO** [16]	**ACOFS** [17]	IPE-ELM
ION	89.4	89.54	90.14	89.01	89.06	91.16	91.33	-	-	**93.44**
SON	-	80.83	80.17	79.50	78.75	71.15	83.00	-	-	**87.63**
VEH	-	80.95	81.20	77.32	69.17	73.64	82.02	75.3	74.9	**84.55**
MUS	-	85.27	85.63	84.73	79.55	84.87	**88.13**	-	-	81.91
IRI	-	-	-	-	-	-	-	97.4	97.7	**99.33**
WDB	-	95.14	95.73	94.84	95.80	93.14	97.10	-	-	**97.53**
WIS	96.0	-	-	-	-	-	-	97.6	97.4	**97.81**
SPM	90.2	-	-	-	-	-	-	92.1	92.2	**90.76**
WAV	**90.6**	-	-	-	-	-	-	79.5	79.7	80.54