View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Montana

University of Montana

ScholarWorks at University of Montana

Graduate Student Theses, Dissertations, & Professional Papers

Graduate School

2003

Parallel Genetic Ensemble Feature Selection

Navin Elango
The University of Montana

Follow this and additional works at: https://scholarworks.umt.edu/etd

Let us know how access to this document benefits you.

Recommended Citation

Elango, Navin, "Parallel Genetic Ensemble Feature Selection" (2003). *Graduate Student Theses, Dissertations, & Professional Papers*. 5103.
https://scholarworks.umt.edu/etd/5103

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.

Maureen and Mike
MANSFIELD LIBRARY

The University of
Montana

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

Please check "Yes" or "No" and provide signature

Yes, I grant permission ✓

No, I do not grant permission ____

Author's Signature: [illegible]

Date: July 28 2003

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.

8/98

Parallel Genetic Ensemble Feature Selection

A Thesis

presented in partial fulfillment of the requirements

for the degree of

Master of Science

with a major in Computer Science

in the

The University of Montana

August 2003

by

Navin Elango

MSc(Tech). Engineering Technology,

Birla Institute of Technology and Science, Pilani, India.

Approved by:

[illegible]

Chairperson

[illegible]

Dean, Graduate School

7-30-03

Date

UMI Number: EP40567

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI
Dissertation Publishing

UMI EP40567

Published by ProQuest LLC (2014). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.
All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code

ProQuest

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Navin Elango. MS, August 2003 Computer Science

Parallel Genetic Ensemble Feature Selection

Director: Dr. David W. Opitz [illegible]

Ensembles have emerged as a useful machine learning technique, in which several individual learning models are used together to solve a problem. The combination of many learning models often improves predictive power. Researchers have found that ensembles work well if each of the learning model entities are accurate and at the same time diverse. The choice of input features to be used to define the problem greatly impacts the predictive power of the learning model. Irrelevant features may act destructive to a learning algorithm. Feature selection algorithms address this problem by finding the optimal subset of features to be presented to the learning model in order to achieve greater predictive power. Traditionally, feature selection algorithms have been used to find the optimal subset of features to be presented to a single learning algorithm. The motivation of ensemble feature selection is to find the optimal feature subset to be presented to each member of an ensemble. Genetic Ensemble Feature Selection (GEFS) is an ensemble feature selection approach that is based on genetic algorithms.

The most time consuming part of the GEFS algorithm is the training of each learning algorithm in the ensemble. The speed of the GEFS algorithm can be greatly increased by parallelizing and training each learning algorithm in a different processor. This research extends the GEFS algorithm by parallelizing it using the OpenMp paradigm and presents the results of the analysis of GEFS and Parallel GEFS algorithms on various domains.

Acknowledgements

I am grateful to God for the infinite love and happiness. I am thankful to my parents Mr. T. Elango and Mrs. E. Sumathi for their eternal love and encouragement. I thank my advisor Dr. David Opitz for the inspiration and knowledge he has provided me with. I thank Dr. Don Morton and Dr. Brian Steele for their guidance and useful suggestions during this work. I am indebted to my many student colleagues at the Machine Learning Research Laboratory for their enthusiastic help. Lastly, but importantly, I thank all faculty and staff of the Computer Science Department at the University of Montana, Missoula for their friendliness and support.

Contents

List of Figures and Tables

1.Introduction

Machine Learning is the study of computer algorithms that improve automatically through experience [Mitchell 97]. An algorithm is said to gain experience whenever it changes its structure or data based on its inputs, to be better adapted for its future performance. These algorithms, also called *learning algorithms* or *learners,* create a trained model from a set of training data. This trained learning model can then be used for tasks like decision-making, prediction, and further refinement of knowledge. Applications of machine learning range from automatic spell checker in text editors to precision guided missiles.

The suitability of a learning model for a particular task depends on the accuracy of the model on performing that task. The accuracy of a learning model relies on the inputs presented to it. Obviously, irrelevant inputs will confuse the learner and make it less accurate. Increasing the accuracy of a learning model by giving it an optimal set of inputs (*features*) has been an active research area in the field of machine learning [Aha 94], [Aha, Bankert 95]. *Feature selection algorithms* are algorithms that search for an optimal set of features to be presented to a learning model.

Traditionally, feature selection algorithms have been used to find the optimal subset of features to be presented to a single learning model. The Genetic Ensemble Feature Selection (GEFS) algorithm [Opitz 99] is a novel approach to find the optimal feature subset to be presented to an *ensemble*. Ensembles ([Hansen, Salamon 90], [Perrone, Cooper 93], [Krogh, Vedelsby 95], [Opitz, Shavlik 96], [Maclin, Opitz 97]) have emerged as a useful machine learning technique, in which several individual learning models are used together to solve a problem. The combination of many learning models often improves the accuracy. Bagging [Breiman 96] and Boosting [Freund et al. 96] are examples of successful algorithms used to create ensembles.

GEFS uses a genetic algorithm [Koza 92] to search for the optimal subset of features to be presented to an ensemble. GEFS increases the accuracy of the ensemble, but it is inefficient in terms of the time taken to execute the algorithm. This occurs because it trains a new learner for each potential optimal feature set found by the genetic algorithm. Each one of these new models then becomes a potential component for the ensemble.

Parallel Processing is a concept for speeding-up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. By training each potential component learner of the ensemble on different processors, we can increase the efficiency of the GEFS algorithm. This research extends the GEFS algorithm by parallelizing it and presents the results of the analysis of GEFS and Parallel GEFS algorithms on various domains. The GEFS algorithm has been parallelized using the OpenMp [Chandra et al. 2000] paradigm. Results show that the efficiency of GEFS can be increased significantly by parallelization.

The rest of the thesis is organized as follows. In Chapter 2, a brief description about unsupervised learning, artificial neural networks, feature selection algorithms and genetic algorithms is given. In Chapter 3, the GEFS is briefly introduced. Chapter 4 proposes

Parallel GEFS. Chapters 5, 6 and 7 explain the datasets, experimental methodology and results respectively. Finally the last chapter draws conclusions and presents several issues for future research.

2. Background

This thesis focuses on a category of machine learning algorithms called *supervised learning algorithms or empirical learning algorithms.* In supervised learning, the learner inductively learns a target function *f,* called the *target concept,* from a set of examples. The learner is given a set of examples called the *training set* of the form $\{(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)\}$, where *n* is the number of training examples. Each example in the training set is called an *instance*. The x_i values are typically in vector form $<x_{i1}, x_{i2} \ldots x_{im}>$, where *m* is the number of features. This vector is called the *feature set.* The *y* value is the output of the target concept when applied to the feature set of that particular instance. The y value is also called as the *label or class.*

The goal of a supervised learning algorithm is to generate a close approximation of the target concept, based on the training set. In supervised learning, training is the process of generating an approximation of the target concept from the training set. A trained learner should then be able to *classify* (predict the class of) novel instances; that is, the instances that were not a member of the training set. This set of novel instances is called the *test set.* The *accuracy* of an algorithm on a set of examples can be defined as the rate of correct predictions made by the model over that set.

$$Accuracy = \frac{Number\ of\ examples\ classified\ correctly}{Total\ number\ of\ examples\ in\ the\ set} * 100$$

This thesis studies a type of supervised learning algorithm called Artificial Neural Network. The following subsections give a brief description of Artificial Neural Networks, Ensembles, Feature Selection and Genetic Algorithm.

2.1 Artificial Neural Networks

Artificial Neural Network is a supervised learning algorithm motivated by the way a human brain processes information. An Artificial Neural Network consists of many non-linear computational units operating in parallel and arranged in patterns similar to biological neural networks in the brain. These computational units are arranged in *layers* (see Figure 1 below) and are connected with *weights* that are customized during the training process to learn the target concept.

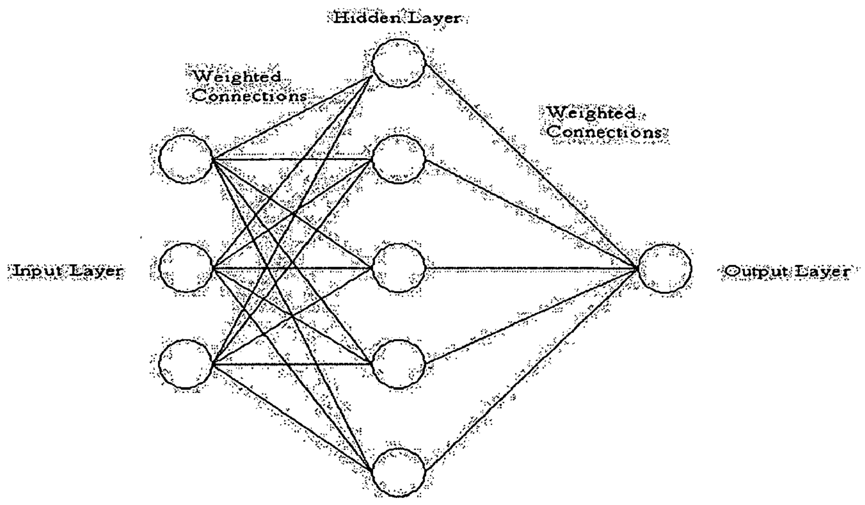

Figure 1. An artificial neural network

The back-propagation algorithm [Arbib 95] is an algorithm that can be used for training Artificial Neural Networks. The goal of back-propagation is to minimize the mean square error (E_q) over the training set by adjusting the weights.

$$E_q = \sum_{p}^{n_p} \sum_{o}^{n_o} (t_o^p - y_o^p)^2$$

Where n_p, n_o, t_o^p, y_o^p are the number of instances in the training set, number of output units, target output and the output predicted by the neural network respectively. It is called back-propagation because the error is computed first at the output layer and then propagated backward through the network, to compute the errors at the hidden layers.

Artificial Neural Networks are suited for a broad range of problems. They have been successfully used for nonlinear modeling and approximation in fields like speech recognition [Barnard et al. 95] and expert systems [Gallant 93]. For certain types of problems, such as learning to interpret complex real-world sensor data, Artificial Neural Networks are among the most effective methods currently known [Mitchell 97]. The increasing significance of Artificial Neural Networks is demonstrated by the fact that currently many of the standard statistical software packages include Artificial Neural Network modeling in their toolbox.

An Artificial Neural Network is accurate and robust when trained with a large feature set. It can approximate complex target functions and multiple interactions between features. It does not restrict the type of data presented as input and is fairly insensitive to noise or unreliability in data. These properties of the Artificial Neural Network make it suitable for this thesis.

2.2 Ensembles

An ensemble consists of a committee of individually trained supervised learning models whose predictions are combined when classifying novel instances [Maclin, Opitz 97]. The learners in the committee are trained to learn the same target function. The output of the individual learners can be combined in different ways like averaging or weighted averaging to get the output of the ensemble. The ensemble that we consider in this thesis is a simple ensemble of neural networks that averages the output of each individual learner. Figure 2 shows a simple Ensemble Learning model.

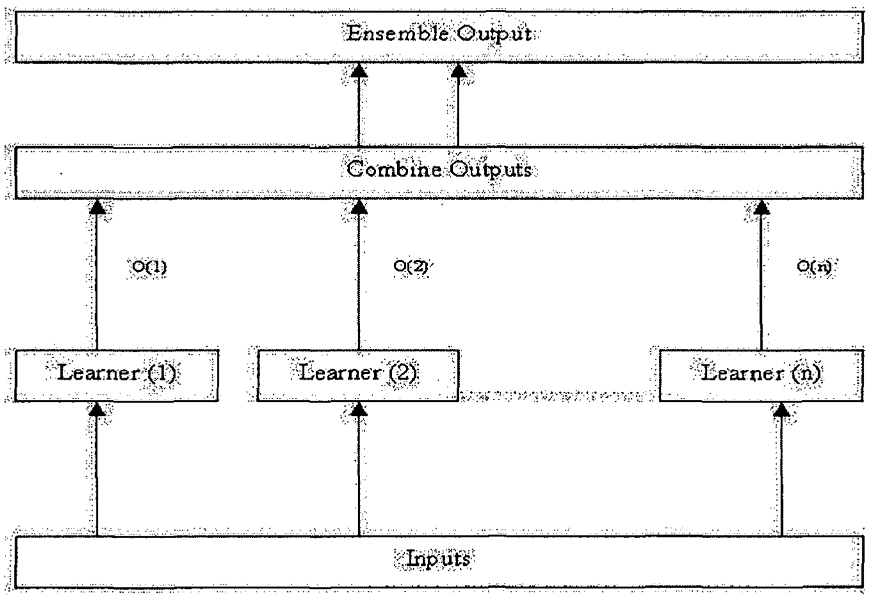

Figure 2. An Ensemble Learning Model

Many researchers ([Opitz, Shavlik 96], [Krogh, Vedelsby 95] and [Hansen, Salamon 90]) have shown that a good ensemble is one in which the learners are *accurate* and at the same time make their errors on different parts of the input space (*diverse*). Bagging and boosting are two popular methods of creating a good ensemble by random sampling of the training instances presented to the individual learners. Several machine learning scientists (for e.g. [Hansen, Salamon 90], [Perrone, Cooper 93]) have investigated ensembles and proved that the combination of individual learners normally improves the predictive power. The accuracy of the ensemble is often higher than the accuracy of the component learning models. This makes ensembles the learning algorithm of choice for many real world applications where accuracy is a key factor.

2.3 Feature Selection

Feature selection is defined as the process of selecting the best subset of features out of a larger set of features to be presented to a classification algorithm to maximize its performance. In supervised learning, the information known about the class of an instance is inherent to the features presented to the classifier and determines the accuracy of the learner. When presented with many features that are not necessary for predicting the desired output, learning algorithms such as Artificial Neural Networks perform poorly (degrade in accuracy). Searching for an accurate subset of features is a difficult search problem. Search spaces to be explored could be very large. Feature selection algorithms have three components [Aha 94]:

- *The Search Algorithm,* which searches through the different possible sets of features to find the best that can be presented to the classification model.
- *The Evaluation function,* which evaluates which feature subset is better for the learning algorithm. It takes in a feature subset as its input, and outputs a numeric value that provides a measurement of the feature subset's suitability to the problem. The goal of the search algorithm is to maximize this function. This function is user defined. If the user wants to increase the accuracy of the learning model, the evaluation function can just be the accuracy of the learning model when a particular feature subset is used.
- *The Classifier* is the learning model for which the algorithm is searching the feature subset. It can be a single learning model like a neural network or a combination of learning models like an ensemble of neural networks.

Based on the manner in which these three components interact with each other, the feature selection algorithms can be divided into the *filter model*, and the *wrapper model.*

2.3.1 The Filter Model

Figure 3 shows the filter model. The filter model works by searching for the best feature subset independent of the classifier. The best feature subset is selected based on the correlation between the features and the relevance of each feature to the problem (which can be obtained from the training set which is to be presented to the classifier). The disadvantage of this method is that it does not consider the effect of the selected features on the classifier.

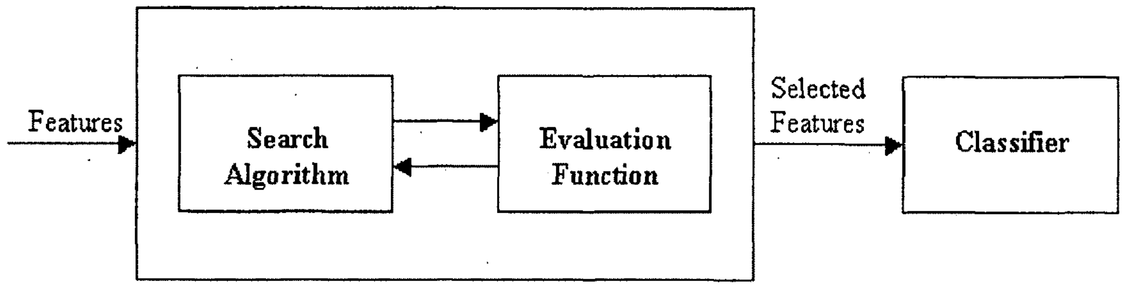

Figure 3. The Filter Model

2.3.2 The Wrapper Model

In the wrapper model, the evaluation function consults the classifier algorithm to find the best feature subset. In this case, all three components, namely, the search algorithm, the evaluation and the classifier algorithm work closely. The evaluation function considers the biases of the classifier when selecting features (see Figure 4 below). This method chooses the features based on the performance of the classifier, by giving preference to a classification model with high predictive accuracy on unseen data.

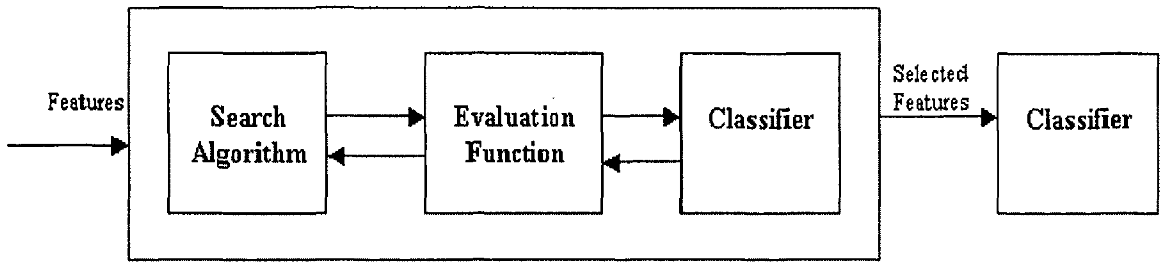

Figure 4. The Wrapper Model

2.4 Genetic Algorithms

Genetic algorithms (GAs) provide a search method motivated by an analogy to biological evolution [Mitchell 97]. Unlike most of the search algorithms, which gives one solution to a search problem, the GA gives a set of solutions as output. The following sections give a brief description of the biological background and the algorithm respectively.

2.4.1 Biological Background

All living organisms are made of cells, which is the basic unit of life. In each cell there is the same set of structures called *chromosomes.* This *population* of chromosomes is the information store of the cell. Chromosomes are made of discreet units called *genes* .During reproduction, the genes undergo *crossover.* Crossover is process in which parent chromosomes exchange parts to form the offspring. The newly created offspring can then be mutated. *Mutation* means that small elements of gene are changed randomly. These changes are mainly caused by errors in copying genes from parents. These offspring's along with the current population forms the new population. Evolution follows the concept of *survival of the fittest;* only the fittest among the chromosomes in the population reproduce and survive. After many generations of crossovers and mutations, the chromosome, and hence the cell is said to be fit for the environment.

2.4.2 The Algorithm

Genetic algorithms start with a random set of solutions *encoded* as chromosomes. The fitness of each of the candidate solutions (chromosome) is evaluated by *a fitness function.* The GA then applies *genetic operators* such as mutation and crossover to evolve the solutions in order to find the best one(s). As the outline of the algorithm suggests, the three main components of the GA are the encoding, fitness function and the genetic operators, which are illustrated below.

- *Encoding*

To use the GA, we have to encode a potential solution of the problem on a chromosome like data structure. The commonly used data structures are binary strings and trees.

- *Fitness Function*

The fitness of each of the solutions to the problem is evaluated by the fitness function. Given the encoded solution as input, the fitness function gives a numerical value of the fitness as output.

- *Genetic Operators*

Crossover and *mutation* are the two types of operators that are normally used. The *crossover* operator produces two new offspring from two parent strings, by copying selected bits from each parent [Mitchell 97]. The bit at position *i* in each offspring is copied from the bit at position *i* in one of the two parents. The *mutation* operator introduces a certain amount of randomness. It chooses a single bit at random and changes its value. It can help find solutions that crossover alone might not encounter.

3. Genetic Ensemble Feature Selection

GEFS is a novel ensemble feature selection approach that searches for the optimal subset of features to be presented to an ensemble, hence increasing its accuracy. Moreover, presenting a different feature subset to each learner in the ensemble makes it diverse and consequently produces a better ensemble. The GEFS algorithm is a classic example of the *wrapper model.* The search algorithm used by GEFS is a Genetic Algorithm. Each individual chromosome of the GA is a potential optimal feature set .The Genetic Algorithm consults the classifier (in this case an ensemble) each and every time it evaluates the fitness of an individual. The evaluation function used by the algorithm is

$$\mathit{Fitness}\ (i) = \mathit{accuracy}\ (i) + \lambda * \mathit{diversity}\ (i)$$

Where *accuracy (i)* is the training set accuracy of the component learner that was trained using the feature subset *i. Diversity (i)* is the average difference between the prediction of the component learner and the ensemble. λ is the tradeoff between accuracy and diversity. The GEFS algorithm is shown in Table 1 below. GEFS automatically changes λ based on the discrete derivates of the ensemble error $\hat{E}$, the average population error $\ddot{E}$ and the average diversity *Đ* within the ensemble. λ is never changed when $\hat{E}$ is decreasing . When $\hat{E}$ is increasing, GEFS (a) increases λ when $\ddot{E}$ is not increasing and *Đ* is decreasing, or (b) decreases λ when $\ddot{E}$ is increasing and *Đ* is not decreasing.

```
GOAL: Find a set of input subsets to create an accurate and diverse classifier ensemble.
1. Using varying inputs, create initial population of learners.
2. Train the initial population of learners
3. Until a stopping criterion is reached:
    (a) Use genetic operators to create new learners.
    (b) Train the new learners and calculate its accuracy on the training set.
    (c) Measure the diversity of each learner with respect to the current population.
    (d) Normalize the accuracy scores and the diversity scores of the individual learners.
    (e) Calculate the fitness of each population member.
    (f) Prune the population to the N fittest learners.
    (g) Adjust $\lambda$
    (h) The current population composes the ensemble.
```

Table 1. The Genetic Ensemble Feature Selection (GEFS) Algorithm

Opitz successfully demonstrated that the GEFS algorithm increases the accuracy of the ensemble [Opitz 99]. Although GEFS produces a better ensemble by creating highly accurate and diverse component learners, it is slow. This occurs because it trains a new learner for each potential optimal feature set found by the genetic algorithm (Step 3b). This new learner then becomes a potential learner to be added to the ensemble. The Parallel Genetic Ensemble Feature Selection (PGEFS) increases the efficiency of GEFS by training each potential component learner in a different processor. The next section describes the PGEFS algorithm.

4. Parallel GEFS

Parallel processing can be used to increase the efficiency of an algorithm if the algorithm has parts that are independent of each other and can be executed simultaneously. In GEFS, the training of each component learner in the ensemble is independent of each other. This makes step 2 and 3b of the GEFS algorithm (see Table 1) suitable for parallelization. Table 2 shows the PGEFS algorithm.

```
GOAL: To efficiently find a set of input subsets to create an accurate and diverse classifier ensemble.
1. Using varying inputs, create initial population of learners.
2. Train each learner in the initial population in a different processor.
3. Until a stopping criterion is reached:
(a) Use genetic operators to create new learners.
(b) Train each of the new learners in a different processor and calculate its accuracy on the training set.
(c) Measure the diversity of each learner with respect to the current population.
(d) Normalize the accuracy scores and the diversity scores of the individual learners.
(e) Calculate the fitness of each population member.
(f) Prune the population to the N fittest learners.
(g) Adjust λ
(h) The current population composes the ensemble.
```

Table 2. The Parallel Genetic Ensemble Feature Selection (PGEFS) Algorithm

The main differences between the GEFS and the PGEFS algorithm are in step 2 and step 3b. The PGEFS as opposed to GEFS trains each of the potential component learners in a different processor. This reduces the time taken to execute the algorithm. The OpenMP [Chandra et al. 2000] parallel programming model for *shared memory* multiprocessors was used to implement the PGEFS algorithm. In *shared memory architecture*, the processors are allowed to communicate with each other using variables stored in a shared address space. The OpenMP is a portable application program interface with a set of library functions, compiler directives, and environment variables that can be used for shared-memory parallelism. It supports parallel programming in Fortran and C/C++ on UNIX and Windows NT architectures. OpenMp provides a simple, flexible, portable and scalable interface for developing shared-memory parallel applications. It was developed by a group of major hardware and software vendors and standardized for easy use.

5. Datasets

The datasets for this thesis were obtained from the University of California Irvine dataset repository [Murphy, Aha 94] and the University of Wisconsin Machine Learning repository. Table 3 below gives a brief description of the datasets.

Dataset	Number of Instances	Number of classes	Number of Continuous Features	Number of Discrete features
Breast-cancer	699	2	-	9
Credit-a	690	2	6	9
Credit-g	1000	2	7	13
Diabetes	768	2	9	-
Glass	214	6	9	-
Heart-Cleveland	303	2	8	5
Hepatitis	155	2	6	13
House-votes-84	435	2	-	16
Ionosphere	351	2	34	-
Iris	159	3	4	-
Labor	57	2	8	8
Promoters-936	936	2	-	57
Sonar	208	2	60	-
Soybean	683	19	-	35
Vehicle	846	4	18	-

Table 3. Dataset Description

As the table depicts, the datasets are diverse in the number of discrete and continuous features and in the number of instances. These datasets were obtained to solve real world problems where the accuracy of the learning model is crucial. This supplements the suitability of the datasets for testing GEFS and PGEFS.

6. Experiments and Methodology

The experiments were run on The National Center for Supercomputing Applications (NCSA) Silicon Graphics Origin2000 supercomputer. The Origin2000 is a cache coherent, non-uniform memory access supercomputer. It has sixty-four MIPS R10000 processors. Each of the processors has a clock speed of 195 MHz and shares a memory of 16 Gigabytes. It runs the IRIX 6.5 operating system. All the settings except the number of processors used were maintained constant for GEFS and PGEFS. The Table 4 below shows the neural network settings that were used for the experiments.

Neural Network Settings:

Learning Rate: 0.1

Num Epochs: 100

Momentum: 0.9

Initial Random weights: [- 0.5 to 0.5]

Dataset	Number of Input Nodes	Number of output Nodes	Number of hidden Nodes
Breast-cancer		2	-
Credit-a	47	1	10
Credit-g	63	1	10
Diabetes	8	1	5
Glass	9	6	10
Heart-Cleveland	13	1	5
Hepatitis	32	1	10
House-votes-84	16	1	5
Ionosphere	34	1	10
Iris	4	3	5
Labor	29	1	10
Promoters-936	228	1	20
Sonar	60	1	10
Soybean	134	19	25
Vehicle	18	4	10

Table 4. Neural Network Settings

The Table 5 below shows the settings for the GEFS and PGEFS algorithm. The number of processors used to evaluate PGEFS was 25.

Variable	Value
Number of Learners in Ensemble	25
Number of Learners searched	250
Minimum number of learners evolved in each generation	25
Crossover Probability	0.5
Mutation Probability	0.5
Initial λ value	1.0

Table 5. GEFS Settings

Speedup is defined as the factor by which the time to execute the program is improved using multiple processors compared to using only a single processor.

$$Speedup = \frac{\text{Time taken to run on a single processor}}{\text{Time taken to run on multiple processors}}$$

Since the goal of PGEFS is to increase the computational efficiency of GEFS, we consider speedup as the most important metric for our experiments. In the experiments, we compare the accuracy of GEFS with PGEFS on the fifteen datasets and calculate the speedup that can be achieved using twenty-five processors. In GEFS and PGEFS there are two accuracies that we are interested in. The *Initial Accuracy*, of the ensemble before using GEFS or PGEFS to select features and the *Final Accuracy* of the ensemble after using GEFS or PGEFS to select features.

Accuracy is determined using standard *N-Fold cross validation,* which is a reliable accuracy estimation technique. The dataset is first divided into *N* subsets. Each time, *N-1* subsets are put together to form a training set and the one left out is used as the test set. The model is trained and tested N times, choosing a different training and test set from the *N* sets. Then the average accuracy across all *N* trials is computed. The advantage of this method is that it does not matter how data gets divided into test set and training set. Every instance gets to be in a test set exactly once, and gets to be in a training set *N-1* times.

In our experiments, we calculated the 5-fold ($N = 10$) cross validation accuracy on each dataset. For each fold, a new ensemble was created and evolved using GEFS or PGEFS. In other words, for each fold, a new ensemble of 25 learners was created and 250 learners were searched. We ran five trials of 5 fold cross validation on each dataset and averaged the accuracy and time values. We then calculated the speedup from the average time taken to run GEFS and PGEFS on each dataset.

7.Results and Discussion

Table 6 below shows the results. The accuracies shown are the initial and final 5-fold cross validation accuracies. All the accuracies and time taken were averaged over five trials.

Dataset	GEFS Initial Population Accuracy (%)	GEFS Final Population Accuracy (%)	Time Taken (Seconds)	PGEFS Initial Population accuracy (%)	PGEFS Final population accuracy (%)	Time Taken (Seconds)	Speedup
Breast-cancer	94.82	95.95	2638.02	95.32	96.06	188.11	14.02
Credit-a	81.89	85.93	8016.13	83.77	87.95	671.9	11.93
Credit-g	71.9	73.32	14486.94	69.72	74.58	1189.98	12.17
Diabetes	73.47	75.72	2644.41	73.01	73.88	181.89	14.53
Glass	74.74	73.66	1088.12	72.95	75.33	77.03	14.12
Heart-Cleveland	75.63	76.7	1383.86	74.31	77.56	98.91	13.99
Hepatitis	81.57	81.96	1164.72	81.73	79.12	95.69	12.17
House-votes-84	89.99	94.26	2049.07	90.72	93.55	150.21	13.64
Ionosphere	82.97	87.49	2438.66	83.05	84.8	196.73	12.39
Iris	94.44	95.1	968.79	82.72	86.93	69.85	13.86
Labor	91.71	90.16	531.96	93.87	94.12	45.57	11.67
Promoters-936	91.04	94.54	211675	90.38	93.72	19699.31	10.74
Sonar	81.91	80.72	4047.53	81.58	83.61	346.12	11.69
Soybean	93.64	94.71	14758.55	94.8	95.13	1406.93	10.48
Vehicle	81.93	82.89	5326.91	80.74	82.97	348.77	15.27

Table 6. Experiment results

The results demonstrate that PGEFS is more efficient than GEFS. Table 6 also shows that the speed of GEFS is increased by ten to fifteen times when using twenty-five processors without adversely affecting the accuracy of the ensemble. The accuracies of the GEFS and PGEFS vary a little because of the randomness involved in the genetic algorithm search and the neural network learning. Although training of the component learners of the ensemble is the major time consuming part of the GEFS algorithm, training them in parallel using twenty-five processors did not speedup the algorithm by a factor of twenty-five. This is because the other important parts of the algorithm, such as the genetic operations, were not parallelized. Parallelizing the genetic algorithm and Artificial Neural Network learning algorithm should further increase the performance of PGEFS.

8.Conclusions and Future Work

The goal of our research is to increase the efficiency of the GEFS algorithm by parallelizing it. The initial results that we have presented in this thesis are encouraging. We achieved a speedup of a factor of ten to fifteen using twenty-five processors. Though our research illustrated that PGEFS has a better performance than GEFS, there are certain issues that require further investigation in future research. Future work includes:

- studying the effects of parallelizing the neural network and genetic operators;
- analyzing the relationship between various parameters (number of learners searched, number of learners evolved in each generation, number of inputs) and speedup;
- measuring the effects of the number of processors used to parallelize on speedup; and
- parallelizing GEFS using the distributed memory architecture.

Bibliography

[Aha 94] Feature selection for case based classification of cloud types: an empirical comparison. *Case-Based Reasoning: Papers from the 1994 Workshop, edited by David W Aha, Technical Report WS-94-07, pp. 106-112. Menlo Park, CA: AAAI Press.*

[Aha, Bankert 95] A comparative evaluation of sequential feature selection algorithms: David W Aha, Richard. L Bankert. *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.} Ft. Lauderdale.*

[Arbib 95] The Handbook of Brain Theory and Neural Network: Michael A. Arbib. *The MIT Press, 1995.*

[Barnard et al. 95] Real world speech recognition with neural networks: E. Barnard, R. Cole, M. Fanty and P. Vermeulen.

[Breiman 96] Bagging Predictors: Breiman *Machine Learning*, 24(2): 123-140, 1996.

[Chandra et al. 2000] Parallel Programming in OpenMP: R. Chandra, R. Meoon, L. Dagum, D. Kohr, D. Maydan, J. McDonald. *Morgan Kaufman, 2001.*

[Freund et al. 96] Experiments with a New Boosting Algorithm: Yoav Freund, Robert E. Schapire. *Thirteenth International Conference on Machine Learning, 1996.*

[Gallant 93] Neural Network Learning and Expert Systems: Gallant. S. *MIT Press, Cambridge, Massachusetts.*

[Hansen, Salamon 90] Neural Network Ensembles: L. Hansen and P. Salamon. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1990.*

[Koza 92] Genetic Programming: On the Programming of Computers by Means of Natural Selection: John Koza. *MIT Press, 1992.*

[Krogh, Vedelsby 95] Neural Network Ensemble, cross validation and active learning: Krogh,A. Vedelsby, J. *Neural Information Processing Systems volume 7, 231-238. Cambridge, MA. MIT Press.*

[Maclin, Opitz 97] An Empirical Evaluation of Bagging and Boosting: Maclin. R, Opitz. D. *Fourteenth National Conference on Artificial Intelligence (AAAI),* Providence, Rhode Island.

[Mitchell 97] Machine Learning: Tom M. Mitchell, *McGraw-Hill Publications, 1997.*

[Murphy, Aha 94] UCI repository of machine learning databases (machine- readable data repository): Murphy, P.M., and Aha D.W. 1994. *The University of California-Irvine, Department of Information and Computer Science*

[Opitz, Shavlik 96] Actively searching for an effective neural-network ensemble: Opitz. D, Shavlik. J, *Connection Science 8(3/4): 337-353*

[Opitz 99] Feature Selection for Ensembles: Opitz, D. (1999). *Sixteenth National Conference on Artificial Intelligence (AAAI),* (379-384). Orlando, FL.

[Perrone, Cooper 93] When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. Michael P. Perrone, Leaon N. Cooper. *Neural Networks for Speech and Image Processing, 1993.*

[Ripley 96] Pattern Recognition and neural networks: B.D. Ripley, *Cambridge University Press, 1996.*