

University of Montana

## ScholarWorks at University of Montana

---

Graduate Student Theses, Dissertations, &  
Professional Papers

Graduate School

---

2000

### Genetic ensemble feature selection

Ganesh J. Prabu

*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.**

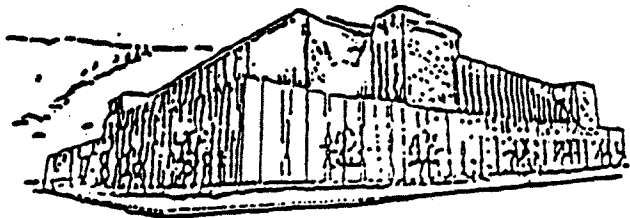
---

#### Recommended Citation

Prabu, Ganesh J., "Genetic ensemble feature selection" (2000). *Graduate Student Theses, Dissertations, & Professional Papers*. 5099.

<https://scholarworks.umt.edu/etd/5099>

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](mailto: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 \_\_\_\_\_

Date \_\_\_\_\_

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

# Genetic Ensemble Feature Selection

by

Ganesh J Prabu

BE, Computer Science and Engineering,

Government College of Technology, Coimbatore, INDIA

presented in partial fulfillment of the requirements

for the degree of

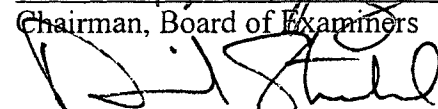
Master of Science

The University of Montana

December 2000

Approved by:

  
Chairman, Board of Examiners

  
Dean of the Graduate School

12-20-00

Date

UMI Number: EP40563

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 EP40563

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

Ganesh J Prabu, MS, December 2000,

Computer Science

Genetic Ensemble Feature Selection *DNO*

Director: David Opitz, Ph.D.

Feature selection is the process of picking the relevant features of a task so as to maximize the accuracy of an inductive learning algorithm when applied to that task. Irrelevant features confuse a learning algorithm and thus decrease accuracy. At the same time there is a potentially opposing need to include a sufficient set of relevant features to achieve acceptably high performance. This has led to the development of a variety of search techniques for finding an “optimal” set of features to apply the learning task, but most of these techniques focus on finding the feature set for *one* learned model. Ensembles are a set of learned models that act cooperatively in their predictions. Ensembles have been shown (Breiman 1996; Hansen and Salamon 1990) to be more accurate on average than learning a single model. Given the success of ensembles, Opitz (AAAI, 1999) presented a GEFS (Genetic Ensemble Feature Selection) algorithm that uses a “genetic algorithm” to search the feature selection for ensembles. GEFS has been shown to generalize better than existing ensemble approaches using backpropagation as its component learning algorithm.

In this thesis we extend the GEFS algorithm to include the inductive learning algorithms of naïve Bayesian, K-nearest neighbor and probabilistic neural networks. Our experiments demonstrate the quality of this approach for each of these learning algorithms across a wide variety of problem domains. For each inductive learner, GEFS generalizes better than the single component learning algorithm.

## Acknowledgements

I am extremely thankful to Dr. David Opitz for his help in providing funding and inspiration for this study, for the machine learning background, and for the countless hours of consultation and review of this document. I would like to thank Dr. Alden Wright for the software engineering background that I needed to implement GEFS, and Dr. Brian Steele for his help with the statistical aspects of the project. I would like to thank Sean J. Hart and Mark Hammond of the Naval Research Laboratory for providing me the probabilistic neural network code. I want to thank Kathy Lockridge for her assistance from the day I join this university. I want to thank Mohan Prabhu and Shreedevi Prabhu for their support and for motivating me to see this project through to completion. I want like to thank Jegathees Chandran and Hema Latha for giving me an opportunity to do my higher studies.

# TABLE OF CONTENTS

ABSTRACT.....	II
ACKNOWLEDGEMENTS.....	III
1 INTRODUCTION .....	1
2 RELATED WORK .....	4
2.1 <i>Learning Algorithms</i> .....	4
2.1.1 Backpropagation .....	5
2.1.2 Naïve Bayesian .....	6
2.1.3 K-nearest neighbor.....	8
2.1.4 Probabilistic Neural Networks.....	10
2.2 <i>Ensembles</i> .....	12
2.3 <i>Feature Selection</i> .....	13
2.4 <i>Genetic Algorithms</i> .....	15
3 GENETIC ENSEMBLE FEATURE SELECTION.....	17
3.1 <i>The GEFS Algorithm</i> .....	17
3.2 <i>Genetic Operators</i> .....	20
3.3 <i>Extensions to GEFS</i> .....	20
4 METHODOLOGY .....	21
5 RESULTS .....	27
5.1 <i>Classification Tasks</i> .....	27
5.2 <i>Regression Tasks</i> .....	37
6 DISCUSSION AND FUTURE WORK .....	44
7 CONCLUSION .....	46
8 REFERENCES .....	47

## LIST OF TABLES

TABLE 1 GENETIC ENSEMBLE FEATURE SELECTION .....	18
TABLE 2 SUMMARY OF THE CLASSIFICATION DATASETS .....	21
TABLE 3 REGRESSION DATASETS .....	22
TABLE 4 FOUR LEVELS OF COMPLEXITY.....	23
TABLE 5 PROPERTIES OF THE DATASETS .....	24
TABLE 6 ERROR RATES FOR THE NAÏVE BAYESIAN .....	28
TABLE 7 ERROR RATES FOR THE K-NEAREST NEIGHBOR .....	29
TABLE 8 ERROR RATES FOR THE BACKPROPAGATION .....	30
TABLE 9 ERROR RATES FOR PNN .....	31
TABLE 10 TEST ERROR OF ALL LEARNING ALGORITHMS.....	36
TABLE 11 CORRELATION COEFFICIENT FOR BACKPROPAGATION .....	37
TABLE 12 CORRELATION COEFFICIENT FOR K-NEAREST NEIGHBOR .....	40
TABLE 13 $R^2$ ERROR FOR K-NEAREST NEIGHBOR AND BACKPROPAGATION.....	43



## LIST OF FIGURES

FIGURE 1 LAYOUT OF ANN .....	5
FIGURE 2 K-NEAREST NEIGHBOR, $k = 5$ .....	9
FIGURE 3 ENSEMBLES FRAME WORK .....	12
FIGURE 4 WRAPPER APPROACH TO FIND FEATURED SET.....	14
FIGURE 5 ERROR RATES FOR NAÏVE BAYESIAN CLASSIFIER.....	32
FIGURE 6 ERROR RATES FOR K-NEAREST NEIGHBOR.....	33
FIGURE 7 ERROR RATES FOR BACKPROPAGATION.....	34
FIGURE 8 ERROR RATES FOR PNN.....	35
FIGURE 9 CORRELATION COEFFICIENT FOR BACKPROPAGATION .....	39
FIGURE 10 CORRELATION COEFFICIENT FOR K-NEAREST NEIGHBOR.....	42

# 1 Introduction

In feature subset selection, a learning algorithm is faced with the problem of selecting a relevant subset of features upon which to focus its attention, while ignoring the rest. Previous work on feature selection has focused on finding the appropriate subset of relevant features for one learned model. However, recent work on ensembles has shown that combining the output of a set of models that are generated from separately trained inductive learning algorithms can greatly improve generalization accuracy (Breiman, 1996; Maclin and Opitz 1997; Shapire et al. 1997). Opitz (AAAI, 99) presented an approach to feature selection for ensembles called GEFS (Genetic Ensemble Feature Selection). Opitz showed GEFS generalizes better than existing ensemble approaches using backpropagation as its component learning algorithm. *This thesis extends GEFS to include the inductive learning algorithms of the naïve Bayesian classifier, K-nearest neighbor and probabilistic neural networks (PNN) and apply these algorithms to a wide variety of problem domains.*

The objective of the feature selection is to reduce the number of features used to characterize a dataset so as to improve an algorithm's performance on a given task. In machine learning, finding a minimal set of relevant features increases the generalization accuracy and to a lesser extent the speed. This has led to the development of a variety of search techniques for finding an "optimal" subset of features from a larger set of possible features. Exhaustively trying to find all the subsets is computationally prohibitive when there are a large numbers of features. There are two main approaches for avoiding this

combinatorial explosion. The first involves developing problem specific strategies (heuristics), which use domain knowledge to prune the *feature space to a manageable size* (Dom, B., Niblack, W., and Sheinvald, J. 1989). The second approach is to use generic search strategies (primarily hill climbing algorithms) when domain knowledge is costly to exploit or unavailable (Kittler, J. 1978). This thesis concentrates on the problems where there is no domain-specific knowledge available.

In this thesis, we focus on a genetic algorithm (GA) approach to search for the best feature subsets. We use GAs because they have demonstrated substantial *improvement over a variety of random and local search methods* (De Jong, K, 1975) on large search spaces, such as the feature subset selection problem. Many researchers have used GAs to do the feature selection (J. Yang and V. Honavar, 1997; Leardi, R, 1994; Rainer Stotzka et al. 2000), but they all focus on finding the feature selection for one learned model.

Opitz (AAAI, 1999) presents the GEFS algorithm that does feature selection for ensembles. GEFS accomplishes this by creating an initial population of classifiers where each classifier is generated randomly. It then continually produces new candidates by using the generic genetic operators like crossover and mutation. The fitness of the classifier is a combination of accuracy and diversity. The fit individuals make up the population. Opitz used backpropagation as its inductive learner.

In this thesis, we have extended the GEFS algorithm using the inductive learning algorithms of naïve Bayesian, K-nearest neighbor and PNNs. In this thesis we have also extended GEFS for problems requiring regression analysis (i.e., problems where the targets are real values); whereas, Opitz (1999) only concentrates on classification tasks.

Our results show that GEFS applied to these learning algorithms increases the generalization accuracy over the single component learning algorithms on a wide variety of domains. For the most part, GEFS produces a good initial population is both fast and accurate. The accuracy of the system increases as the system continues to run.

## 2 Related Work

Machine Learning is a field in Computer Science that aims to make computers learn from experience. The necessary step in any machine learning application is to know the available learning algorithms and their characteristics. This helps to identify the learning algorithm that is most suitable for the task. This chapter gives an introduction to inductive learning algorithms and an overview of the methods that are used in this thesis.

### 2.1 Learning Algorithms

An inductive learner is a system that learns from a set of examples. The set of examples that is given to a learning algorithm is known as a training set. Each example will have a set of inputs (independent variables) and a set of outputs (dependent variables). The learning algorithm will learn from this training set and come up with a hypothesis that will be able to predict the output for any unseen example. There are many different types of inductive learners; each having their own bias for a particular type of problem. One has to choose the learning algorithm that is best suited for his/her assignment based on the property of the learning task and the dataset characteristics. A properly chosen learning algorithm should achieve high generalization accuracy, which is the accuracy that a learner encompasses with the examples beyond the training data. We next provide an overview of the inductive learning algorithms used in this thesis: backpropagation, K-nearest neighbor, naïve Bayesian classifier, and probabilistic neural networks.

### 2.1.1 Backpropagation

Backpropagation is the most common training technique for artificial neural networks (ANN) with hidden layers (intermediate layers). ANNs are weighted interconnections of nodes that are loosely based on the replicated structure of the human brain cells called neurons. ANNs work by propagating inputs forward through the network to the outputs. The output of each neuron in an ANN is calculated by first summing the values of each incoming link (input multiplied by connection weight), then converting it, in a nonlinear fashion, into a number that the program can use (a real number between 0 and 1, for example). The connection weights and topology of the ANN determine the knowledge function. Figure 1 shows a typical ANN.

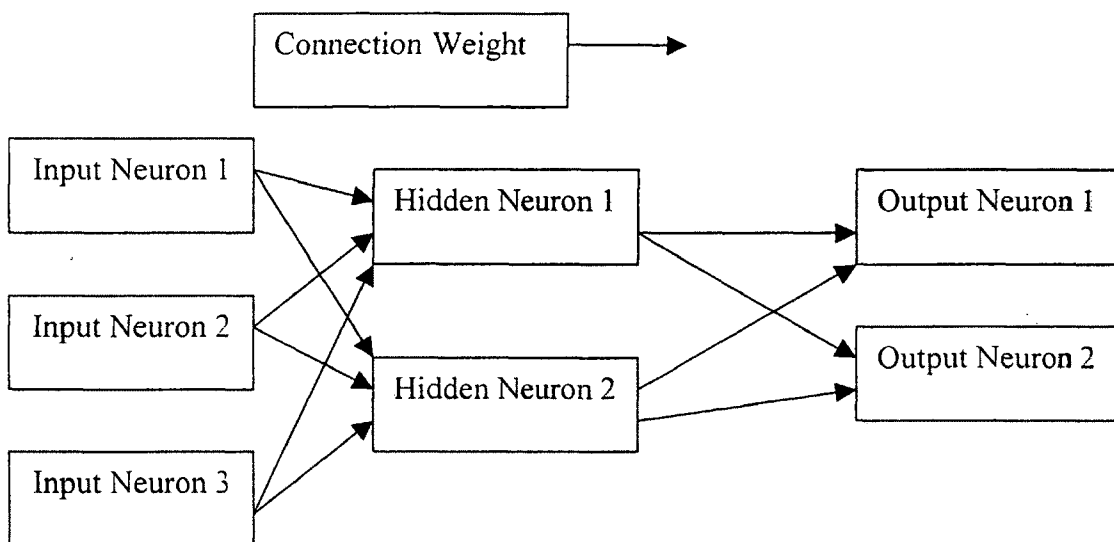


Figure 1 Layout of ANN

---

Training an ANN requires the modification of weights in the neural network using a learning algorithm, for example backpropagation. Backpropagation begins by constructing a network with the desired number of input, hidden and output units and initializing all network weights to small random values. Given this fixed network structure, the main loop of the algorithm then repeatedly iterates over the training examples. For each training example, it applies the network to the example, calculates the error of the network output for this example, computes the gradient with respect to the error on this example, and then updates all weights in the network. This gradient descent step is iterated until the network performs acceptably well. For more details on backpropagation, please refer to McClelland and Rumelhart (1986).

Like any learning algorithm, Backpropagation has some disadvantages. It has an extremely long training time, an offline-encoding requirement, and the hypotheses generated are hard to understand. Despite these deficiencies, Backpropagation has numerous advantages. It has the ability to acquire arbitrarily complex nonlinear mappings, it provides locally optimal predictions, it classifies future training examples fairly quickly, and, most importantly, it generalizes well for many types of domains. Because of its effective generalization ability, Opitz (1999) selected backpropagation for GEFS' component learning algorithm. Thus, we also include backpropagation in our study.

### **2.1.2 Naïve Bayesian**

Naïve Bayesian reasoning provides a probabilistic approach to learning. The naive Bayesian classification is based on Bayes's theorem. It is done by calculating,

- the probability of each class,

- the probability of each class occurring in combination with the training data, and
- the probability of that data, independent of the class.

A Bayesian classifier estimates probabilities based on the measured frequency of similar instances in the training example set. Bayesian learning has its limitations, however. The limitations arise because classification must often be made for examples comprised of more than just a few attributes and some combinations of attributes may be exceedingly rare, and the training set of examples is typically not adequately populated to provide a good estimate of the probability of each new instance. For example, if a particular combination of attributes never occurs in the training set, then a Bayesian classifier will return an estimated probability of zero for that instance.

The naïve Bayesian classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. This means that each attribute can be multiplied together to calculate the probability. The inductive learner will take the most probable outcome. Regardless of this assumption the naïve Bayesian classifier generalizes well in many circumstances. The approach used by the naïve Bayesian classifier is,

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

where,  $v_{NB}$  denotes the target value by the naïve Bayesian classifier,  $v_j$  is the target value,  $a_i$  is the input,  $P(v_j)$  is the probability of each class,  $P(a_i/v_j)$  is the probability of each class occurring in combination with the training data. In this thesis, we developed the naïve Bayesian classifier code found in Mitchell (1996); Chapter 6.9.



The main disadvantage of the naïve Bayesian classifier is, since the attributes are treated as though they were completely independent, the addition of redundant ones skews the learning process; however, naïve Bayesian has numerous advantages: a) it is simple; b) it has clear semantics for representing, using and learning probabilistic knowledge; c) it is fast to train and classify; and d) in many cases it outperforms more sophisticated learning methods. Given these advantages, we investigate in this thesis the utility of using naïve Bayesian as a learning algorithm for GEFS.

### 2.1.3 K-nearest neighbor

Nearest neighbor is an instance based learning method, where the training is simply storing the examples; generalizing beyond these training examples is postponed until a new instance has to be classified. With every new instance, the relationship to the previously stored training example is scrutinized to assign a target function value to the new instance. Instance based learning are sometimes referred to as a “lazy learner” because they delay generalizing until a new instance is given.

Nearest neighbor often uses the standard Euclidean distance (i.e., the straight-line distance between two points) to find the neighbors of an instance. The target function value for a new query is estimated from the known values of k nearest training examples. For example, Figure 2, shows the classification of the point “q” with five nearest neighbors. The point “q” will be classified as “-“ since three out of five neighbors are “-“. One refinement to the nearest neighbor algorithm is to weight the contribution of each of the neighbors according to their distance from the point, giving greater weight to the closer neighbors. For example, in Figure 2 point “q” may be classified as “+” because the

two “+” points are physically closer to it than the three “-“ points. We use the K-nearest neighbor algorithm given in Mitchell (1996), Chapter 8.2.

---

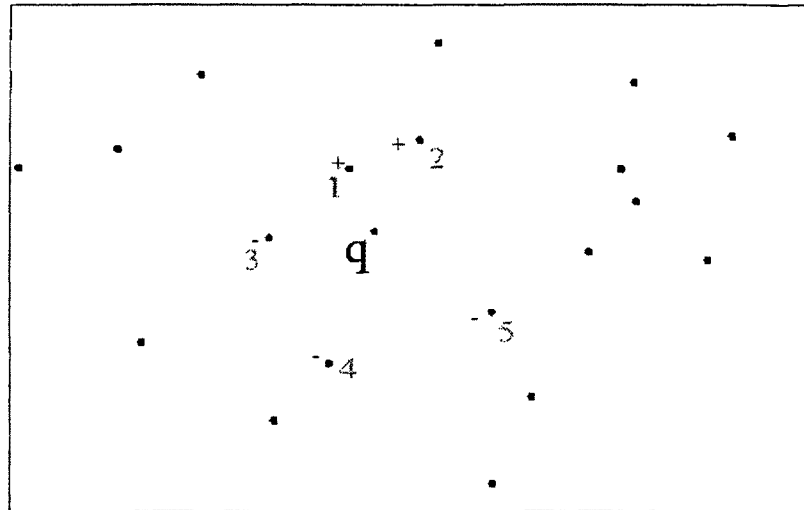


Figure 2 K-Nearest Neighbor,  $k = 5$

---

One disadvantage of distance-weighted K-nearest neighbor is that it is slow to classify new instances since it has to calculate the distance between the query point and all examples; however, K-nearest neighbor algorithms generalize well on many types of domains and are thus a logical choice upon which to include as a learning algorithm for GEFS (Dudani, S.A., 1976). One practical issue in applying the K-nearest neighbor algorithm is that the distance between instances is calculated based on all attributes of the instance (i.e., on all axes in the Euclidean space containing the instances). This lies in contrast to methods such as decision tree learning systems that select only a subset of the instance attributes when forming the hypothesis. By combining K-nearest neighbor with

feature selection, distance is measured only between estimated relevant attributes, thus helping to overcome this drawback.

#### 2.1.4 Probabilistic Neural Networks

PNNs are kernel-based approaches to probability density function (PDF) approximation. Estimating a PDF from data has a long statistical history (Parzen, 1962), and in this context fits into the area of Bayesian statistics. Conventional statistics can, given a known model, inform us of the chances of certain outcomes (e.g. we know that a unbiased die has a 1/6th chance of coming up with a six on any roll). Bayesian statistics turn this situation on its head, by estimating the validity of a model given certain data. More generally, Bayesian statistics can estimate the probability density of model parameters given the available data. To minimize error, we select the model whose parameters maximize this PDF.

In the context of a classification problem, if we can construct estimates of the classes PDF, we can compare the probabilities of the various classes, and select the most probable one. This is essentially what a neural network does during training - learn an approximation to the PDF. A more traditional approach is to construct an estimate of the PDF from the data. This is done by assuming a certain form for the PDF and then estimating the model parameters. A typical PDF assumption is normality where analytic techniques estimate the model parameters of the normal distribution (mean and standard deviation); however, the assumption of normality is often not justified.

An alternative approach to PDF estimation is *kernel-based approximation* (Parzen, 1962; Spekt, 1990; Spekt, 1991; Bishop, 1995; Patterson, 1996). We can reason loosely that the presence of particular cases indicate some probability density at

that point: a cluster of cases close together indicates an area of high probability density. Close to a case, we can have high level of confidence in the probability density, with a lesser and diminishing level as we move away. In kernel-based estimation, simple functions are located at each available case, and added together to estimate the overall PDF. Typically, the kernel functions are each Gaussians (bell-shapes). If sufficient training points are available, this will indeed yield an arbitrarily good approximation to the true PDF.

In the PNN, there are at least three layers: input, radial, and output layers. The radial units are copied directly from the training data, one per case. Each models a Gaussian function centered at a training case. There is one output unit per class. Each is connected to all the radial units belonging to its class, with zero connections from all other radial units. Hence, the output units simply add up the responses of the units belonging to their own class. The outputs are each proportional to the kernel-based estimates of the PDF of the various classes, and by normalizing these to sum to 1.0 estimates of class probability are produced. The only control factor that needs to be selected for PNN training is the window size (i.e. the radial deviation of the Gaussian functions). Refer to Ronald, Susan, and Andrew (1998) for more details on the PNN code.

Some drawbacks of the PNNs are that its training vectors must be stored and used to classify new vectors, requiring a lot of memory; and its computation time for a classification is proportional to the size of the training set. On the other hand, a PNN is a) fast to train, easy and typically requires only a few passes; b) it is resistant to noise; and c) its decision surfaces are **guaranteed** to approach the Bayesian optimal boundaries

as the number of training vectors grows. Given its computationally intense classification time, we use PNNs only on the smaller classification tasks.

## 2.2 Ensembles

Most modern machine learning research uses a single model or learning algorithm at a time, or at most selects one model from a set of candidate models. Recently however, there has been considerable interest in techniques (called ensembles) that integrate the collective predictions of a set of hypotheses in some principled fashion. Figure 3 shows the framework of predictor ensembles; each learning algorithm in the ensembles is trained using the training examples. Then for each example the prediction is the combination of all the individual predictions in the ensembles.

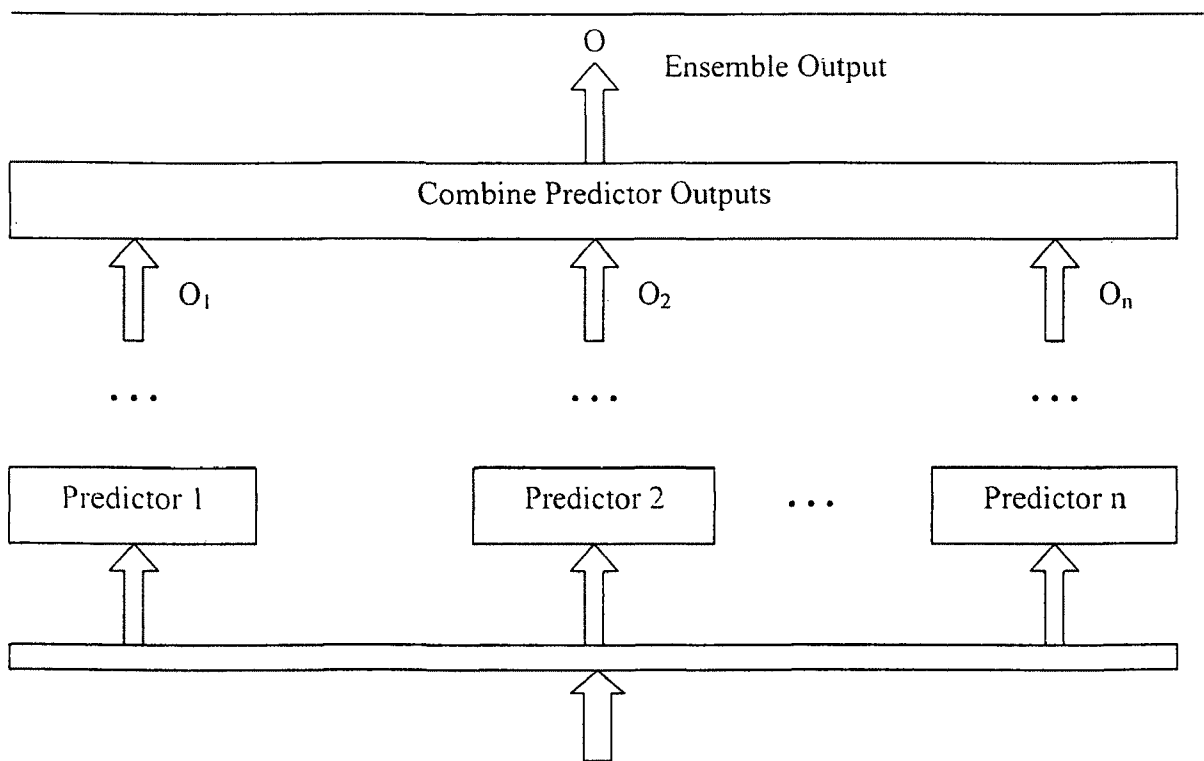


Figure 3 Ensembles Frame Work

---

Many researchers (Breiman 1996; Hansen and Salamon 1990; Opitz and Shavlik 1997) have demonstrated the effectiveness of combining schemes that are simply the weighted average of the predictions; this is the type of ensembles on which this thesis concentrates. Combining the output of several hypotheses is useful only if there is disagreement on some inputs. Obviously combining the same hypotheses is not useful. An ideal ensemble consists of highly correct hypotheses that disagree on their predictions. Numerous authors have empirically verified that such ensembles generalize well (e.g., Opitz and Shavlik 1996; Breiman 1996a; Freund 1996). Hence methods for creating effective ensembles center around producing hypotheses that disagree on their predictions. For example, creating neural networks trained with differing topologies often results in hypotheses that disagree. In this thesis, hypotheses that disagree on their predictions are created using genetic algorithms to vary the inputs given to the learners thus creating a diverse population.

### **2.3 Feature Selection**

The aim of feature selection is to choose a subset of features that improve generalization accuracy. Attempting to select the minimally sized subset of features such that the classification accuracy does not significantly decrease often does this. Algorithms that perform feature selection as a pre-processing step, prior to learning, can generally be placed into one of two broad categories. One approach, referred to as the “filter” approach (John, Kohavi, and Pflieger, 1994) operates independently of any learning algorithm. Undesirable features are filtered out of the data before induction commences. Another approach referred to as the “wrapper” approach (John, Kohavi and

Pfleger, 1994) employs as a subroutine, a statistical re-sampling technique (such as cross validation) using the actual target-learning algorithm to estimate the accuracy of feature subsets. Kohavi and John (1997) showed that the efficacy of a set of features depends on the algorithm itself and thus the ideal feature subset for one algorithm often differs from the ideal feature subset of another algorithm. This approach has proven more accurate (Liu, H., and Setiono, R., 1996), and is thus the method of choice for this thesis. Figure 4 illustrates the wrapper approach,

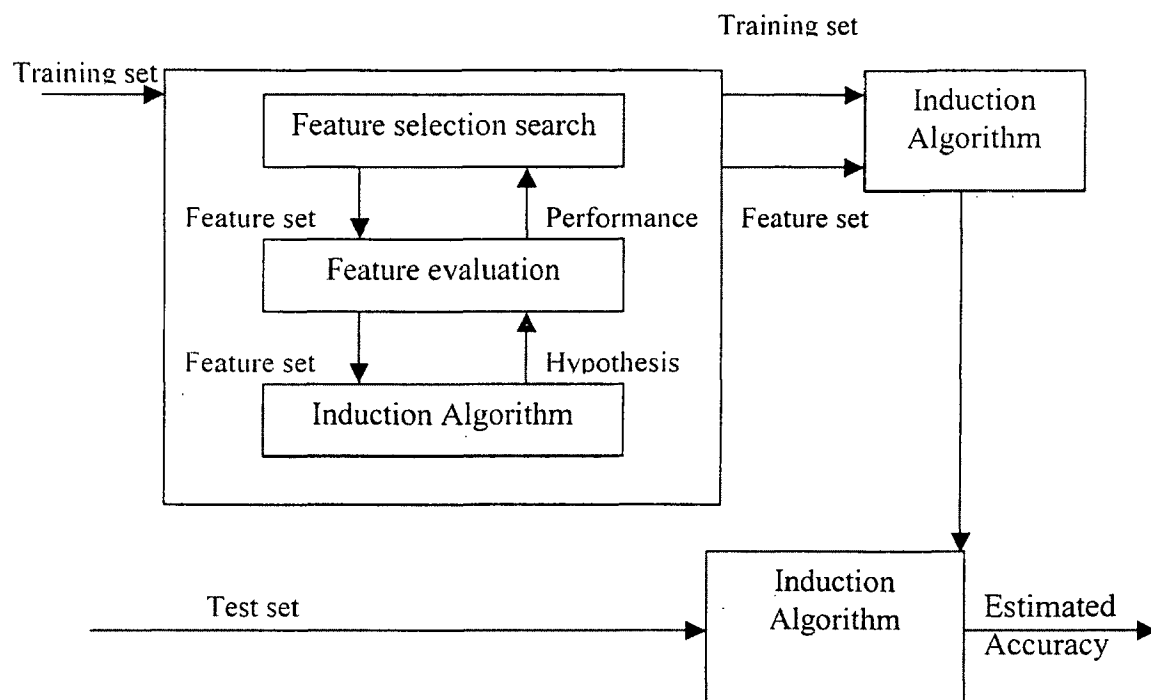


Figure 4 Wrapper Approach to find featured set

If there are “ $n$ ” possible features initially, then there are “ $n^2$ ” possible subsets. Ideally, feature selection methods search through the subsets of features, and try to find the best one among the competing “ $n^2$ ” candidate subsets. The only way to find the best subset would be to try them all – this is clearly prohibitive for all but a small number of

initial features. Many researchers apply heuristic search strategies such as hill climbing and best first search (Rich, Knight, 1991) to search the feature subset space. The best first search algorithm starts with an empty set of features and generates all possible single feature expansions. The disadvantage of the best first search is that it takes a long time to find the best subset.

Another approach is to use generic heuristics (primarily hill climbing algorithm) when domain knowledge is costly to exploit or unavailable (Kittler, J. 1978). In this thesis, we present a GA approach to do the generic heuristics search.

## **2.4 Genetic Algorithms**

GAs are optimization and search procedures inspired by genetics and the process of natural selection. The search for the best hypothesis starts with a population (i.e., a collection of initial hypotheses). The initial hypotheses then gives rise to new hypothesis by means of genetic operations like mutation and crossover on the better of the currently known hypotheses. The motivations behind GAs are:

- evolution is known for its success,
- they can do effective global search optimization (Holland 1975; Mitchell 1996),  
and
- they are easily parallelized and can take advantage of powerful computer hardware.

The generation of successors in GAs is determined by the set of operators that recombine and mutate selected members of the current population. The fitness function defines the criteria for ranking the hypotheses and selecting them for inclusion into a



steady-state (fixed size) population. The fitness function determines which of the population member's genetic material is to be passed onto descendants.

The GA offers an attractive approach to solving the feature subset selection problem in inductive learning (J. Yang and V. Honavar, 1997, H.Vafaie and K.Jong, 1997). Many researchers have used GAs to do the feature selection (J. Yang and V. Honavar, 1997; Leardi, R, 1994; Rainer Stotzka et al. 2000) but not for ensembles. The task of ensemble feature selection has an additional goal of finding a set of feature subsets that will promote disagreement among the component members of the ensemble. This search space is enormous for any non-trivial problem. In the thesis, the GAs are used as search mechanism for the problem of feature selection for ensembles. GAs are logical choice since they have shown to be effective global optimization techniques (Holland 1975; Mitchell 1996).

### 3 Genetic Ensemble Feature Selection

The GEFS algorithm (Opitz, 1999) uses the advantages of both feature selection and ensembles to create an effective set of classifier neural networks trained by backpropagation. The objective of this thesis is to extend the GEFS algorithm to additional learning algorithms and apply it to the regression problems as well. Ensemble performance is based on producing a set of predictors that disagree on their predictions. In feature selection, the objective is to select an effective subset features upon which to learn. By having an ensemble of feature subsets, we can have predictors that disagree amongst themselves. If these predictors have different parameters then each will have their own set of feature sets leading us to have a diverse set of predictors. We use GAs to search the universal space of feature subsets.

#### 3.1 The GEFS Algorithm

The GEFS algorithm (Table 1) uses a GAs to generate a set of predictors that are accurate and diverse in their predictions. GEFS starts by creating and training an initial population of predictors. The representation of each individual is simply a dynamic length string of integers that maps to a feature. GEFS creates the initial population by randomly varying the input features and training the algorithm using these features. GEFS creates other predictors by using the genetic operators of crossover and mutation.

GEFS trains each of these predictors with the training examples and calculates their fitness with respect to their prediction accuracy and diversity. GEFS defines fitness of each predictor as:

$$\text{Fitness} = \text{Accuracy} + \lambda \text{ Diversity}$$

where,  $\lambda$  defines the tradeoff between the accuracy and diversity. We define accuracy as 1.0 minus the absolute value of the difference between the target and the predicted output. We define diversity to be the average squared difference between the prediction of a component classifier and the prediction of the ensemble.

#### GEFS (Population\_size)

*Population\_size*: number of predictors in each population.

- *Initialize Population*: Generate *Population\_size* predictors at random using varying inputs and train these predictors
- *Until stopping criteria is reached*
  - Use one of the genetic operators to create a new predictor.
  - Measure the diversity of the predictor with respect to the current population.
  - Normalize the accuracy scores and the diversity scores of the predictor.
  - Calculate the fitness of each predictor in the population.
  - Prune the population to the *Population\_size* fittest predictors.
  - Adjust  $\lambda$ .
  - The current population composes the ensemble.

**Table 1 Genetic Ensemble Feature Selection**

Finally, GEFS deletes the predictor with the worst fitness in the population to the  $N$  most-fit members, then repeats this process until a stopping criterion is met, thus maintaining a steady state population. At any point in time the ensemble output is the mean average output of each member of the current population. As the population evolves so does the ensemble. We define accuracy to be predictors  $i$ 's accuracy. We define diversity to be the average difference between the prediction of a component classifier and the ensemble. Normalizing both the diversity and accuracy, each between 0.0 and 1.0, allows  $\lambda$  having the same meaning across domains. The value of  $\lambda$  is adjusted based on the discrete derivatives of the ensemble error  $\hat{E}$ , the average population error  $\bar{E}$ , and the average diversity  $\bar{D}$  within the ensemble. First, we never change  $\lambda$  if  $\bar{E}$  is decreasing; otherwise we (a) increase  $\lambda$  if  $\bar{E}$  is decreasing and the population diversity  $\bar{D}$  is decreasing; or we (b) decrease  $\lambda$ , if  $\bar{E}$  is increasing and  $\bar{D}$  is not decreasing.  $\lambda$  starts at 1.0 and changes 10% of its current value. We create the initial population by randomly choosing the number of features to include in each feature subset. For classifier  $i$ , the size of each feature subset ( $N_i$ ) is uniformly chosen randomly between 1 and twice the size of original feature set. From the original feature subset, each feature is randomly selected with replacement. Thus some features can be selected more than once whereas others may not be selected. Replication of a feature is useful in two cases, (a) it allows the predictor more chance to utilize that feature, and (b) it increases the probability of that feature's existence in future generations.

### **3.2 Genetic Operators**

The two genetic operations that generate new populations are crossover and mutation. In this thesis we use a dynamic length uniform crossover. We place each feature in the parents in one of the two children. By doing this, both resulting children may be shorter than the shortest parent or longer than the longest parent. Only one of the two children will be considered for inclusion into the population. The mutation operator works much like the traditional mutation operator; we randomly replace a small percentage of the features with some other feature. This allows features that were not selected during the initialization phase to occur in the later populations. Either crossover or mutation is done at each step, but not both as is typical for many GAs. If the newly created predictor has higher fitness than the predictor with least fitness, it will replace that one. With both operators, the network is trained from scratch using the new feature subset.

### **3.3 Extensions to GEFS**

The original GEFS algorithm (Opitz 1999) was implemented for classification problem using backpropagation. The objective of this thesis is to extend GEFS to use the inductive learning algorithms of naïve Bayesian classifier, K-nearest neighbor and PNNs and apply these algorithms to a wide variety of problem domains. Opitz (1999) used roulette wheel fitness proportional reproduction; we modify GEFS to use random selection. However, selection based on fitness is done in the step of the algorithm where the least fit individual is removed from the population when keeping a steady state in population size.

## 4 Methodology

To evaluate the performance of the GEFS, we obtained a number of datasets from the University of Wisconsin Machine Learning repository, the University of California, Irvine dataset repository, the Naval Research Laboratory and Natural Resources Research Institute at the University of Minnesota. These data sets are selected since they a) came from real world problems, b) are used by many researchers, and c) have varied characteristics. Table 2 provides a summary of the classification datasets. The last two columns give the number of hidden units and epochs used for backpropagation.

**Table 2 Summary of the classification datasets**

Dataset	Cases	Classes	Inputs	Outputs	Hidden	Epochs
breast	699	2	9	9	5	20
credit-a	690	2	47	1	10	35
credit-g	1000	2	63	1	10	30
diabetes	768	2	8	1	5	30
glass	214	6	9	6	5	80
heart-cleveland	303	2	13	1	5	40
hepatitis	155	2	32	1	10	60
house-votes-84	435	2	16	1	5	40
hypo	3772	5	55	5	15	40
ionosphere	351	2	34	1	10	40
iris	159	3	4	3	5	80
kr-vs-kp	3196	2	74	1	15	20
labor	57	2	29	1	10	80
promoters-936	936	2	228	1	20	30
ribosome-bind	1877	2	196	1	20	35
satellite	6435	6	36	6	15	30
segmentation	2310	7	19	7	15	20
sick	3772	2	55	1	10	40
sonar	208	2	60	1	10	60
soybean	683	19	134	19	25	40
splice	3190	3	240	3	25	60
vehicle	846	4	18	4	10	40
BBR1	87	2	6	1	5	30
BBR2	243	2	7	1	5	30
Buckley	392	2	8	1	5	30

Table 3 shows the list of the regression datasets used in this thesis. Regression datasets are datasets that have real-valued outputs, whereas classification datasets have outputs that fall into one of N discrete categories. Opitz (1999) studied only classification tasks. The regression datasets are collected from the Natural Resources Research Institution, at the University of Minnesota, Duluth. Quantitative structure-activity relationships (QSAR) represent an attempt to correlate structural or property descriptors of compounds with activities. These datasets are QSAR theoretical descriptors for finding the toxicity of various chemicals. Table 4 gives the parameters associated with each of these datasets.

**Table 3 Regression Datasets**

<i>Lr25</i> is the hierarchical QSAR approach for estimating toxicity of 54 phenols.
<i>Lr47</i> is the hierarchical QSAR study of fish bio-concentration factors (BCF's) of 87 organic pollutants from the molecular connectivity model.
<i>Lr50</i> is the hierarchical QSAR study of estimating toxicity of 34 benzonitriles to the ciliate tetrahymena pyriformis.
<i>Lr76</i> is the hierarchical QSAR study of estimating toxicity of 91 benzothiazolium salts against euglena gracilis using the Free-Wilson approach
<i>Lr78</i> is the hierarchical QSAR approach for response-surface analyses for toxicity to tetrahymena pyriformis on 56 reactive carbonyl-containing aliphatic chemicals.
<i>Hall</i> is the hierarchical QSAR study of the toxicity of 69 benzene derivatives.

Each dataset listed in Table 3 has four different level of complexity. Each complexity represents different indices describing the activity of a chemical. QSARs

have come into widespread use for the prediction of various molecular properties, as well as biological, physiochemical, pharmacological, and toxicological responses. In this thesis we have focused on the role of four distinct sets of theoretical descriptors: topostructural, topochemical, geometric and quantum chemical indices.

The topostructural (TSI) and topochemical (TCI) indices fall into the category normally considered topological indices. TSIs are topological indices that only encode information about the adjacency and distance of atoms (vertices) in molecular structures (graphs), irrespective of the chemical nature of the atoms involved in bonding or factors such as hybridization. TCIs are parameters that quantify information regarding the topology (connectivity of atoms), as well as specific chemical properties of the atoms comprising molecule. These indices are derived from weighted molecular graphs where each vertex (atom) or edge (bond) is properly weighted with selected chemical or physical property information. The geometrical indices (3D) are three-dimensional Wiener numbers for hydrogen filled molecular structure, hydrogen-suppressed molecular structure, and van der Waals volume. The quantum chemical indices (QC) are energy of the highest occupied molecular orbital, energy of the second highest occupied molecular orbital, energy of the lowest unoccupied molecular orbital, energy of the second lowest unoccupied molecular orbital, heat of formation, and dipole moment.

**Table 4 Four Levels of complexity**

Dataset	TSI	TCI	3D	QC	Total
Lr25	37	59	3	6	105
Lr47	39	63	3	6	111
Lr50	34	53	3	6	96
Lr76	35	55	3	6	99
Lr78	33	51	3	6	93
Hall	35	51	3	6	95



Table 5 illustrates the property of inputs for each dataset. It lists how many of the features are discrete and how many are continuous. This is important for analyzing the utility of any approach to feature selection.

**Table 5 Properties of the Datasets**

Dataset	Features	
	Continuous	Discrete
credit-a	6	9
credit-g	7	13
diabetes	9	-
glass	9	-
heart-cleveland	8	5
hepatitis	6	13
house-votes-84	-	16
hypo	7	22
ionosphere	34	-
iris	4	-
kr-vs-kp	-	36
labor	8	8
promoters-936	-	57
ribosome-bind	-	49
satellite	36	-
segmentation	19	-
sick	7	22
sonar	60	-
soybean	-	35
vehicle	18	-
BBR1	6	-
BBR2	7	-
Buckley	8	-
Lr25	105	-
Lr47	111	-
Lr50	96	-
Lr76	99	-
Lr78	93	-
Hall	95	-

Cross-validation is a method for estimating generalization error based on "re-sampling" (Weiss and Kulikowski 1991; Efron and Tibshirani 1993; Hjorth 1994;

Plutowski, Sakata, and White 1994; Shao and Tu 1995). In n-fold cross-validation, we divide the data into “n” subsets of (approximately) equal size. We train the learner “n” times, each time leaving out one of the subsets from training, but using only the omitted subset to compute accuracy. If “n” equals the sample size, this is called "leave-one-out" cross-validation.. This thesis uses 10-fold cross-validation to estimate the generalization error for the classification datasets, while the regression tasks use leave-one-out.

In this thesis, we use the number of predictors in the population as 20 (for a total of 200 predictors for each 10-fold cross validation). The mutation rate for altering a feature is 1.5%. The crossover rate is 50% (i.e., half the time we do crossover, half the time mutation); the high mutation likelihood is picked to: a) have a diverse and accurate population so that the ensemble accuracy will increase, and b) include new features to the population. The value of  $\lambda$  is initialized to be 1.0. The maximum number of predictors is set to 100 (the stopping criteria for GEFS). The inputs and outputs used for the learning algorithms are listed in Table 2. Parameter setting for the neural network include a learning rate of 0.15, a momentum of 0.9 and weights are initialized randomly to be between  $-0.5$  to  $0.5$ . For the K-nearest neighbor the k value is set as 7. For PNN the window size is set as 25.

This thesis uses two metrics to measure the efficacy of GEFS on the regression tasks: a) Standard error, which is a measure of the amount of error in the prediction of y for an individual x, and b) the Pearson product moment correlation, which reflects the extent of a linear relationship between the two data sets. These two metrics estimate how far the predicted target value varies from the actual value. The metrics are as follows:

$$\text{Standard Error} = \sqrt{\left[ \frac{1}{n(n-2)} \right] \left[ n \sum y^2 - (\sum y)^2 - \frac{[n \sum xy - (\sum x)(\sum y)]^2}{n \sum x^2 - (\sum x)^2} \right]}$$

and

$$\text{Pearson Product moment (R)} = \frac{n(\sum xy - (\sum x)(\sum y))}{\sqrt{[n \sum x^2 - (\sum x)^2] * [n \sum y^2 - (\sum y)^2]}}$$

where x, y are the range of dependent and independent data points.

## 5 Results

The results from the two types of problem domains, classification and regression, are discussed in the following two sections.

### 5.1 Classification Tasks

We trained and tested GEFS using naïve Bayesian, K-nearest neighbor, and backpropagation classifiers on the classification domains shown in Table 2 in section 4. We tested PNNs only for small datasets that have only two classes as its output (due to code limitations beyond our control). Each table in this section contains a win-loss-tie comparison between the Single learner, GEFS initial population and after generating 100 hypotheses.

Table 6 thru Table 9 show the test set errors for:

- (1) a Single traditional naïve Bayesian learner,
- (2) a Single traditional K-nearest neighbor learner,
- (3) a Single traditional backpropagation learner,
- (4) a Single traditional PNN learner,
- (5) the ensemble of GEFS initial population (for all four learners), and
- (6) GEFS run to consider 100 hypotheses (for all four learners).

From Table 6, we can see that naïve Bayesian classifier does poorly when the initial population is considered, however, as the search continues GEFS is able to increase accuracy. In addition, the accuracy of naïve Bayesian classifier is better for those datasets having more discrete features than continuous features. GEFS loses to the single naïve Bayesian classifier in **only** three of the twenty-five datasets. In addition only

four times does the initial population have better accuracy than after searching 100 hypotheses.

**Table 6 Error Rates for the naïve Bayesian**

Dataset	Single Learner	Initial Pop	100 hypotheses
Breast	7.9	22.1	3.43
credit-a	14.4	12.6	12.6
credit-g	30.0	24.2	24.3
Glass	33.3	35.0	42.0
heart-cleveland	20.6	15.8	15.8
hepatitis	20.3	17.4	16.1
house-votes-84	6.74	14.0	5.51
hypo	7.41	9.23	7.38
ionosphere	15.3	14.5	14.2
iris	5.23	21.3	4.66
kr-vs-kp	2.0	4.59	4.32
Labor	17.5	21.0	9.82
pima	34.5	39.3	24.7
promoters-936	5.44	5.44	4.48
ribosome-bind	6.53	7.72	5.59
satellite	20.1	18.3	15.5
segmentation	4.3	4.1	3.09
sick	6.12	7.13	5.23
splice	20.4	17.3	16.4
sonar	14.2	15.3	13.7
soybean	12.7	12.6	11.9
Vehicle	25.1	27.4	27.1
BBR1	3.44	8.04	17.2
BBR2	4.90	0.82	1.06
Buckley	0.765	10.2	0.20
Single Learner		14-11-0	3-22-0
Initial Pop			4-20-1

Table 7 shows that K-nearest neighbor reacts similarly to naïve Bayesian; K-nearest neighbor does poorly when the initial population is considered, however as the search continues we are able to get better accuracy. Only for four of the twenty-five dataset does GEFS lose to the single K-nearest neighbor classifier. In addition only once is the initial population more accurate than after searching 100 hypotheses.

**Table 7 Error Rates for the K-nearest neighbor**

Dataset	Single Learner	Initial Pop	100 hypotheses
Breast	3.00	3.14	2.71
credit-a	13.3	11.7	11.3
credit-g	26.6	25.9	25.4
Glass	33.6	24.7	19.1
heart-cleveland	19.4	16.5	16.5
Hepatitis	17.4	18.7	18.0
house-votes-84	6.43	8.96	5.05
Hypo	6.73	8.70	5.8
ionosphere	15.6	18.2	13.3
Iris	4.66	21.3	4.66
kr-vs-kp	3.50	6.89	3.12
Labor	19.2	31.5	10.5
Pima	26.0	25.1	24.8
Promoters-936	10.4	5.66	5.44
ribosome-bind	12.5	7.72	7.51
Satellite	9.33	9.40	9.54
segmentation	4.6	4.19	4.02
Sick	3.7	3.65	3.21
Splice	38.6	16.48	15.29
Sonar	16.3	22.5	9.61
Soybean	11.5	13.3	12.0
Vehicle	29.9	27.4	22.6
BBR1	9.19	25.2	8.04
BBR2	0.41	23.0	1.64
Buckley	0.255	20.4	0.0
Single Learner		15-10-0	4-20-1
Initial Pop			1-23-1

Table 8 shows the results for backpropagation. Note that backpropagation creates a more accurate initial population than either naïve Bayesian and K-nearest neighbor. In addition, GEFS is able to further improve accuracy as it continues its search (except in four cases). In fact, GEFS is able to improve generalization accuracy in all 25 domains over the single ANN.

**Table 8 Error Rates for the backpropagation**

Dataset	Single Learner	Initial Pop	100 hypotheses
Breast	3.6	3.42	3.28
credit-a	14.8	13.6	13.1
credit-g	27.9	25.1	25.1
Glass	48.6	41.8	38.1
heart-cleveland	18.6	15.4	15.1
Hepatitis	20.1	16.6	14.7
house-votes-84	4.9	4.35	3.44
Hypo	7.7	7.4	5.98
Ionosphere	17.1	15.6	15
Iris	16	6	4
kr-vs-kp	2.32	3.84	2.31
Labor	6.1	6.89	3.44
Pima	26.4	22.6	22.7
promoters-936	5.3	5.01	5.22
ribosome-bind	9.3	8.78	8.41
Satellite	18.6	17.2	17
Segmentation	17.7	8.87	7.57
Sick	5.9	6.17	4.24
Splice	4.3	4.04	4.13
Sonar	24	21	18.1
Soybean	9.2	7.3	6.14
Vehicle	38.3	27.3	24
BBR1	17.2	10.2	11.3
BBR2	8.6	1.63	0.0
Buckley	3.6	0.25	0.25
Single Learner		3-22-0	0-25-0
Initial Pop			4-20-1

Table 9 gives the results when using PNN as the component learning algorithm.

Like K-nearest neighbor and naïve Bayesian, PNN does poorly on the initial population but as the search continues, GEFS is able to obtain improved accuracy. For only one of the thirteen dataset does GEFS lose to the single PNN. In addition only once is the initial population more accurate than after searching 100 hypotheses

**Table 9 Error Rates for PNN**

Dataset	Single Learner	Initial Pop	100 Predators
Breast	35.05	37.62	37.91
credit-a	44.49	15.07	14.6
credit-g	30	29.6	22.3
heart-cleveland	55.77	53.79	49.17
hepatitis	34.1	56.77	16.12
house-votes-84	54.7	54.25	54.25
ionosphere	36.18	36.18	36.18
kr-vs-kp	52.3	33.44	22.46
promoters	24.4	10.63	3.19
Sonar	55.2	14.9	14.9
BBR1	55.17	59.77	58.62
BBR2	21.8	25.9	11.11
Buckley	31.1	15.3	7.4
Single Learner		4-8-1	2-10-1
Initial Pop.			1-11-1

To better analyze Table 6 to Table 9 results, Figure 5 to Figure 8 plot the percent reduction in error GEFS obtains with both the initial population and after generating 100 hypotheses when compared to their component classifiers. In all figures, the X-axis refers to the difference in the percentage reduction in error and the Y-axis refers to the classification datasets from Table 2 in Section 4. In each of the figures, the bar extending to the positive side of X-axis indicates GEFS has a higher accuracy than the traditional component learning algorithm. Conversely, the bar extending to the negative side of X-axis shows that the traditional learning algorithm has more accuracy. Examining these figures we note that for most cases, GEFS significantly improves the accuracy of the component learning algorithm. This appears especially true for both backpropagation and PNNs.



### Traditional Bayes vs GEFS

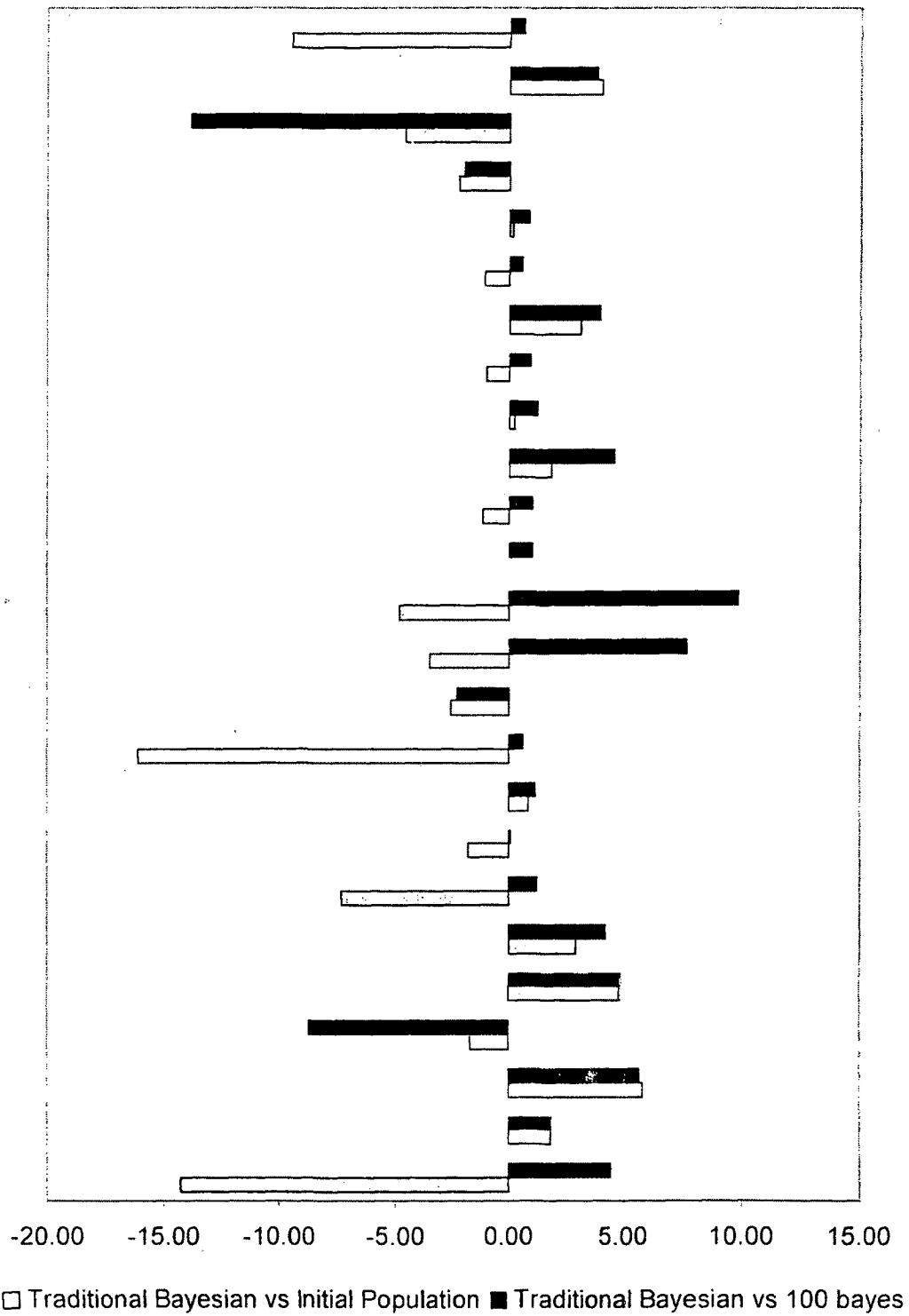


Figure 5 Error Rates for naïve Bayesian classifier

### Traditional K-Nearest Neighbor vs GEFS

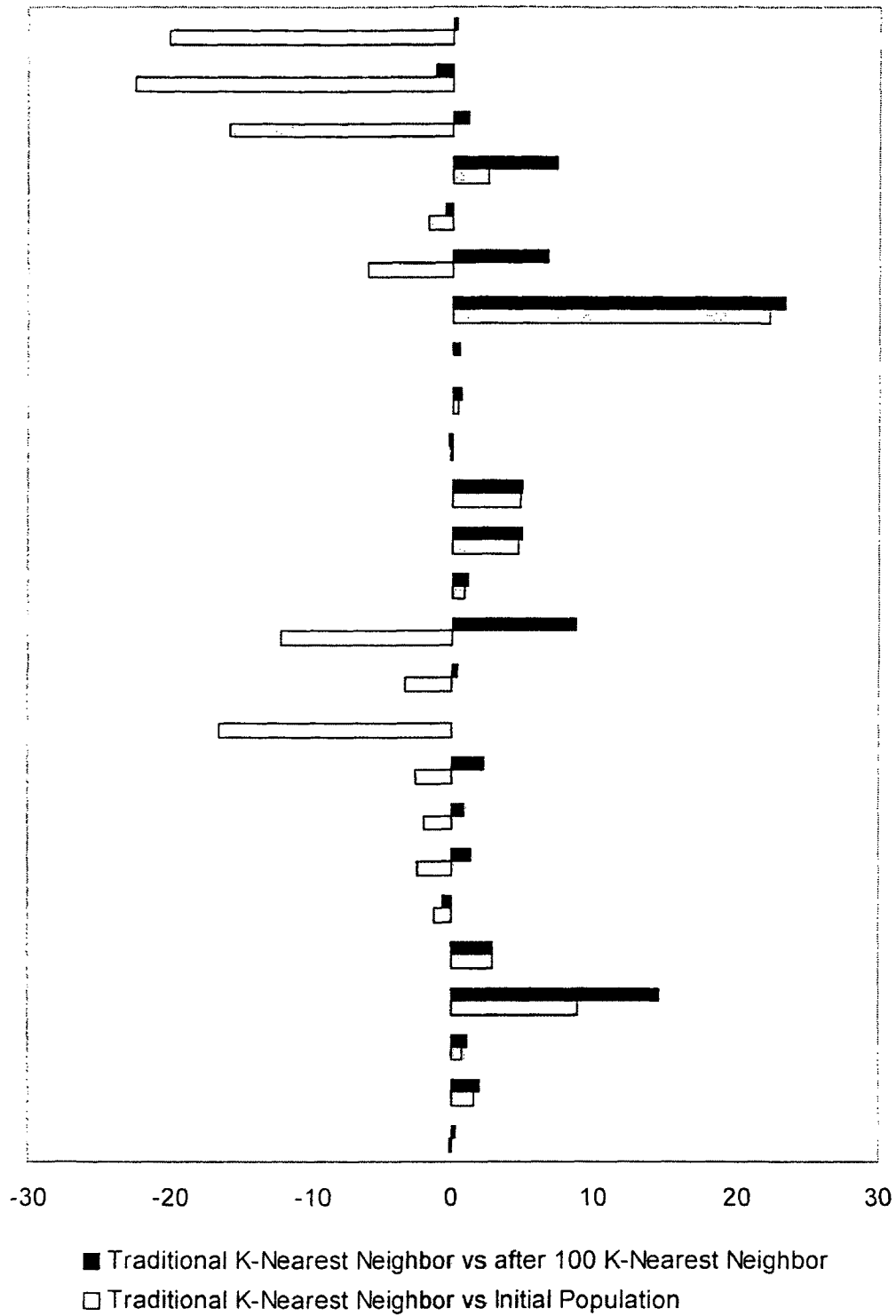


Figure 6 Error Rates for K-nearest neighbor

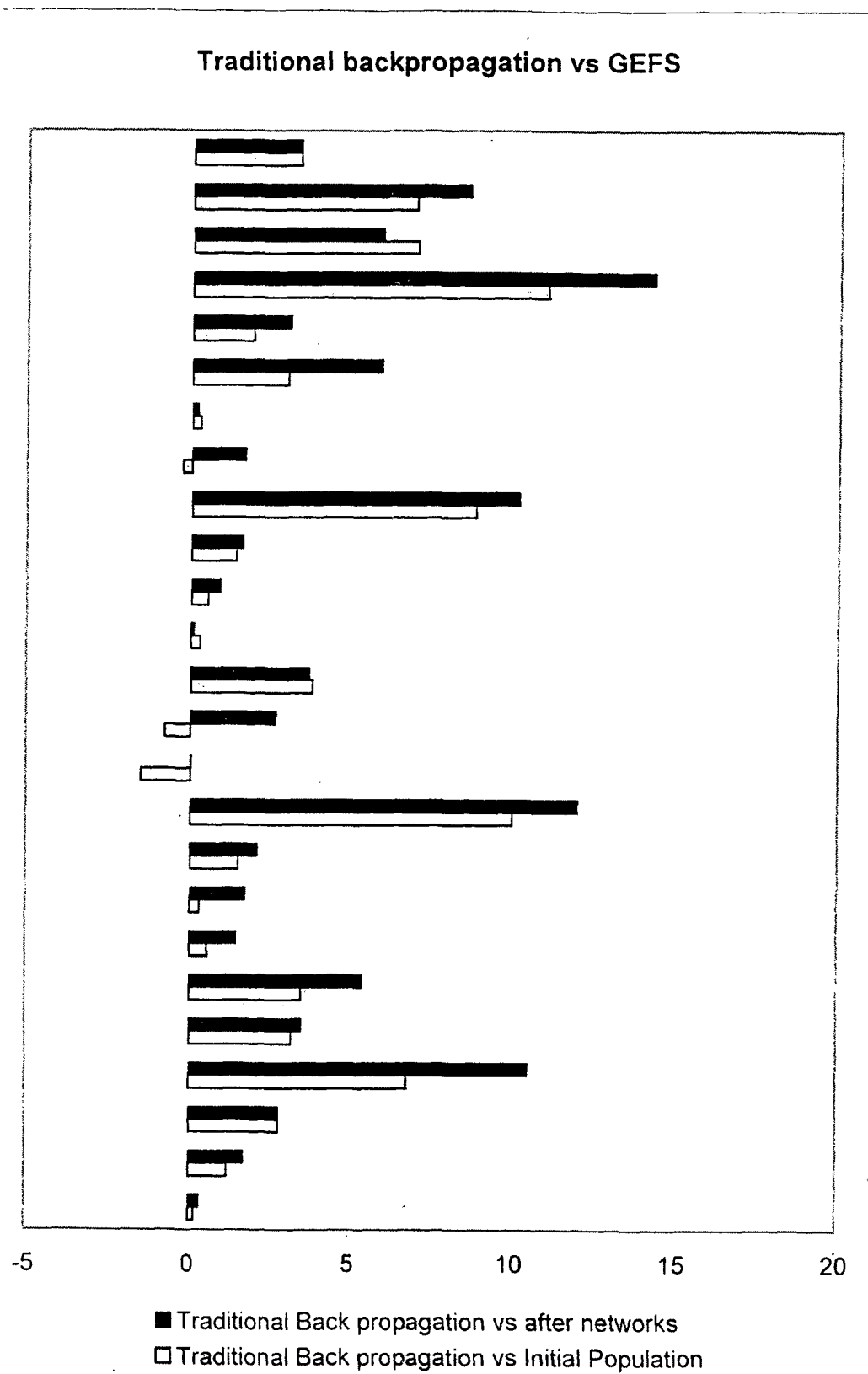


Figure 7 Error Rates for backpropagation

### Traditional PNN vs GEFS



Figure 8 Error Rates for PNN

For convenience, Table 10 shows the test error rates for all four learning algorithms after running GEFS for 100 hypotheses. From Table 10, we note that K-nearest neighbor and backpropagation appear to achieve higher accuracy than naïve Bayesian and PNNs; however, the proper learning choice is still task dependant.

**Table 10 Test Error of all learning algorithms**

Dataset	Naïve Bayesian	K-nearest neighbor	Backpropagation	PNN
breast	3.43	2.71	3.28	37.91
credit-a	12.6	11.3	13.1	14.6
credit-g	24.3	25.4	25.1	22.3
glass	42	19.1	38.1	
heart-cleveland	15.8	16.5	15.1	49.17
hepatitis	16.1	18	14.7	16.12
house-votes-84	5.51	5.05	3.44	54.25
hypo	7.38	5.8	5.98	
ionosphere	14.2	13.3	15	36.18
iris	4.66	4.66	4	
kr-vs-kp	4.32	3.12	2.31	22.46
labor	9.82	10.5	3.44	
pima	24.7	24.8	22.7	
promoters-936	4.48	5.44	5.22	3.19
ribosome-bind	5.59	7.51	8.41	
satellite	15.5	9.54	17	
segmentation	3.09	4.02	7.57	
sick	5.23	3.21	4.24	
splice	16.4	15.29	4.13	
sonar	13.7	9.61	18.1	14.9
soybean	11.9	12	6.14	
Vehicle	27.1	22.6	24	
BBR1	17.2	8.04	11.3	58.62
BBR2	1.06	1.64	0	11.11
Buckley	0.2	0	0.25	7.4
Naïve Bayesian		10-14-1	9-16-0	10-2-1
K-nearest neighbor	14-10-1		13-12-0	10-3-0
Backpropagation	16-9-0	12-13-0		10-3-0
PNN	2-10-1	3-10-0	3-10-0	

## 5.2 Regression Tasks

Table 11 shows the correlation coefficient for the backpropagation algorithms on all the regression datasets. The correlation coefficients are calculated for (1) a Single neural network (Single Learner); (2) the GEFS initial population, and (3) GEFS after 100 networks. The win-loss-tie results are as before, the row wins first, the column wins second, and ties third. For instance, GEFS considering 100 hypotheses beats the single neural network 20 out of 24 times in the aggregated results. From Table 11, we conclude the same thing as the classification tasks; the initial population produces good accuracy and as we run GEFS longer, this accuracy increases.

Figure 9 shows the comparisons between GEFS and Single Learner. Figure 9 graphs the *difference* in correlation coefficient between the Single backpropagation and GEFS initial population and after generating 100 hypotheses.

**Table 11 Correlation coefficient for backpropagation**

Backpropagation	Single Learner		GEFS			
			Initial Population		100 Networks	
	R <sup>2</sup>	Std. Error	R <sup>2</sup>	Std. Error	R <sup>2</sup>	Std. Error
<b>Lr25</b>						
TSI	0.4732	<b>0.6455</b>	0.4678	0.6451	0.4873	0.6242
TSI+TCI	0.7804	<b>0.4487</b>	0.7830	0.4025	0.7843	<b>0.4025</b>
TSI+TCI+Geometric	0.7633	<b>0.4600</b>	0.7864	0.4057	0.8001	<b>0.3887</b>
TSI+TCI+Geometric+QC	0.7388	<b>0.4813</b>	0.7999	0.3887	0.8126	0.3746
<b>Lr47</b>						
TSI	0.6959	<b>0.5399</b>	0.7009	0.5314	0.6911	<b>0.8800</b>
TSI+TCI	0.7653	<b>0.4843</b>	0.7738	0.4613	0.7745	<b>0.4555</b>
TSI+TCI+Geometric	0.7390	<b>0.5040</b>	0.7711	0.4577	0.7833	<b>0.4472</b>
TSI+TCI+Geometric+QC	0.7617	<b>0.4884</b>	0.7511	0.4751	0.7613	0.4706

Lr50						
TSI	0.1928	0.7883	0.0653	0.9391	0.1296	0.8538
TSI+TCI	0.3015	0.7180	0.3868	0.6614	0.3927	0.6661
TSI+TCI+Geometric	0.2928	0.7216	0.5141	0.5921	0.5745	0.5560
TSI+TCI+Geometric+QC	0.2037	0.7979	0.2809	0.7326	0.4270	0.6403
Lr76						
TSI	0.0005	0.7530	0.0187	0.7511	0.0493	0.7295
TSI+TCI	0.5354	0.5184	0.8159	0.3156	0.8205	0.3905
TSI+TCI+Geometric	0.4896	0.5409	0.8246	0.3068	0.8118	0.3202
TSI+TCI+Geometric+QC	0.7446	0.3992	0.7613	0.4706	0.7999	0.3289
Lr78						
TSI	0.2610	0.7425	0.2735	0.7251	0.3404	0.6991
TSI+TCI	0.5840	0.5546	0.6661	0.4796	0.6712	0.5086
TSI+TCI+Geometric	0.6647	0.5119	0.6517	0.5039	0.6713	0.5078
TSI+TCI+Geometric+QC	0.6235	0.5137	0.6467	0.4955	0.6620	0.5130
Hall						
TSI	0.5073	0.5571	0.5956	0.4776	0.5994	0.4790
TSI+TCI	0.6887	0.4568	0.6670	0.4611	0.6953	0.45112
TSI+TCI+Geometric	0.7001	0.4503	0.6578	0.4586	0.6845	0.4344
TSI+TCI+Geometric+QC	0.7564	0.4103	0.6754	0.4400	0.7000	0.4233
Single Learner						
TSI			2-4-0		2-4-0	
TSI+TCI			1-5-0		0-6-0	
TSI+TCI+Geometric			2-4-0		1-5-0	
TSI+TCI+Geometric+QC			2-4-0		1-5-0	
Aggregated Result			7-17-0		4-20-0	
Initial Population						
TSI					1-5-0	
TSI+TCI					0-6-0	
TSI+TCI+Geometric					1-5-0	
TSI+TCI+Geometric+QC					0-6-0	
Aggregated Result					2-22-0	



Figure 9 Correlation coefficient for backpropagation



Table 12 shows the correlation coefficient for (1) a Single K-nearest neighbor; (2) the GEFS initial population, and (3) GEFS after 100 K-nearest neighbors. Note that K-nearest neighbor is accurate with the initial population and as the search continues we are able to further increase accuracy. Thus, unlike the classification tasks, the initial population of the K-nearest neighbor produces good correlation and accuracy.

Figure 10 shows the comparisons between the single K-nearest neighbors and both GEFS initial populations and after generating 100 K-nearest neighbors. This figure illustrates the dramatic increases in accuracy that GEFS is able to make over a single K-nearest neighbor.

**Table 12 Correlation coefficient for K-nearest neighbor**

K-Nearest Neighbor Method	Single Learner		GEFS			
			Initial Population		100 Networks	
	R <sup>2</sup>	Std. Error	R <sup>2</sup>	Std. Error	R <sup>2</sup>	Std. Error
Lr25						
TSI	0.0092	0.8438	0.3401	0.6858	0.4274	0.6404
TSI+TCI	0.2895	0.7645	0.6600	0.5024	0.7104	0.4780
TSI+TCI+Geometric	0.2038	0.7790	0.7228	0.4661	0.7228	0.4661
TSI+TCI+Geometric+QC	0.3641	0.7487	0.6949	0.4759	0.7368	0.4549
Lr47						
TSI	0.5155	0.7979	0.7251	0.4889	0.7604	0.4561
TSI+TCI	0.5273	0.8005	0.7520	0.4651	0.7911	0.4258
TSI+TCI+Geometric	0.4666	0.8112	0.7523	0.4648	0.7977	0.4190
TSI+TCI+Geometric+QC	0.6083	0.7935	0.7627	0.4547	0.8133	0.4025
Lr50						
TSI	0.0795	0.7700	0.2686	0.6800	0.3841	0.6310
TSI+TCI	0.0587	0.7775	0.1962	0.7155	0.3778	0.6480
TSI+TCI+Geometric	0.0977	0.7662	0.2156	0.7087	0.3883	0.6437
TSI+TCI+Geometric+QC	0.0094	0.7918	0.2252	0.7109	0.4540	0.6242

Lr76						
TSI	0.0021	0.7431	0.0393	0.7388	0.1126	0.6893
TSI+TCI	0.0080	0.7251	0.2407	0.6320	0.3942	0.5805
TSI+TCI+Geometric	0.0144	0.7206	0.2381	0.6330	0.4161	0.5725
TSI+TCI+Geometric+QC	0.0049	0.7431	0.4507	0.5612	0.5603	0.5228
Lr78						
TSI	0.1160	0.7710	0.3612	0.6485	0.4461	0.6038
TSI+TCI	0.0581	0.7878	0.5184	0.5638	0.6065	0.5174
TSI+TCI+Geometric	0.0617	0.7866	0.5231	0.5609	0.5999	0.5223
TSI+TCI+Geometric+QC	0.0889	0.7786	0.5453	0.5494	0.6310	0.5047
Hall						
TSI	0.0226	0.7469	0.3831	0.6026	0.5499	0.5345
TSI+TCI	0.2375	0.7040	0.5507	0.5234	0.6681	0.4827
TSI+TCI+Geometric	0.1600	0.7133	0.5620	0.5158	0.7022	0.4622
TSI+TCI+Geometric+QC	0.2448	0.7010	0.5995	0.4983	0.7057	0.4616
Single Learner						
TSI			0-6-0	0-6-0		
TSI+TCI			0-6-0	0-6-0		
TSI+TCI+Geometric			0-6-0	0-6-0		
TSI+TCI+Geometric+QC			0-6-0	0-6-0		
Aggregated Result			0-24-0	0-24-0		
Initial Population						
TSI					0-6-0	
TSI+TCI					0-6-0	
TSI+TCI+Geometric					0-5-1	
TSI+TCI+Geometric+QC					0-6-0	
Aggregated Result					0-23-1	

### K-Nearest Neighbor vs GEFS

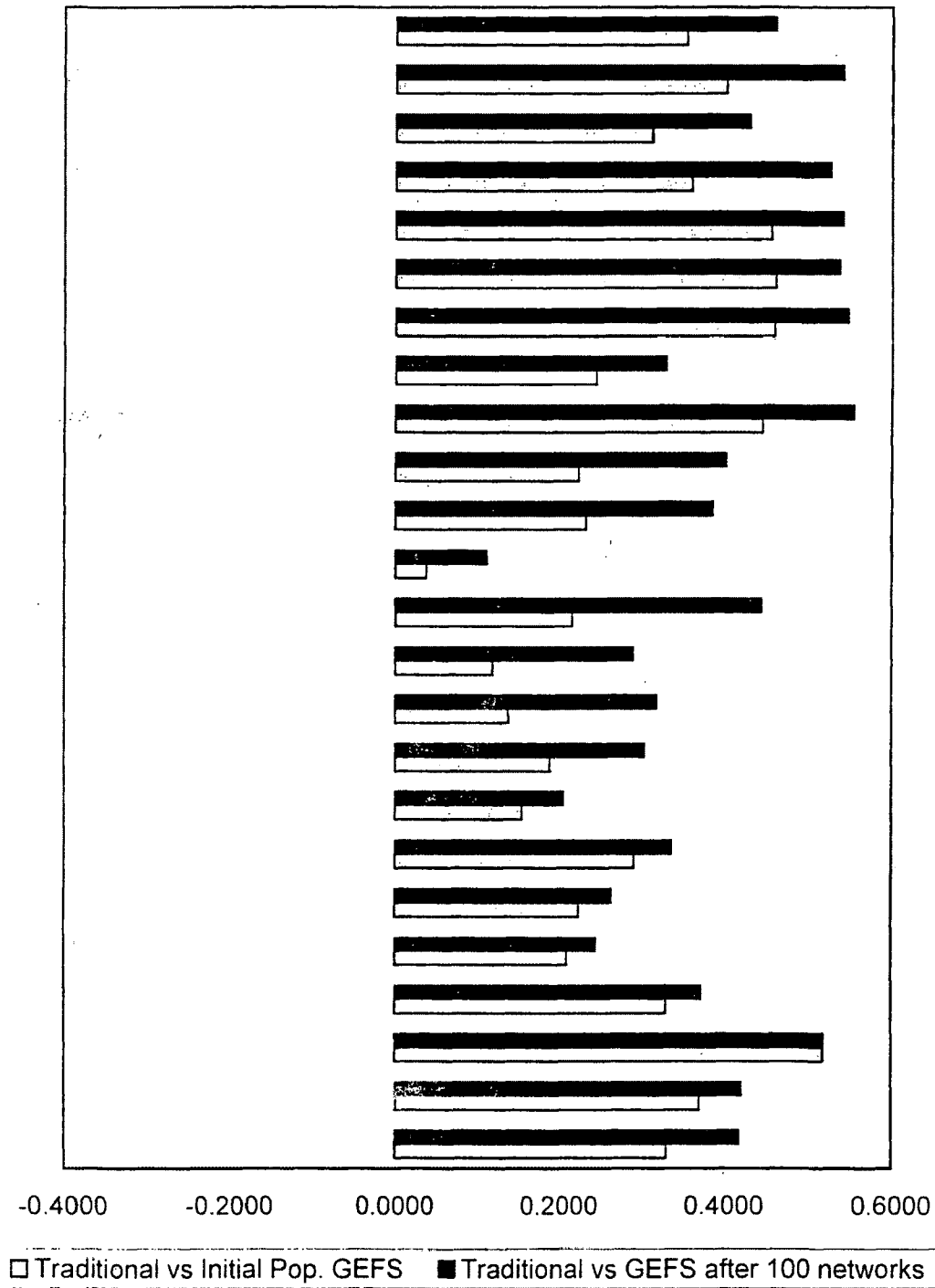


Figure 10 Correlation coefficient for K-nearest neighbor

Table 13 shows the correlation coefficient  $R^2$  of GEFS for both K-nearest neighbor and backpropagation after generating 100 hypotheses. The accuracy of both learning approaches is similar, with K-nearest neighbor holding a slight advantage.

Table 13  $R^2$  Error for K-nearest neighbor and Backpropagation

Data Set	K-nearest neighbor	Backpropagation
<b>Lr25</b>		
TSI	0.4873	0.4274
TSI+TCI	0.7843	0.7104
TSI+TCI+Geometric	0.8001	0.7228
TSI+TCI+Geometric+QC	0.8126	0.7368
<b>Lr47</b>		
TSI	0.6911	0.7604
TSI+TCI	0.7745	0.7911
TSI+TCI+Geometric	0.7833	0.7977
TSI+TCI+Geometric+QC	0.7613	0.8133
<b>Lr50</b>		
TSI	0.1296	0.3841
TSI+TCI	0.3927	0.3778
TSI+TCI+Geometric	0.5745	0.3883
TSI+TCI+Geometric+QC	0.4270	0.4540
<b>Lr76</b>		
TSI	0.0493	0.1126
TSI+TCI	0.8205	0.3942
TSI+TCI+Geometric	0.8118	0.4161
TSI+TCI+Geometric+QC	0.7999	0.5603
<b>Lr78</b>		
TSI	0.3404	0.4461
TSI+TCI	0.6712	0.6065
TSI+TCI+Geometric	0.6713	0.5999
TSI+TCI+Geometric+QC	0.6620	0.6310
<b>Hall</b>		
TSI	0.5994	0.5499
TSI+TCI	0.6953	0.6681
TSI+TCI+Geometric	0.6845	0.7022
TSI+TCI+Geometric+QC	0.7000	0.7057
K- nearest neighbor		14-10-0
Backpropagation	10-14-0	

## 6 Discussion and Future Work

We make two important conclusions regarding the new GEFS algorithm. First, for backpropagation and PNNs, GEFS produces a good initial population is both fast and simple. While this is not true for the K-nearest neighbor and naïve Bayesian learners, these learners comprise initial populations that form a good basis for later searching. That is, running GEFS further almost always increases accuracy for all learners. This is desirable, since accuracy is usually more important than the time it takes to train. Running AdaBoost and Bagging (other popular ensemble methods) longer does not appreciably increase performance since previous results have shown their performance nearly fully asymptotes at around 20 networks (Opitz, 1999). On the regression tasks, the correlation coefficient between the predicted value from the computational model and the target value derived from the toxicity test is an extremely informative metric of accuracy. The exact numeric value of most toxicity test is not as important as the relative ordering and spread of these values. Thus a perfect correlation ( $R^2 = 1.00$ ) between the computation model and the target toxicity shows the computational model is as informative as toxicity obtained from a battery of expensive and time consuming tests – regardless of the standard error. GEFS obtains impressively high correlations on fairly small datasets.

While the results of GEFS are already impressive, this is just the first step toward ensemble feature selection. Many improvements are possible and need to be explored. Areas of future research are (1) combining GEFS with AdaBoost's approach of emphasizing examples not correctly classified, (2) tuning the parameters such as the

maximum size of the feature subset in the initial population, (3) conduct the feature selection phase using fast learners like naïve Bayesian and use those feature subsets to create an ensemble neural networks, (4) instead of randomly creating the initial population, we can select the features probabilistically, and (5) include a greedy search algorithm with the GA.

## 7 Conclusion

In this thesis, we looked at the effectiveness of GEFS using a variety of machine learning techniques on a wide variety of problem domains. For both classification and regression problem domains, GEFS produced generalization accuracy that was significantly better than that of the traditional single inductive learner.

The ensemble feature selection algorithm is straightforward, simple, generates good results, and has the ability to further increase its accuracy if allowed to run longer. Thus this thesis shows the utility of feature selection for ensembles and provides an important and effective next step in this direction.

## 8 References

- Aha, D.W. (1998). Feature weighting for lazy learning algorithms. In *H. Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell MA: Kluwer.
- Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford: University Press.
- Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24(2), 123-140.
- Chtioui, Y., Bertrand, D., and Barba, D. (1998). Feature selection by a genetic algorithm. Application to seed discrimination by artificial vision, *Journal of the Science of Food and Agriculture*, 76(1).
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Controls, Signals, and Systems*, 2:303-314.
- De Jong, K. (1975). Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Thesis, *Department of Computer and Communications Sciences, University of Michigan, Ann Arbor, MI*.
- Dom, B., Niblack, W., and Sheinvald, J (1989). Feature selection with stochastic complexity, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Rosemont, IL.,
- Dudani, S. A. (1976) The distance-weighted k-nearest neighbor rule, *IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-6, Number 4, April 1976, 325-327*.
- Efron, B. and Tibshirani, R.J. (1993), *An Introduction to the Bootstrap*, London: Chapman & Hall.
- Freund, Y. Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 148-156 Bari, Italy.
- Goldberg, D. E. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms, I*, 69-93.
- Hall, M.A., and Smith, L.A. (1999). Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. *Proceedings of the Florida Artificial Intelligence Symposium (FLAIRS-99)*.
- Hansen, L. Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 993-1001.



- Hjorth, J.S.U. (1994), Computer Intensive Statistical Methods: Validation, Model Selection, and Bootstrap. *London: Chapman & Hall.*
- Holland, J. (1975) Adaptation In Natural and Artificial Systems. *The University of Michigan Press, Ann Arbor.*
- Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 359-366.
- James L. McClelland and David E. Rumelhart (1988). Explorations in parallel distributed processing, A handbook of models, programs. and excercises. *The MIT Press.*
- John. G., Kohavi, R., & Pflieger, K. (1994). Irrelevant features and the subset selection problem. *To appear in Proceedings of the Eleventh International Machine Learning Conference. New Brunswick, NJ: Morgan Kaufmann*
- Kittler, J. (1978). Feature set search algorithms, in Pattern Recognition and Signal Processing, *C.H. Chen, Ed., Sijthoff and Noordhoff, The Netherlands.*
- Kohavi, R., and John, G. (1995). Wrappers for feature subset selection. *Technical Report, Computer Science Department, Stanford University.*
- Kohavi. R., and John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence*. 97 (1-2), 245—271.
- Kohavi, R., and Sommerfield, D. (1995). Feature subset selection using the wrapper model: Over fitting and dynamic search space topology. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95).*
- Leardi, R, (1994). Application of a Genetic Algorithm to Feature Selection under Full Validation Conditions and to Outlier Detection. *Journal of Chemometrics*
- Maclin. R. and Opitz, D. (1997). An empirical evaluation of bagging and boosting In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 546-551 Providence, RI.
- Mitchell, M. (1996). An Introduction to Genetic Algorithms. *MIT Press.*
- Mitchell, T (1997). *Machine Learning, WCB/McGraw-Hill.*
- Opitz, D. (1999). Feature Selection for Ensembles. *Sixteenth National Conference on Artificial Intelligence (AAAI)*, (379-384). Orlando, FL.
- Opitz. D. and Shavlik, J. (1999). A Genetic Algorithm Approach for Creating Neural Network Ensembles. Combining Artificial Neural Nets. *Amanda Sharkey (ed.)*. (pp. 79-97). *Springer-Verlag, London.*

- Parzen, Emanuel, (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*. Vol. 33, pp. 1065-1076.
- Patterson, D. (1996). *Artificial Neural Networks*. Singapore: Prentice Hall.
- Plutowski, M., Sakata, S., and White, H. (1994), Cross-validation estimates IMSE. in Cowan, J.D., Tesauro, G., and Alspector, J. (eds.) *Advances in Neural Information Processing Systems 6, San Mateo, CA: Morgan Kaufman*, pp. 391-398.
- Rainer Stotzka, (2000) Automatic feature selection using genetic algorithms.
- Rich, E. and Knight, K., (1991). *Artificial Intelligence*. McGraw-Hill, Singapore.
- Ronald, S., Susan, R., and Andrew, M (1998) Probabilistic Neural Networks for chemical sensor array pattern recognition: comparison studies, improvements and automated outlier rejection.
- Schapire, R. (1990). The strength of weak learnability *Machine Learning*, 5(2), 197-227.
- Schapire, R., Freund, Y., Bartlett, P., Lee, W. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods *In Proceedings of the Fourteenth International Conference on Machine Learning*, 322-330 Nashville, TN
- Scherf, M., and Brauer, W. (1997). Feature selection by means of a feature weighting approach. *Forschungsberichte Künstliche Intelligenz FKI-221-97 (ISSN 0941-6358)*, Technische Universität München.
- Setiono, R., and Liu, H. (1996). Improving backpropagation learning with feature selection, *Journal of Applied Intelligence*, Vol. 6, No. 2, April, pages 129-140.
- Shao, J. and Tu, D. (1995), *The Jackknife and Bootstrap*, New York: Springer-Verlag.
- Sheinvald, J., Dom, D. and Niblack, W. (1989). Detection of Useless Features by Information Theoretic Criteria. *Proceedings of the AAAI Spring Symposium on the Theory and Application of Minimal-Length Encoding*. November.
- Sollich, P., and Krogh, A. (1996). Learning with ensembles: How over-fitting can be useful. In Touretsky, D., Mozer, M., and Hasselmo, M. (Eds.), *Advances in Neural Information Processing Systems, Vol. 8*, pp. 190--196 Cambridge, MA. MIT Press.
- Speckt, D.F. (1990). Probabilistic Neural Networks. *Neural Networks* 3 (1), 109-118.
- Speckt, D.F. (1991). A Generalized Regression Neural Network. *IEEE Transactions on Neural Networks* 2 (6), 568-576.

Vafaie, H., and De Jong, K. (1993) Robust feature selection algorithms. *Proceedings of the Fifth International Conference on Tools with Artificial Intelligence* 356-363

Weiss, S. M. & Kulikowski C. A. (1991). *Computer Systems That Learn*. Morgan Kaufmann, San. Mateo.

Whitley, D., Beveridge, R., Guerra, C. and Graves, C. (1997). Messy Genetic Algorithms for Subset Feature Selection. *International Conference on Genetic Algorithms*. T. Baeck, ed. Morgan Kaufmann.

Yang, Y and Honavar, V. (1997). Feature subset selection using a genetic algorithm. *In Proc. 2nd International Conference on Genetic Programming (GP-97)*, pages 380--385. Morgan Kaufmann.