Parametric Optimization in Data Mining Incorporated with GA-Based Search

Ling Tam, David Taniar, and Kate Smith

Monash University, School of Business Systems, Vic 3800, Australia
{Ling.Tan, David.Taniar, Kate.Smith}@infotech.monash.edu.au

Abstract. A number of parameters must be specified for a data-mining algorithm. Default values of these parameters are given and generally accepted as 'good' estimates for any data set. However, data mining models are known to be data dependent, and so are for their parameters. Default values may be good estimates, but they are often not the best parameter values for a particular data set. A tuned set of parameter values is able to produce a data-mining model of better classification and higher prediction accuracy. However parameter search is known to be expensive. This paper investigates GA-based heuristic techniques in a case study of optimizing parameters of back-propagation neural network classifier. Our experiments show that GA-based optimization technique is capable of finding a better set of parameter values than random search. In addition, this paper extends the island-model of Parallel GA (PGA) and proposes a VC-PGA, which communicates globally fittest individuals to local population with reduced communication overhead. Our result shows that GA-based parallel heuristic optimization technique provides a solution to large parametric optimization problems.

1 Introduction

Data mining is a process of extracting useful information from data. To use any data mining algorithm, a set of parameters often needs to be specified. Generally, a set of default values is provided for a given data mining algorithm, and these default parameter values were considered a 'good' estimate for any data set. However, data mining models are known to be data dependent, and so are for their parameter values. Default values may be good estimates for some data sets, but they are often not tuned to a particular data set.

Parametric search scans through parameter space in order to find the best set of parameter values. However, parameter space is often very large, and brute-force search is simply too expensive to be feasible. This paper investigates genetic algorithm based heuristic search techniques to optimize parameters of data mining algorithms. As a case study, back-propagation neural network (NN) will be used for parameter optimization in this paper.

This rest of paper is organized as follows. Section 2 gives brief introduction on data mining and genetic algorithm (GA). Section 3 shows how to apply genetic algorithms to parametric optimization in a neural network classifier. The proposed

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 582−591, 2002.

parallel version of GA-based search will also be presented. Section 4 describes environment and parameters of performance studies and shows results of performance studies, which is then followed by the conclusions in Section 5.

2 Background

In this section, we introduce data mining and its techniques. Then we briefly describe genetic algorithm (GA) and its operators. Finally, we explain reasons why GA-based search is better than random search.

2.1 Data Mining Techniques

Data mining is a process of discovering previously unknown and potential useful patterns and co-relations from data. Many data mining techniques have been proposed in literature, including *classification*, *association rule*, *sequential patterns*, and *clustering* [5].

In *classification*, we are given a set of example records, where each record consists of a number of attributes. One of the attributes is called the classifying attribute, which indicates the class to which each example belongs. The objective of classification is to build a model of how to classify a record based on its attributes. A number of classification methods have been proposed in past, including decision trees and neural networks. In *association rule*, the objective is to discover a set of rules that imply certain association relationships among a set of objects, such as "occur together" or "one implies the other". *Sequential patterns* are to find a common sequence of objects occurring across time from a large number of transactions. *Clustering* identifies sparse and dense regions in a large set of multi-dimensional data point.

To use any of these data mining algorithms, typically a set of parameters often needs to be specified. In this paper, we adopt a neural network classifier as a case study for parametric optimization.

2.2 Genetic Algorithm (GA)

GA is a domain-independent global search technique. It works with a set of solutions that are encoded in strings and a fitness function to evaluate these solutions. These strings represent points in search space. In each iteration (or generation), a new set of solutions is created by crossing some of good solutions in current iteration and by occasionally adding new diversity of search [6]. This process is done through three important operators, i.e. *selection*, *crossover*, and *mutation*. In the following section, we describe each operator in turn.

Selection. Candidate selection is based on evaluating the fitness of candidates. Candidate fitness is calculated though objective function. Fitness values generally

need to be scaled for further use. Scaling is important to avoid early convergence caused by dominant effect of a few strong candidates in the beginning, and to differentiate relative fitness of candidates when they have very close fitness values near the end of run [1]. Two basic methods are stochastic selection and deterministic selection [1]. In stochastic approach, selection is based on a probability of candidate fitness in a population, e.g. roulette wheel method. In deterministic approach, the relative fitness of a candidate is mapped to an integer number, which stands for the number of copies it will be selected.

Crossover. Crossover is the most important operator of GA. It explores new search space by recombining existing search elements, which may produce better solutions based on partial and local solutions. In a simple crossover, the operation takes two candidates and recombines them at a random point. Crossover rate is generally set with a high probability.

Mutation. Mutation is an operator to explore new search space by introducing new search elements not found in existing candidates. In a binary string, the operation flips a single bit. Mutation rate is generally set with a very low probability.

By iteratively applying these operators, GA-based search is able to converge on one of global optimal. GA is a powerful technique to solve optimization problems in a large search space, where random walk is not feasible. The strengths of GA-based search are summarized as follows:

- GA is global search method. It starts with a set of initial points rather than a single point. These points are randomly chosen from the whole search space. Search with a set of points helps GA to skip local optima. In addition, crossover operator allows neighborhood search space of current search points to be covered. And mutation operator injects new search points that may not exist in current search.
- GA-based search is directional with help of selection operator. It is directed at search spaces where current good solutions are found. With each evaluation in selection operator, inferior search points are abandoned and search then focuses on more promising points. Search diversity is large at the beginning with a population of very different fitness values, and then it is gradually reduced to a population of similar or same fitness values at the end.
- Implicit parallelism of GA. When a search point is evaluated, all schemas of which it is an instance are actually evaluated [1]. For example, a search point in binary representation of length n is an instance of 2^n schemas. When this point is evaluated, all schemas are simultaneously evaluated. It has been proven a minimum of M^3 schemas are effectively processed each time when the population of size M is evaluated [7].

It is our intention in this paper to explore GA-based search optimization for data mining algorithms, as well as its potential in parallelism.

3 GA-Guided Parameter Optimization in NN

This section describes details of GA-based search in application to parameter optimization in a back-propagation neural network classifier (NN). The reason we choose to use NN in our parameter optimization case study is that (*i*) parameter value of NN is difficult to choose, and default values are not always good estimates; and (*ii*) it has a relatively large parameter space, which is too expensive to enumerate. A number of parameters need to be specified for a back-propagation NN. The choice of single parameter value and the combining effects of different parameter values have an influence over prediction accuracy of the final data mining model. Parameters in a back-propagation NN include learning rate, momentum, the number of hidden layer and hidden node, and epoch; and their overall search space is 2^{34}.

Like any other GA applications, two essential problems have to be addressed before using GA to parameter optimization.

First is string encoding. Many encoding schemes have been proposed, for example, integer coding and gray coding. There is no standard way to choose these schemes and the choice really depends on the nature expression of problems. The order of putting encoded parameter into a string is also important. In principle, closely associated parameters need to be put together to avoid disruption caused by crossover operator. For parameter optimization problem, we use binary encoding. In addition, we assume no knowledge on relationship of parameters, and parameters are encoded in a random order.

The second issue is how to evaluate string. The fitness of string is evaluated though prediction accuracy of NN model based on a data set. In the following section, we illustrate GA and Parallel version of GA (PGA) to be used in our performance study.

3.1 The Existing GA-Based Searches

There are many variations of GA operators. Our purpose is to see the effectiveness of GA-guided parameter optimization, rather than to compare different GA operators. Therefore, we use the simple genetic algorithm (SGA) described in [1]. However, we use elitism, which always include the best search point from the last iteration to the current iteration, to prevent early convergence. The procedure of SGA is described as the following.

Step 1: Initialize population.

Step 2: For a specified number of iteration, do selection, crossover and mutation according to a set of user specified parameters. And then evaluate population individuals for next iteration.

Two major parallel versions of GA (PGA) are reported in literature. *Island model* (or network model) runs an independent GA with a sub-population on each processor, and the best individuals in a sub-population are communicated either to all other sub-populations or to neighboring population [2, 3].

Cellular model (or neighborhood model) runs an individual on each processor, and cross with the best individual among its neighbors [4]. Cellular model can be seen as a massive parallel extension of island model where population is reduced to a single individual on each processor.

One problem of island model is that best individuals in population are not immediately introduced to local sub-populations due to communication overhead. This may cause inferior search in those sub-populations, which are not physically adjacent to the sub-population where the best individual is found. To solve this problem, we propose an extended version of island model, *Virtual Community PGA* (VC-PGA).

3.2 The Proposed PGA-Based Search: Virtual Community Model (VC-PGA)

In virtual community model, each processor hosts a sub-population just like in island model. A *local community* (LC) is formed among neighboring processors, and each local community has a server processor to facilitate exchanges of best individuals within community and with other communities as shown in Figure 1. To facilitate exchange of best local individuals across community, *virtual community* (VC) can be formed and a virtual server processor performs similar tasks as a local community server processor.

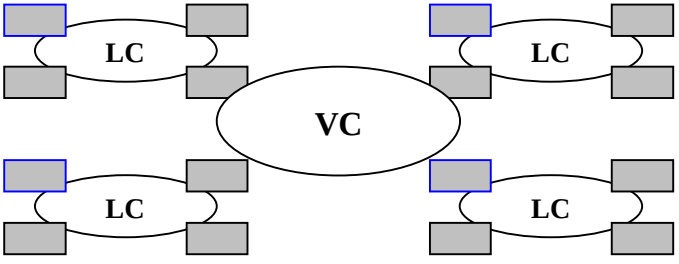

Figure 1. The Proposed VC-PGA Model

Virtual community model gives three advantages over island model: (*i*) Local sub-populations can get globally fittest individuals, (*ii*) The communication

overhead is much less expensive, and (*iii*) The evolution surface of solution space is independent from topology of network of workstations. The procedure of VC-PGA is described as follows:

Step 1: Initialize a sub-population of size *p* on each processor across network. Form a local community with adjacent processors, and randomly choose one processor as its community-server. Form virtual communities by grouping local community-servers. Higher-level of virtual communities can be formed if necessary.

Step 2: On each processor of a local community, run GA for a fixed number of iterations, and send top *n* fittest of local sub-population to the community-server. At the mean time, compare the fitness of any received individual with the best local fitness, and include it to local population if the received has a higher fitness value.

Step 3: On each community-server, find the best individuals of local community from the received best individuals of community members. Send top *n* fittest individuals of local community to parent-community server. At the mean time, compare the fitness of any received individual from its virtual community with the best local fitness, and include it to local population if the received has a higher fitness value. Send the best individuals to community members.

4 Performance Evaluation

4.1 Environment and Parameters

Performance study of sequential version GA-based parameter optimization is conducted on a single workstation, while performance of PGA-based parameter optimization is conducted over local area network, which consists of eight homogeneous Pentium workstations. Back-propagation neural network classifier in WEKA [5] data mining library is used to build and evaluate classification models. Fitness value is derived from root mean square error returned by neural network classifier. A small data set, iris.arff (150 records and 4 attributes in each record), is used in experiments. Parameters and their value ranges used in parametric optimization of backpropagation neural network classifier are summarized in Figure 2. Default values in Figure 2 refer to default values used in standard back-propagation Neural Network classifier, and included here for cross reference.

Parameters	Value Range	Bit Length	Default Value
Learning Rate	[0, 1.00]	7	0.3
Momentum	[0, 1.00]	7	0.2
Hidden layer	[1, A] where A is the number of attributes and classifiers	Round($\log_2 A$)	4
Epochs in training	[1, 1000]	10	500
Error threshold in validation testing	[1, 100]	7	30

Figure 2. Parameters in Backpropagation NN Classifier

Elitist deterministic selection is used in both GA and PGA. For GA-related parameters, a high cross over rate and a low mutation rate are used. To study speed-up of PGA, we keep total population size constant, and decrease population size on each workstation while increasing the number of workstations. Parameters in GA and PGA are summarized in Figure 3.

Parameters	GA	PGA
Generation	100	20
Population size	10, 20, 50, 100	15, 20, 30, 60, 120
Cross over rate	0.8	0.8
Mutation rate	0.01	0.01

Figure 3. Parameters in GA and PGA

4.2 Performance Analysis

We compared the performance of GA-based search with random search on a variety of population sizes. The experiments take into consideration of GA overhead by measuring results of random search and GA-based search at end of GA iterations. Performance is measured in terms of average fitness of population, and the best fitness at end of run.

As shown in Figure 4, GA-based search performs considerably better than random search regardless of population size in terms of average fitness. From the figure, we can see the speed of convergence is decreasing while population size increases. It shows that these GA-based searches actually found some good solutions after small number of iterations, and therefore the average fitness of smaller population size is higher than that of larger population size. Another point we want to mention is that at the end of 100 generations, all populations have not converged yet. It is reasonable to use fixed iteration number as a stop criteria than convergence,

because it takes a considerable time for a population to converge even although the best solution is already in the population.

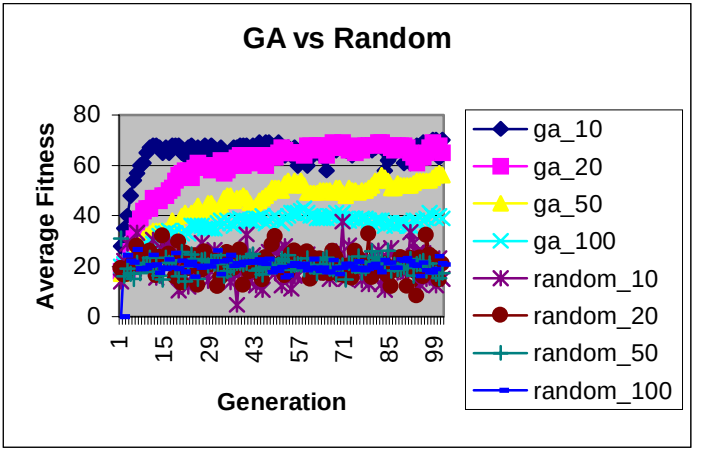

Figure 4. GA vs. Random: Average Fitness

In addition, we compared the best fitness of GA-based search with random search as shown in Figure 5. The best fitness of GA-based search always outperforms random search regardless of population size. However, the out-performance is marginal.

Population Size	GA-based Search	Random Search
10	72.53	71.76
20	73.46	72.29
50	73.89	72.24
100	73.89	72.56

Figure 5. GA vs. Random: Best Fitness

We implemented VC-PGA model on up to 8 stations in a local area network to observe its speed-up. To obtain speed-up, we keep total population size and the number of iterations constant, while using a different cluster size from 1 to 8. For example, population size is 60 on each station in a 2-station cluster, population size 30 on each station in a 4-station cluster and so on. The time for each cluster to complete its job is taken using the total time spent across all stations in a cluster at end of iteration divided by its cluster size. As shown in Figure 6, VC-PGA closely follows a linear speed-up.

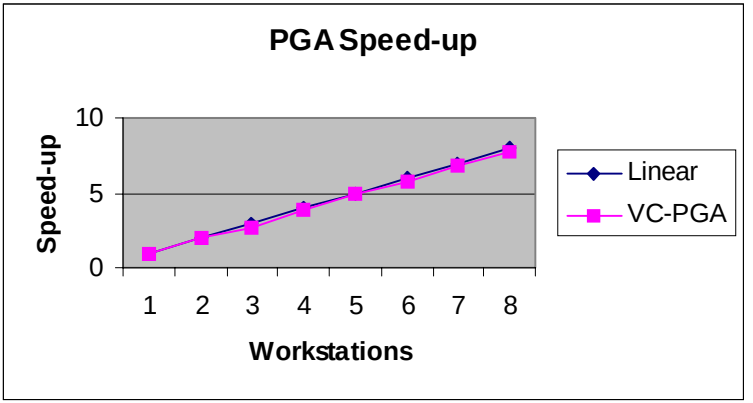

Figure 6. PGA Speed-up

Figure 7 shows population fitness in speed-up of VC-PGA parametric optimization. Best fitness found at all cases is similar to each other. It indicates that search outcomes are comparable among different experiments when more workstations are used to process the same job size. Average population fitness across all sub-populations seems to increase as more nodes are used to work with smaller population size. This is because the results are taken at end of relatively small number of generation where sub-populations have not converged.

Nodes	Population Size	Average Fitness	Best Fitness
1	120	70	77.66264
2	60	71	77.51973
4	30	73.5	77.52943
6	20	73.3	78.20998
8	15	75.3	77.51269

Figure 7. Population Fitness in PGA Speedup

5 Conclusions

Our main objective in this paper is to show that parallel heuristic optimization techniques, like GA can be useful to solve problems where large search space needs to be explored in data mining algorithms. This paper uses simple genetic algorithm (SGA) and a new variant of PGA to optimize parameter space of back-propagation neural network classifier. Our result shows that GA-based parallel heuristic

optimization technique provides a solution to large parametric optimization problems.

References

[1] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co., 1989.
[2] T. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithm", *Parallel Problem Solving from Nature*, pp. 176-185, Springer Verlag, 1991.
[3] M. Gorges-Schleuter, "Explicit parallelism of genetic algorithms through population structures", *Parallel Problem Solving from Nature*, pp 150-159, Springer Verlag, 1991.
[4] V. S. Gordon and D. Whitley, "A Machine-Independent Analysis of Parallel Genetic Algorithms", *Complex Systems*, 8:181-214, 1994.
[5] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2000.
[6] S. M. Sait and Y. Youusef, "Iterative Computer Algorithms with Application in Engineering", *Solving Combinatorial Optimization Problems*, IEEE Computer Society, 1999.
[7] L.B. Booker, D.E. Goldberg, and J.H. Holland, "Classifier Systems and Genetic Algorithms", *Artificial Intelligence*, Vol 40, pp. 235-282, 1989.