A Study on the Combination of Evolutionary Algorithms and Stratified Strategies for Training Set Selection in Data Mining

Jose Ramon Cano[1], Francisco Herrera[2] and Manuel Lozano[2]

[1] Dept. of Electronic Engineering, Computer Systems and Automatics, Escuela Superior de La Rabida, University of Huelva, 21819, Huelva, Spain. jose.cano@diesia.uhu.es

[2] Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain. herrera@decsai.ugr.es, lozano@decsai.ugr.es

Evolutionary algorithms are adaptive methods based on natural evolution that may be used for search and optimization. As Training Set Selection can be viewed as a search problem, it could be solved using evolutionary algorithms.

In this paper, we have carried out an empirical study of the performance of CHC as representative evolutionary algorithm model. This study includes a comparison between this algorithm and other non-evolutionary instance selection algorithms applied in large size data sets for Training Set Selection. The results show that the stratified evolutionary instance selection algorithms consistently outperform the non-evolutionary ones, the main advantages being: better instance reduction rates, higher classification accuracy and models that are easier to interpret.

1 Introduction

Advances in digital and computer technology that have led to the huge expansion of the Internet means that massive amounts of information and collection of data have to be processed. Due to the enormous amounts of data, much of the current research is based on scaling up [16] Data Mining (DM) ([1], [2], [3]) algorithms. Other research has also tackled scaling down data. The main problem of scaling down data is how to select the relevant data and then apply a DM algorithm [1]. This task is carried out in the data preprocessing phase in a Knowledge Discovery in Databases (KDD) process.

Our attention is focused on Data Reduction (DR) [9], which can be achieved in many ways: by selecting features [4], by making the feature-values discrete [10], and by selecting instances ([11], [17], [18], [19]). We led our study to instance selection (IS) as DR mechanism , where we reduce the number of

rows in a data set (each row represents and instance). IS can follow different strategies: sampling, boosting, prototype selection (PS), and active learning. We are going to study the IS from the PS perspective.

Instance selection mechanisms have been proposed to choose the most suitable points in the data set to become instances for the training data set used by a learning algorithm. This is our Training Set Selection (TSS) problem. TSS has been studied previously in the literature, for example, in [12], a Genetic Algorithm [5] (GA) is used for training data selection in radial based function networks.

EAs ([6], [7]) are general-purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions to problems. The basic idea is to maintain a population of chromosomes, which represent plausible solutions to the problem, which evolves over time through a process of competition and controlled variation. EAs have been used to solve the IS problem, with promising results ([12], [20]).

As with any algorithm, the issue of scalability and the effect of increasing the size of data on algorithm behaviour are always present. This scaling up drawback appears in EAs in the chromosome's size, which reduces the EAs convergence capabilities.

To avoid this drawback we offer a combination of EAs and the stratified strategy. In large and huge size we can't evaluate the algorithms over the complete data set so the stratification is the only way to carry out the executions. Combining the subset selected per strata we can obtain the subset seleted for the whole initial data set. The stratification reduces the data set size, while EAs select the best local TSS.

The aim of this paper is to study the application of a representative and efficient EA model for data reduction (CHC [13]) in TSS, and to compare it with non-evolutionary instance selection algorithms (hereafter referred to as classical ones) following a stratified strategy.

To address this, we have carried out a number of experiments with increasing complexity and size of data.

In order to do this, this paper is set out as follows. In Section 2, we introduce the main ideas about IS, describing the process which IS algorithms take part (TSS), and we also summarise the classical IS algorithms used in this study. In Section 3, we introduce the foundations of EAs and summarise the main features of them, giving details of how EAs can be applied to the TSS problem in large size data sets. In Section 4, we explain the methodology used in the experiments. Section 5 deals with the results and the analysis of large and huge data sets. Finally, in Section 6, we reach our conclusion.

2 Instance selection on data reduction

In this section we describe the strategy which IS takes part in, as a DR mechanism, as well as a summary of classical IS algorithms.

2.1 Instance selection

In IS we want to isolate the smallest set of instances which enable us to predict the class of a query instance with the same (or higher) accuracy as the original set [9]. By reducing the "useful" data set size, which can reduce both space and time complexities of subsequent processing phases. One can also hope to reduce the size of formulas obtained by a subsequent induction algorithm on the reduced and less noise data sets. This may facilitate interpretation tasks.

IS raises the problem of defining relevance for a prototype subset. From the statistical viewpoint, relevance can be partly understood as the contribution to the overall accuracy, that would be e.g. obtained by a subsequent induction. We emphasize that removing instances does not necessarily lead to a degradation of the results: we have observed experimentally that a little number of instances can have performances comparable to those of the whole sample, and sometimes higher. Two reasons come to mind to explain such an observation. First, some noises or repetitions in data could be deleted by removing instances. Second, each instance can be viewed as a supplementary degree of freedom. If we reduce the number of instances, we can sometimes avoid over-fitting situations.

2.2 Instance Selection for Training Set Selection

There may be situations in which there is too much data and this data in most cases is not equally useful in the training phase of a learning algorithm. Instance selection mechanisms have been proposed to choose the most suitable points in the data set to become instances for the training data set used by a learning algorithm. For example, in [12], a Genetic Algorithm (GA) [5] is used for training data selection in radial based function networks.

Figure 1 shows a general framework for the application of an IS algorithm for TSS. Starting from the data set, TR, the IS algorithm finds a suitable set, SS, then a learning or DM algorithm is applied to evaluate each subset selected ($C4.5$ [8] in our case) to obtain a model from the data set. This model is assessed using the test data set, TS.

2.3 Overview of Instance Selection Algorithms

Historically, IS has been mainly aimed at improving the efficiency of the Nearest Neighbour (NN) classifier. The NN algorithm is one of the most venerable algorithms in machine learning. This algorithm calculates the Euclidean distance (possibly weighted) between an instance to be classified and each training-neighbouring instance. The new instance obtained is assigned to the class of the nearest neighbouring one. More generally, the k-nearest neighbours (k-NN) are computed, and the new instance is assigned to the most frequent class among these k neighbours. The k-NN classifier was also

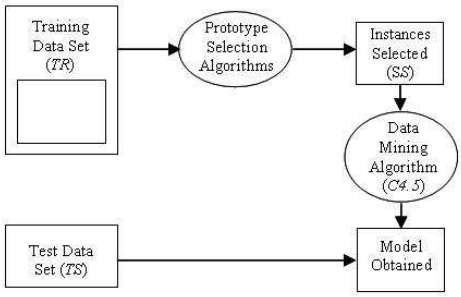

Fig. 1. TSS Strategy

widely used and encouraged by early theoretical results related to its Bayes error generalization.

However, from a practical point of view, the k-NN algorithm is not suitable for dealing with very large sets of data due to the storage requirements it demands and the computational costs involved. In fact, this approach requires the storage of all the instances in memory. Early research in instance selection firstly tried to reduce storage size. Taking as reference our study in [21] we select the most effective classic algorithms to evaluate them.

The algorithms used in this study will be:

Methods based on Nearest Neighbour Rules

- Cnn [22] - It tries to find a consistent subset, which correctly classifies all of the remaining points in the sample set. However, this algorithm will not find a minimal consistent subset.
- $Ib2$ [14] - It is similar to Cnn but using a different selection strategy.
- $Ib3$ [14] - It outperforms $Ib2$ introducing the acceptable instance concept to carry out the selection.
- Icf [17] - It tries to select the instances which classify more prototypes correctly. Icf uses coverage and reachable concepts to carry out the selection.

Methods based on ordered removal

- $Drop1$ [15] - Essentially, this rule tests to see if removing an instance would degrade leave-one-out cross-validation generalization accuracy, which is an estimate of the true generalization ability of the resulting classifier.
- $Drop2$ [15] - $Drop2$ changes the order of removal of instances. It initially sorts the instances in TR by the distance to their nearest enemy (nearest instance belonging to another class). Instances are then checked for removal beginning at the instance furthest from its nearest enemy. This tends to remove instances furthest from the decision boundary first, which in turn increases the chance of retaining border points.

- $Drop3$ [15] - $Drop3$ uses a noise filtering pass before sorting the instances in TR. This is done using the rule: Any instance not classified by its k-nearest neighbours is removed.

3 Evolutionary Instance Selection Algorithms

Most of the success of EAs is due to their ability to exploit the information accumulated about an initially unknown search space. This is their key feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic, etc.) are inappropriate. In such cases they offer a valid approach to problems requiring efficient and effective search techniques.

In this section we firstly present the key-points of their application to our problem as well as the representation and the fitness function, and then describe the EA (CHC [13]) used in this study.

3.1 Evolutionary Algorithms

EAs ([6], [7]) are stochastic search methods that mimic the metaphor of natural biological evolution. All EAs rely on the concept of a population of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as mutation, selection, and (sometimes) recombination to evolve towards increasingly better fitness values of the individuals. The fitness of an individual reflects its objective function value with respect to a particular objective function to be optimized. The mutation operator introduces innovation into the population by generating variations of individuals, and the recombination operator typically performs an information exchange between different individuals from a population. The selection operator imposes a driving force on the process of evolution by preferring better individuals to survive and reproduce when the members of the next generation are selected.

In the appendix we describe the model of EA that will be used in this paper as evolutionary instance selection algorithm. CHC is a classical model that introduces different features to obtain a trade-off between exploration and exploitation.

3.2 Evolutionary Instance Selection

EAs may be applied to the IS problem, because it can be considered as a search problem.

The objective of this paper is to study the performance of the EAs as IS algorithm applied to TSS for data reduction in KDD, comparing its results with the ones obtained by the algorithms introduced in Section 2.3.

The application of *EAs* to these approach is accomplished by tackling two important issues: the specification of the representation of the solutions and the definition of the fitness function.

Representation

Let's assume a data set denoted TR with n instances. The search space associated with the instance selection of TR is constituted by all the subsets of TR. Then, the chromosomes should represent subsets of TR. This is accomplished by using a binary representation. A chromosome consists of n genes (one for each instance in TR) with two possible states: 0 and 1. If the gene is 1, then its associated instance is included in the subset of TR represented by the chromosome. If it is 0, then this does not occur.

Fitness Function

Let SS be a subset of instances of TR to evaluate and be coded by a chromosome. We define a fitness function that combines two values: the classification performance ($clas\_per$) associated with SS and the percentage of reduction ($perc\_red$) of instances of SS with regards to TR:

$$Fitness(SS) = \alpha \cdot clas\_rat + (1 - \alpha) \cdot perc\_red. \quad (1)$$

The 1-NN classifier (Section 2.3) is used for measuring the classification rate, $clas\_rat$, associated with SS. It denotes the percentage of correctly classified objects from TR using only SS to find the nearest neighbour. For each object y in SS, the nearest neighbour is searched for amongst those in the set $SS \setminus \{y\}$. Whereas, $perc\_red$ is defined as:

$$perc\_red = 100 \cdot (|TR| - |SS|)/|TR|. \quad (2)$$

The objective of the *EAs* is to maximize the fitness function defined, i.e., maximize the classification performance and minimize the number of instances obtained. In the experiments presented in this paper, we have considered the value $\alpha = 0.5$ in the fitness function, as per a previous experiment in which we found the best trade-off between precision and reduction with this value.

Stratification Strategy

The algorithms we have studied, both classical and evolutionary, are affected when the size of the data set increases. The main difficulties they have to face are as follows:

- Efficiency. The efficiency of IS algorithms is at least $O(n^2)$, where n is the size of the data set. Most of them present an efficiency order greater than $O(n^2)$. When the size increases, the time needed by each algorithm also increases.

- Resources. Most of the algorithms assessed need to have the complete data set stored in memory to carry out their execution. If the size of the problem were too big, the computer would need to use the disk as swap memory. This loss of resources has an adverse effect on efficiency due to the increased access to the disk.
- Representation. EAs are also affected by representation, due to the size of their chromosomes. When the size of these chromosomes is too big, the algorithms experience convergence difficulties.

To avoid these drawback we led our experiments towards a stratified strategy. This strategy divides the initial data set in strata. The strata are disjoints sets with equal class distribution. We evaluate the algorithm over each strata to carry out the selection and finally we reunite the subresult to conform the final one, where we use a DM algorithm to evaluate the quality of the subset selected.

In the following section (Fig. 2) we describe the use of stratified strategy combined with EA.

Evolutive algorithms and Stratification Strategy

Following this strategy, data sets are divided into disjoint sets of equal size, T_1, T_2, ..., and T_t. We maintain class distribution within each set in the partitioning process. EA (CHC) is applied to each T_i obtaining a subset selected SS_i. We then conduct t pairs of training and test sets, (TR_i, TS_i), $i = 1, ..., t$. For each pair i, the test set, TS_i, is T_i, and the training set, TR_i, is the union of all of the other SS_j, $j \neq i$. We evaluate each pair (TR_i, TS_i) using $C4.5$ for TSS.

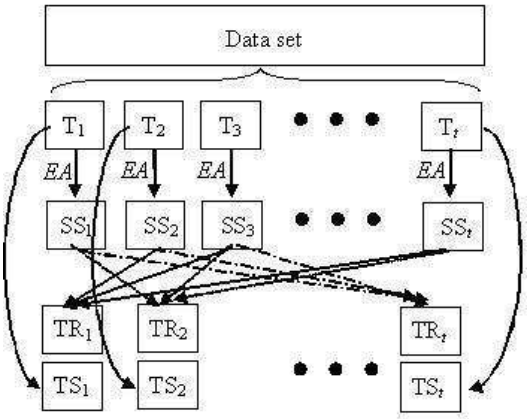

Fig. 2. Combination of EAs and Stratified Strategy.

4 Experimental Methodology

We have carried out our study of TSS problem using two size problems: large and huge. We intend to study the behavior of the algorithms when the size of the problem increases. Section 4.1 describes the data sets used (large and huge) and introduces the parameters associated with the algorithms. Finally, in Section 4.2 we present the tables where we show the results.

4.1 Data Sets

Large Size Data Set

Table 1. Large size data set

Data set	Num.Instances	Num.Features	Num.Classes
Adult	30132	14	2

Huge Size Data Set

Table 2. Huge size data set

Data set	Num.Instances	Num.Features	Num.Classes
Kdd Cup'99	494022	41	23

Parameters

Whether either small or medium data sets are evaluated, the parameters used are the same:

Table 3. Parameters

Algorithm	Parameters
$Ib3$	Acept. Level=0.9, Drop Level=0.7
CHC	Population=50, Evaluations=10000

4.2 Table of Results

In the following section we will present the structure of tables where we present the results.

Our table shows the results obtained in TSS by the classical and evolutionary instance selection algorithms, respectively. In order to observe the level of robustness achieved by all algorithms, the table presents the average (for (TR_i, TS_i) , $i = 1, ..., t$) of the results offered by each algorithm in the data sets evaluated. Each column shows:

- The first column shows the name of the algorithm
- The second column contains the average execution time associated to each algorithm. The algorithms have been run in a *Pentium* 4, 2.4 Gh, 256 RAM, 40 Gb HD.
- The third column shows the average reduction percentage from the initial training sets.
- The fourth column contains the accuracy when $C4.5$ is applied using the training set selected. This column presents two sub-columns: the first one shows the training accuracy when using the training set selected from the initial set; the second sub-column shows the test accuracy of over the test data set. Each sub-column presents the accuracy mean. We study both sub-columns in an effort to evaluate the existence of over-fitting in the instance selection phase.

5 Experimental Study

5.1 Large size data sets: Adult

Table 4 shows the results obtained in TSS by instance selection algorithms.

Table 4. TSS for Adult data set

	Exec. Time(sec)	% Reduction	$C4.5$	
			%Ac.Trn	%Ac.Test
$C4.5$	2		88.72%	85.40%
Cnn Strat	1	97.34%	88.02%	36.45%
*Drop*1 *Strat*	44	95.09%	100.00%	26.31%
*Drop*2 *Strat*	48	70.33%	71.00%	83.09%
*Drop*3 *Strat*	41	95.57%	85.34%	77.29%
*Ib*2 *Strat*	1	99.57%	95.81%	36.37%
*Ib*3 *Strat*	3	76.69%	76.72%	82.73%
Icf Strat	33	85.62%	88.75%	82.21%
CHC Strat	20172	99.38%	97.43%	82.76%

The number of stratus used in Adult data set is 10.

The following conclusions about the instance selection algorithms for TSS studying Table 4 can be made:

- In Table 4 we can see that CHC has not been improved in their test accuracy by classic methods which offer small reduction rates. EAs maintain their behaviour in test and training rates, avoiding over-fitting.
- Classic algorithms produce over-fitting in most of the cases, or offer small reduction rates.
- CHC offers the best balance between reduction and accuracy rates.

The main conclusion that can be drawn when using large data sets is that CHC is a very good algorithm for data reduction having both high reduction rates and accuracy, and avoiding over-fitting.

5.2 Huge size data sets: Kdd Cup'99

Table 5. TSS for Kdd Cup'99 data set

	Exec. Time(sec)	% Reduction	$C4.5$	
			%Ac.Trn	%Ac.Test
$C4.5$	265		99.97%	99.94%
Cnn Strat	8	81.61%	99.92%	96.43%
*Drop*1 *Strat*	111	99.97%	92.33%	34.97%
*Drop*2 *Strat*	105	76.66%	99.17%	76.58%
*Drop*3 *Strat*	131	56.74%	99.99%	75.38%
*Ib*2 *Strat*	7	82.01%	99.40%	95.05%
*Ib*3 *Strat*	3	78.92%	98.13%	96.77%
Icf Strat	242	23.62%	99.98%	99.53%
CHC Strat	1960	99.68%	98.97%	97.53%

The number of stratus used in Kdd Cup'99 data set is 100.

We would make the following conclusions about the instance selection algorithms for TSS studying Table 5:

- CHC offers the best behaviour in TSS. It presents the best reduction rates and accuracy rates. The fourth column in Table 5 shows that CHC is the best algorithm offering optimal models that can be applied to DM, as per our study.
- The classical algorithms present high accuracy rates but smaller reduction rates than CHC.

If we pay attention to execution time we can detect that all classic algoritms increase it when the data set size grows, while the CHC one suffer a

important reduction. This situation is due to in Kdd Cup'99 data set, CHC can develop a higher reduction in the first generations of the algorithm so the number of instances selected in each chromosome are reduced and the time for evaluate each solution is reduced too. So its difficult to talk about execution time for CHC as drawback because it depends of the problem where we apply it.

Clearly, when we manage data sets which are bigger than the previous ones, the CHC algorithm improves its behaviour giving the best results for scaling down data.

5.3 On the simplicity obtained for training a decision tree algorithm with a small instance selection set by *EAs* in *TSS*

The presence of noise in data sets is a widely extended pathology in the design of decision trees [23]. Sources of noise can be error in measurements, error in data encoding, error in examples, missing values, etc. Another problem that appears in the design of decision trees is the presence of clashes between examples (same attribute vector but different class). Both problems produce complex hypothesis, poor comprehensibility, over-fitting (hypothesis over-fits the data), and low classification accuracy of new data.

Larger decision trees introduce a loss of human interpretability of the models.

Pruning is the common framework to avoid the problem of over-fitting noisy data. The basic idea is to incorporate a bias towards more general and simpler theories in order to avoid overly specific theories that try to find explanations for noisy examples. The two different ways [24] to deal with this are:

- Pre-pruning: Stop growing the tree when there is not enough data to make reliable decisions, or when the examples are acceptably homogeneous.
- Post-pruning: Grow the full decision tree and then remove nodes for which there is not sufficient evidence.

The drawback of pre-pruning is that it is very difficult to know when to stop growing the tree.

Classical strategies based on post-pruning tend to under-prune or over-prune the decision tree.

We have applied CHC in the TSS problem as seen in Sections 5.1 and 5.2, obtaining greater reduction and accuracy rates. In Table 6 and 7 we show the number of rules offered by the decision tree obtained after CHC execution over the data sets. This is compared with the decision tree obtained after applying $C4.5$ to the complete data set.

In the first column of tables the name of the algorithm is stated, and the remaining ones give the average number of rules after the evaluation of both algorithms.

An analysis of Tables 6 and 7 give the following conclusions:

Table 6. Comparison of the number of rules in the Decision tree for Adult data set.

	Number of Rules
$C4.5$	327.4
$CHC\ Strat$	16.3

Table 7. Comparison of the number of rules in the Decision tree for Kdd Cup'99 data set.

	Number of Rules
$C4.5$	132.7
$CHC\ Strat$	9.1

- CHC offers representative models that maintain the balance between accuracy and reduction.
- Tables 6 and 7 show the reduction that CHC gives in the model of a decision tree generated by the subset selected. As we can see, the number of rules associated to decision trees are significantly reduced, with a more interpretable model.

Briefly summarizing this subsection, we conclude that when the CHC is applied to TSS problems it produces a small model with a high accuracy rate. This small model size increases its speed in classification, reducing its storage necessities and increasing its human interpretability.

6 Conclusion

This paper addressed the analysis of the evolutionary instance selection algorithms (CHC) and their use in data reduction for large data sets in KDD.

An experimental study was carried out to compare the results of a EA model with the classical IS ones, over large and huge size data sets. The main conclusions reached are as follows:

- $Stratified\ CHC$ outperform the classical algorithms, simultaneously offering two main advantages: better data reduction percentages and higher classification accuracy.
- $Stratified\ CHC$ avoid over-fitting in the learning phase, maintaining the accuracy rates in the training and the test sets.
- $Stratified\ CHC$ is the most appropriate algorithm evaluated, according to the algorithms that we have compared. It offers the best when we increase the size of the data set.
- $Stratified\ CHC$ algorithm significantly reduces the size of the decision tree associated to the model studied. This characteristic produces decision trees that are easier to interpret.

- In large and huge size data sets, classical algorithms do not present balanced behaviour. If the algorithm reduces the size then its accuracy rate is poor. When accuracy increases there is no reduction. The stratified version of CHC offers the best results when the data set size increases.

Therefore, as a final concluding remark, we consider Stratified strategy combined with CHC to be a good mechanism for data reduction in KDD. It has become a powerful tool to obtain small selected training sets and therefore scaling down data. CHC can select the most representative instances, satisfying both the objectives of high accuracy and reduction rates. Stratified strategy permits a reduction of the search space so we can carry out the evaluation of the algorithms with acceptable execution time.

Finally, we would say that future research could be directed towards the study of hybrid strategies between classical and evolutionary instance selection algorithms.

References

1. Adriaans P, Zantinge D (1996). Data Mining. Addison-Wesley.
2. Shanahan J G (2000). Soft Computing for Knowledge Discovery. Kluwer Academic Publishers.
3. Witten I H, Frank E (2000). Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann.
4. Liu H, Motoda H (1998). Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers.
5. Goldberg D E (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
6. Back T, Fogel D, Michalewicz Z(1997). Handbook of Evolutionary Computation. Oxford University Press.
7. Holland J H (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press.
8. Quinlan J R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.
9. Liu H, Motoda H (2001). Data Reduction via Instance Selection. In: Instance Selection and Construction for Data Mining, H. Liu, H. Motoda (Eds.), (pp. 3-20). Kluwer Academic Publishers.
10. Frank E, Witten I H (1999). Making Better Use of Global Discretization. In: Proc. Sixteenth International Conference on Machine Learning , I. Bratko, S. Dzeroski (Eds.), (pp. 115-123). Morgan Kaufmann.
11. Kibbler D, Aha D W (1987). Learning representative exemplars of concepts: an initial case of study. In: Proc. of the Fourth International Workshop on Machine Learning, (pp. 24-30). Morgan Kaufmann.
12. Reeves C R, Bush D R (2001). Using Genetic Algorithms for Training Data Selection in RBF Networks. In: Instance Selection and Construction for Data Mining, H. Liu, H. Motoda (Eds.), (pp. 339-356). Kluwer Academic Publishers.
13. Eshelman L J (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search when Engaging in Nontraditional Genetic Recombination. In: Foundations of Genetic Algorithms-1, Rawlins, G.J.E. (Eds.), (pp. 265-283). Morgan Kauffman.

14. Kibbler D, Aha D W (1987). Learning representative exemplars of concepts: an initial case of study. In: Proc. of the Fourth International Workshop on Machine Learning, (pp. 24-30). Morgan Kaufmann.
15. Wilson D R, Martinez T R (1997). Instance Pruning Techniques. In: Proceedings of the International Conference, (pp. 403-411). Morgan Kaufmann.
16. Domingo C, Gavalda R, Watanabe O (2002). Adaptative Sampling Methods for Scaling Up Knowledge Discovery Algorithms. Data Mining and Knowledge Discovery 6: 131–152.
17. Brighton H, Mellish C(2002). Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery 6: 153–172.
18. Liu H, Motoda H (2002). On Issues of Instance Selection. Data Mining and Knowledge Discovery 6: 115–130.
19. Reinartz T (2002). A unifying view on instance selection. Data mining and Knowledge Discovery 6: 191–210.
20. Kuncheva L (1995). Editing for the k-Nearest Neighbors Rule by a Genetic Algorithm. Pattern Recognition Letters 16: 809–814.
21. Cano J R, Herrera F, Lozano M (2003). Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An experimental Study. IEEE Transaction on Evolutionary Computation (In press).
22. Hart P E (1968). The Condensed Nearest Neighbor rule. IEEE Transaction on Information Theory 18 3: 431–433.
23. Safavian S R, Landgrebe D (1991). A Survey of Decision Tree Classifier Methodology. IEEE Transaction on Systems, Man. and Cybernetics 21 3: 660–674.
24. Esposito F, Malerba D and Semeraro G (1997). A comparative Analysis of Methods for Pruning Decision Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 5: 476–491.

A

Description of CHC Algorithm

During each generation the CHC develops the following steps:

1. It uses a parent population of size N to generate an intermediate population of N individuals, which are randomly paired and used to generate N potential offspring.
2. Then, a survival competition is held where the best N chromosomes from the parent and offspring populations are selected to form the next generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator. HUX exchanges half of the bits that differ between parents, where the bit position to be exchanged is randomly determined. CHC also employs a method of incest prevention. Before applying HUX to two parents, the Hamming distance between them is measured. Only those parents who differ from each other by some number of bits (mating threshold) are mated. The initial threshold is set at $L/4$, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold is reduced by 1.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any members of the parent population) the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to re-seed the population. Re-seeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other $N - 1$ new chromosomes in the population. The search is then resumed.