

Evolving Logic Programs to Classify Chess-Endgame Positions

Philip G.K. Reiser and Patricia J. Riddle
philip@cs.auckland.ac.nz

Department of Computer Science
University of Auckland
New Zealand

Abstract. In this paper, an algorithm is presented for learning concept classification rules. It is a hybrid between evolutionary computing and inductive logic programming (ILP). Given input of positive and negative examples, the algorithm constructs a logic program to classify these examples. The algorithm has several attractive features including the ability to explicitly use background (user-supplied) knowledge and to produce comprehensible output. We present results of using the algorithm to tackle the chess-endgame problem (KRK). The results show that using fitness proportionate selection to bias the population of ILP learners does not significantly increase classification accuracy. However, when rules are exchanged at intermediate stages in learning, in a manner similar to crossover in Genetic Programming, the predictive accuracy is frequently improved.

1 Introduction

This work addresses the classification problem in machine learning. That is, given training examples of the form $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ for some unknown function $y = f(\mathbf{x})$, where the \mathbf{x}_i values are vectors of the form $\langle x_{i,1}, x_{i,2}, \dots, x_{i,n} \rangle$, and y values are drawn from a discrete set of classes $\{1, \dots, K\}$; find a definition of function f such that the y value for any \mathbf{x}_i from the same distribution is accurately predicted, [4].

Evolutionary algorithms (EA) have in the past successfully been used for the classification problem. GABIL [3], and REGAL, [5], for example, create a mapping between logic expressions and fixed-length binary strings. A genetic algorithm then searches the space of strings. To evaluate strings they are mapped to the corresponding logical expression which may then be interpreted. Other work has been done on suitably modifying the operators in genetic algorithms to manipulate logical expressions directly, e.g. SAMUEL, [12], GLPS, [13].

However, as the hypothesis language becomes increasingly expressive, the space of logical expressions that needs to be searched grows combinatorially. There is accumulating evidence that indicates that the performance of evolutionary algorithms can be improved through the introduction of a local search method, [1, 7]. A local method progresses by refining an existing solution. Instead

of considering the entire search space, a small subset – the solution’s neighbourhood – is examined. This can result in the rapid location of good solutions. However, if all the points in the neighbourhood are inferior, then the algorithm becomes trapped. Unless the local method is perturbed in some way, no further improvements can be made. Evolutionary algorithms are relatively robust against such local maxima, but are poor at local refinement.

Furthermore, evolutionary algorithms do not easily lend themselves to using domain knowledge explicitly. Typically, they begin *tabula rasa*. The only real alternative is to seed, or bias the initial population of candidate solutions towards likely answers. However, this only makes certain classes of solution less likely, it does not allow solutions to be declared invalid. For example, Wong and Leung’s GLPS [13] induces logic programs by evolving a suitably chosen representation such that the syntactic correctness will be guaranteed. However, the algorithm does not easily allow background knowledge to be used to constrain the space of solutions. The semantics of expressions are ignored.

The aim of this work is to combine the local search properties of an inductive logic programming algorithm with the global search properties of an evolutionary algorithm. The algorithm presented in this paper uses a common language to express input and output, namely function-free Horn clauses. As a result knowledge can be supplied *a priori* in the form of rules and facts; this knowledge constrains the space of candidate solutions and thus eliminates from consideration solutions known to be inappropriate.

The remaining sections of this paper introduce inductive logic programming; describe EVIL_1, a hybrid ILP-EA algorithm; and finally present an empirical study of learning performance on a chess-endgame problem.

2 Evolutionary Inductive Logic Programming

2.1 Inductive Logic Programming: a Brief Introduction

Inductive logic programming (ILP), [9], is an approach to inducing concept descriptions that draws on the foundations of logic programming. The task tackled by ILP is that of developing predicate descriptions given training examples and background knowledge. More specifically, given sets of positive \mathcal{E}^+ and negative \mathcal{E}^- examples and relevant background knowledge \mathcal{B} , construct an hypothesis \mathcal{H} that is consistent and complete with respect to the training data and background knowledge.

The search through the space of logic programs is structured. A partial ordering is imposed on the space of hypothesis clauses and this orders hypotheses by generality and allows large parts of the search space to be pruned. For example, if a clause C does not cover a positive example, then none of the specialisations of C need be considered. In practice, however, this strict condition must be relaxed in order to tolerate noise in training data.

Inductive logic programming is an appealing local search method as it allows the easy incorporation of domain specific knowledge. Furthermore, it produces

comprehensible solutions. However, as ILP algorithms are typically based on the set covering algorithm, a greedy search algorithm, they are susceptible to local maxima.

2.2 Evolutionary Algorithms

Evolutionary algorithms are domain independent search algorithms inspired by principles of population genetics. Using very simple mechanisms, evolutionary algorithms exhibit complex behaviour that has been harnessed to solve some difficult problems, e.g. [2, 6]. However, evolutionary algorithms have certain drawbacks. Domain knowledge cannot be used easily. Furthermore, while such algorithms are good at establishing peaks in discontinuous multimodal objective functions, they have poor local search properties.

2.3 Integrating the EA with ILP

Inductive logic programming and evolutionary algorithms have appealing properties which appear to be complementary. Evolutionary algorithms have good global search properties, whereas inductive logic programming algorithms have good local refinement characteristics. This provokes the question can a concept learning algorithm be constructed that captures both of these properties?

Furthermore, when only a subset of the training set is seen by an ILP algorithm the theories induced are likely to be less accurate than if all the data had been used. However, in most real world applications, data will necessarily only become available gradually, or will be too great to use in one batch for the learner. It is therefore interesting to observe how algorithms perform when only samples of the data are used.

2.4 The EVIL_1 Algorithm

In the approach adopted, an evolutionary algorithm is used to direct the computational effort spent by multiple parallel instances of the ILP algorithm. The evolutionary algorithm maintains a population of agents each comprising of a logical theory (a logic program). The traditional mutation and crossover operators are replaced with the crossover operator used in [11] which in some respects is similar to crossover in Genetic Programming.

Each agent is able to induce logic programs using an ILP algorithm. An agent takes as input a random sample of the training data and induces a theory. This theory is evaluated on a validation set. Those theories with poor predictive accuracy risk extinction, while those with high predictive accuracy are likely to occupy a larger proportion of the population in the next generation. As new rules are discovered they are added to the theory. This augmented theory together with the background knowledge is then used in subsequent trials. The fitness of a theory is measured by determining its predictive accuracy on the validation set¹ comprising of the entire training set. The algorithm is shown below.

¹ It should be noted that this is not to be confused with the test set which is used only for evaluation independently of any learning.

```

procedure EVIL_1 is
begin
  initialise(Population);
  fitness = accuracy(Population,validation_set);

  while not termination_criterion loop
    for each member of population loop
      theory = select_parent(Population);
      subset = sample(training_set,sample_size);
      new_theory = induce(background_knowledge, theory, subset);
    end loop
    fitness = accuracy(Population, validation_set);
    new_population = select(population);
    population = new_population;
  end loop
  return fittest(Population);
end EVIL_1;

```

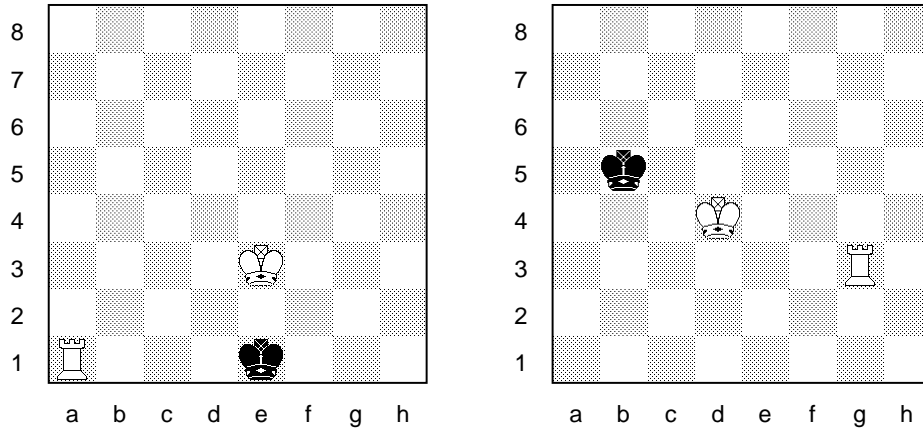
The `induce` procedure refers to a call of the inductive logic programming algorithm. The algorithm chosen was Progol, [10].

3 An Empirical Study

The aim of this empirical investigation is to examine the effect on predictive accuracy of (1) applying fitness proportionate selection on a population of ILP algorithms that only use a sample of the training set; and (2) of exchanging rules between ILP algorithms at intermediate stages.

The task domain is the Chess Endgame (KRK) problem, proposed by [8]. This problem is a widely used test problem for ILP systems. The problem may be characterised as follows. There are three pieces left on a chess board: the white king, white rook, and the black king. The objective of the learning algorithm is to discover rules to describe illegal positions when it is white's turn to move, given a set of positive and negative training instances. For example, an illegal position occurs when the black king is in check with white to move. The predicate `illegal/6` is the target to be learned, and the attributes of the target predicate are file and rank for white king, white rook and black king respectively. Examples, therefore, take the form `illegal(e,3,a,1,e,1)` and `:- illegal(d,4,g,3,b,5)` (where `:-` denotes negation). These examples correspond to board positions as illustrated in Figure 1. The data comprises of 20000 examples which are split into 10000 training and validation instances and 10000 test instances.

The following background knowledge is also supplied. The `adj/2` predicate defines cases where the rank or file represented by the left argument is adjacent



$\text{illegal}(e,3,a,1,e,1).$

$\text{not}(\text{illegal}(d,4,g,3,b,5)).$

Fig. 1. Examples of legal and illegal positions.

to that represented by the right argument. The $\text{lt}/2$ predicate defines pairs of ranks or files, where the left argument is less than the right. Consequently, rules may take the form

$\text{illegal}(A,B,C,D,C,C).$
 $\text{illegal}(A,B,C,D,E,F) :- \text{adj}(A,E).$

A population of 10 learning agents are supplied with subsets of the training set and are allowed to induce a hypothesis using the *Progol* inductive logic programming algorithm. In each generation, each agent is supplied with a 0.2% random sample of the training set². Two experiments were conducted.

The first experiment considers the effect of fitness proportionate selection. Two cases are considered: (1) multiple instances of ILP are run batch incrementally; and (2) also with fitness proportionate selection. The predictive accuracy on the test set was examined for both approaches. Figure 2 shows a scatter plot for the test set accuracy of the fitness-proportionate selection case (y -axis) against the no fitness proportionate selection case (x -axis). Points above the line $y = x$ reflect an improvement in predictive accuracy for the introduction of fitness-proportionate selection. However, the distribution of points indicates that while the introduction of fitness-proportionate selection is not advantageous, it is also not disadvantageous.

The second experiment examined the effect of rule exchange between learners. At certain intervals denoted by the communication period ω_c agents are selected to exchange parts of their clausal theory. In the control case $\omega_c = \infty$, no rule exchange occurs. In the treatment cases, $\omega_c = 3, 5$ or 10 generations. The following types of rule-exchange were considered. (1) Union: new theories

² The choice of sample size was based on a trade-off between providing enough data for rules to be found while avoiding excessive run-times.

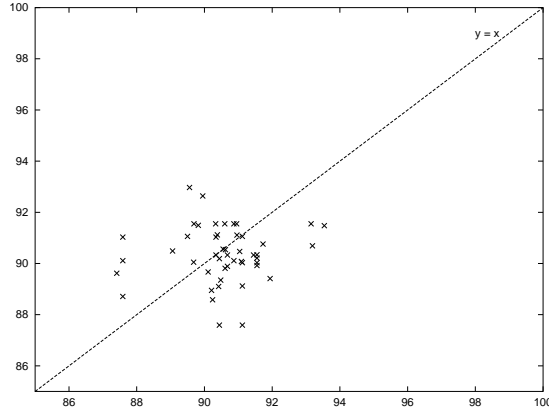


Fig. 2. Fitness-proportionate selection versus no fitness-proportionate selection.

are constructed by taking the union of two parent rule sets; (2) Crossover: new theories are constructed by exchanging rules using the crossover operator described in [11].

Figure 3 shows scatter plots for the test set accuracy of rule exchange using union, and Figure 4 the performance distributions for crossover. The graphs indicate that in the cases where the ILP algorithm performs badly, the introduction of either union or crossover increases predictive accuracy. However, in the cases where the ILP algorithm already performs well, union and crossover have a detrimental effect. The statistical significance of these results were examined using the paired t-test. For union periods 3, 5 and 10, the introduction of union does not result in a statistically significant increase in predictive accuracy, and therefore it may not be reasonably asserted that union improves performance in the chess endgame problem. However, the exchange of rules through crossover with high crossover frequency does result in a statistically significant increase. $\omega_c = 3$ ($P < 0.0005$); $\omega_c = 5$ ($P < 0.05$); $\omega_c = 10$ ($P < 0.1$).

4 Conclusions

A new hybrid evolutionary learning algorithm has been presented that induces first order logic clauses from examples. The algorithm has a number of attractive features. In particular, it allows the use of explicit background knowledge to constrain the space of solutions. In addition, the algorithm is inherently parallel and its output is sufficiently expressive to learn relational concepts.

The algorithm's learning properties were examined on the chess endgame (KRK) problem. It was shown that learning with a population of ILP learners, where fitness proportionate selection is used to bias trials towards good theories does not yield an increase in predictive accuracy. When rules are exchanged using a union operation a statistically significant increase is not observed. However, when crossover is used to exchange rules between learners, then a significantly superior predictive accuracy is attained ($P < 0.0005$).

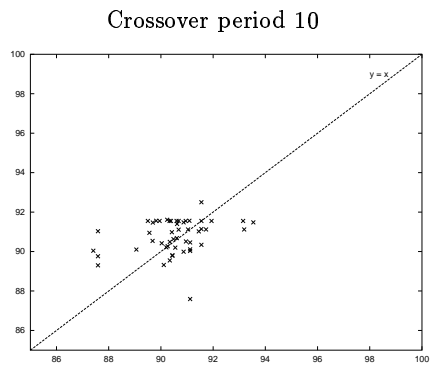
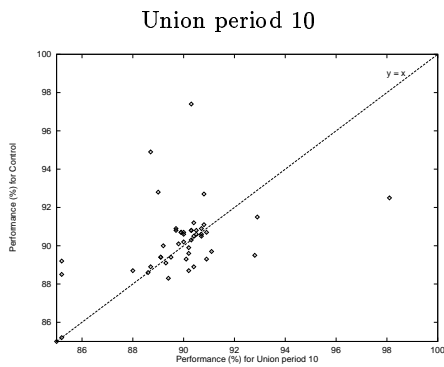
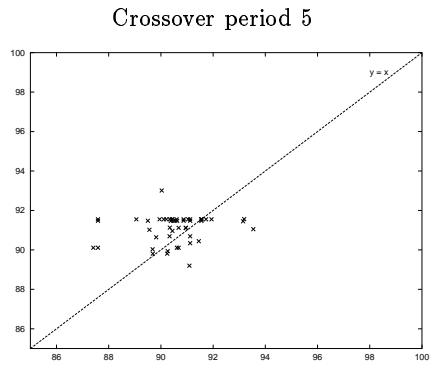
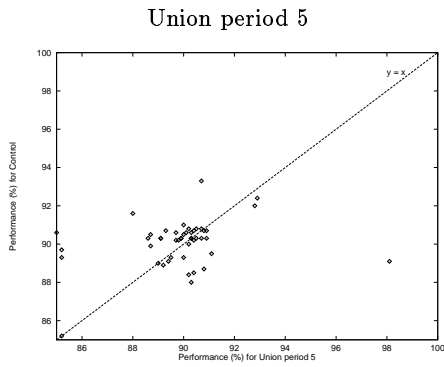
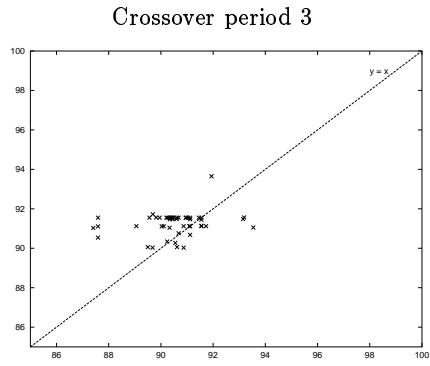
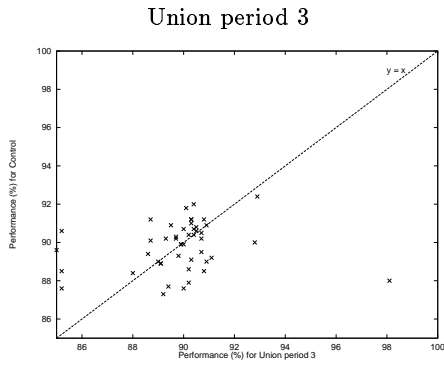


Fig. 3. Union periods 3,5,10

Fig. 4. Crossover periods 3,5,10

One possible explanation for these results is that the ILP algorithm is based on the greedy algorithm which is susceptible to local minima. Crossover, together with fitness-proportionate selection, serves as a global strategy, which can redirect the ILP algorithm to other areas of the search space.

Areas currently being pursued include a more detailed analysis of rule exchange between inductive learners and the application of evolutionary inductive logic programming to more complex problem domains.

References

1. L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
2. Kenneth A. DeJong and William M. Spears. Using genetic algorithms to solve NP-complete problems. In *International Conference on Genetic Algorithms*, pages 124–132, 1989.
3. Kenneth A. DeJong, William M. Spears, and Diana F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.
4. T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.
5. A. Giordana and L. Saitta. Regal: an integrated system for learning relations using genetic algorithms. In *Proceedings of 2nd International Workshop on Multistrategy Learning*, pages 234–249. Morgan Kaufmann, 1993.
6. David E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, Vol. 37:113–119, March 1994.
7. William E. Hart and Richard K. Belew. Optimization with genetic algorithm hybrids that use local search. In Richard K. Belew and Melanie Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms.*, volume 26, chapter 27, pages 483–496. SFI Studies in the Sciences of Complexity, 1996.
8. S. H. Muggleton, M. Bain, J. Hayes-Michie, and D. Michie. An experimental comparison of human and machine learning formalisms. In *Proc. Sixth International Workshop on Machine Learning*, pages 113–118, San Mateo, CA, 1989. Morgan Kaufmann.
9. Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.
10. Stephen Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13, 1995.
11. Philip Reiser. *EVILI*: a learning system to evolve logical theories. In *Proc. Workshop on Logic Programming and Multi-Agent Systems (International Conference on Logic Programming)*, pages 28–34, July 1997.
12. A. C. Schultz and J. J. Grefenstette. Improving tactical plans with genetic algorithms. In *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, number IEEE Cat. No. 90CH2915-7, pages 328–334, Herndon, VA, 6-9 Nov 1990. IEEE Computer Society Press, Los Alamitos, CA.
13. Man Leung Wong and Kwong Sak Leung. Inductive logic programming using genetic algorithms. In J.W. Brahan and G.E. Lasker, editors, *Advances in Artificial Intelligence – Theory and Application II*, pages 119–124, 1994.