

Evolution of Logic Programs: Part-of-Speech Tagging

Philip G.K. Reiser & Patricia J. Riddle

Department of Computer Science

University of Auckland

Auckland

New Zealand

{philip.pat}@cs.auckland.ac.nz

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 use explicit background (user-supplied) knowledge and to produce comprehensible output. We present results of using the algorithm to a natural language processing problem, part-of-speech tagging. The results indicate that using an evolutionary algorithm to direct a population of ILP learners can increase accuracy. This result is further improved when crossover is used to exchanged rules at intermediate stages in learning. The improvement over Progol, a greedy ILP algorithm, is statistically significant ($P < 0.005$).

keywords: evolutionary algorithms, inductive logic programming, natural language processing.

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 [7].

Evolutionary algorithms (EA) have in the past successfully been used for the classification problem. GABIL [5] and REGAL [8] 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 [18], GLPS [19]. However, as the hypothesis language becomes increasingly expressive, the space of logical expressions that needs to be searched grows combinatorially, rendering the task intractable.

There is accumulating evidence to indicate that the performance of evolutionary algorithms can be improved through the introduction of a local search method [3, 10, 16]. 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 [19] induces logic programs. The representation of programs as trees guarantees syntactic correctness. However, the algorithm does not allow background knowledge to be used to define the space of solutions. Since available domain semantics are ignored, many candidate solutions are considered unnecessarily.

The aim of this work is to combine the local search properties of a greedy 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 allows a complex space of candidate solutions to be defined thereby eliminating from consideration solutions known to be inappropriate.

The remaining sections of this paper introduce inductive logic programming; describe a hybrid ILP-EA algorithm (EVIL_1); and finally present an empirical study of learning performance on the part-of-speech tagging problem.

2 Evolutionary Inductive Logic Programming

2.1 Inductive Logic Programming: a Brief Introduction

Inductive logic programming (ILP) [13] 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 e , then none of the specialisations of C will cover e . 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 neo-Darwinian evolution, *cf.* [1]. Using very simple mechanisms, evolutionary algorithms exhibit complex search behaviour that has been harnessed to solve some difficult problems, e.g. [4, 9]. In [15], it is shown that genetic algorithms strongly resemble a global random search method known as the *method of generations* in classical optimisation theory. The genetic algorithm is viewed as a means of efficiently manipulating sampling distributions over the space of candidate solutions. Even when no knowledge of the problem is available, genetic algorithms are able to construct good samples of the space. 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?

2.4 The EVIL_1 Algorithm

Another perspective from which an evolutionary algorithm can be viewed is that of a competing (and sometimes cooperating) population of agents. In the approach adopted, an evolutionary algorithm directs the computational effort spent by a number of agents, where each agent consists of a logical theory (in fact a Prolog logic program). Traditional mutation and crossover operators are replaced with an inductive logic programming algorithm and a suitably-modified crossover operator. The crossover operator is in some respects similar to crossover in Genetic Programming [11].

Each agent is able to induce a theory using an ILP algorithm. An agent takes as input a random sample (with replacement) of the training data. Using this training set sample and user-supplied background knowledge as input, a theory is induced. This theory is evaluated on a validation set. The fitness of an agent is defined by the predictive accuracy of its theory on the validation set¹. 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. Over a number of generations, agents induce new rules and add them to their theory. The EVIL_1 algorithm is outlined below. The `induce`

procedure EVIL_1 **is**

begin

 initialise(population);

while not termination_criterion **loop**

for each member of population **loop**

 subset = sample(training_set, sample_size);

 theory = select_parent(population);

 new_theory = induce(background_knowledge, theory, subset);

 fitness(new_theory) = accuracy(new_theory, validation_set);

end loop

if generation % $w_c = 0$ **then**

 crossover(population);

end if

 new_population = select(population);

 population = new_population;

end loop

 return fittest(population);

end EVIL_1;

procedure refers to a call of the inductive logic programming algorithm. (The algorithm chosen was Progol [14]). The parameter w_c refers to the period with which crossover is applied, (i.e. $w_c = 1$ corresponds to crossover being applied every generation). The `crossover` procedure is described in detail in [17]. A skeletal algorithm is given below and an example shown in Figure 1.

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 be too voluminous to use in one batch for the learner, or will only become available progressively. It is therefore interesting to observe how algorithms perform when only samples of the data are used.

¹The entire training set is used as a validation set for the purpose of computing an estimate of predictive accuracy. As some of the data will already have been used as training data, the estimate will be optimistic, but this is not important as it is the value relative to the population average that is used to determine survival. It should also be made clear that training and validation data are distinct from the test set which is used only for evaluation independent of any learning.

C1: class(A,mammal) :- has_milk(A).
 C2: class(A,reptile) :- has_covering(A,scales), habitat(A,land).
 C3: class(A,bird) :- has_covering(A,feathers).
 C4: class(A,fish) :- has_gills(A).
 C5: class(A,reptile) :- has_covering(A,scales), has_legs(A,4).
 C6: class(A,fish) :- has_eggs(A).

C10: class(A,bird) :- has_legs(A,2), has_eggs(A).
 C11: class(A,mammal) :-has_legs(A,2), homeothermic(A).
 C12: class(A,reptile) :- has_legs(A,4).
 C13: class(A,fish) :- has_gills(A), habitat(A,water).
 C14: class(A,bird) :- habitat(A,land).
 C15: class(A,mammal) :- habitat(A,caves).

C1: class(A,mammal) :- has_milk(A).
 C2: class(A,reptile) :- has_covering(A,scales), habitat(A,land).
 C15: class(A,mammal) :- habitat(A,caves).
 C4: class(A,fish) :- has_gills(A).

C10: class(A,bird) :- has_legs(A,2), has_eggs(A).
 C11: class(A,mammal) :-has_legs(A,2), homeothermic(A).
 C12: class(A,reptile) :- has_legs(A,4).
 C13: class(A,fish) :- has_gills(A), habitat(A,water).
 C14: class(A,bird) :- habitat(A,land).
 C3: class(A,bird) :- has_covering(A,feathers).
 C5: class(A,reptile) :- has_covering(A,scales), has_legs(A,4).
 C6: class(A,fish) :- has_eggs(A).

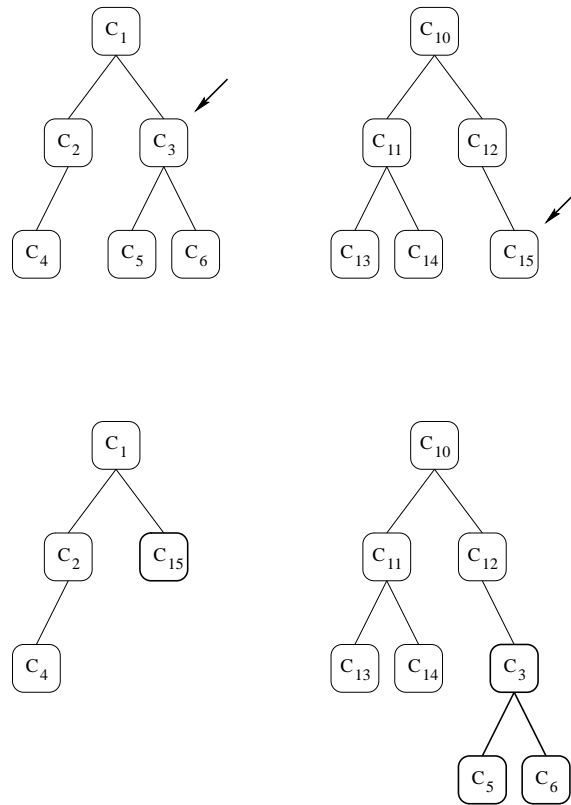


Figure 1: An example of clause crossover.

```

procedure crossover(population) is
begin
  for population_size / 2 loop
    parent_1 = select_parent(population);
    parent_2 = select_parent(population);
    new_theories = exchange_random_subtree(parent_1,parent_2);
    fitness(new_theories) = accuracy(new_theories, validation_set);
  end loop
  return population;
end crossover;

```

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 during learning.

3.1 Problem description

The problem addressed is a natural language processing problem. In analysing the syntactic structure of sentences in a text it is necessary to group words into classes or categories. Part-of-Speech Tagging is a linguistic procedure which attaches word category information to the words in a text. Rules define how, given an input word sequence, each word is assigned its corresponding part-of-speech tag.

More formally, given a sentence S , which can be defined as a string of words w_1, w_2, \dots, w_n . Part-of-speech tagging is the process of assigning each word w_i of S a corresponding tag t_i from the set T of tags. These tags are defined *a priori* by the user. Consequently for each sentence S we get an *alignment*

$$(S, T) = w_1 t_1, w_2 t_2, \dots, w_n t_n.$$

Note, however, that sentences can have more than one alignment because there exist more than one tag for each word. But, only one should be considered correct, and the tagging process should select this correct alignment.

A *tagged corpus* comprises of word and tag pairs that have been identified manually. However, developing a tagged corpus is a laborious task and consequently there is significant interest in automating this process. The learning task addressed in this paper is the discovery from a tagged corpus rules to disambiguate sentences by eliminating sentence

alignments.

Cussens [2] has used inductive logic programming to tackle this problem. When a grammar of English was supplied as background knowledge, ILP proved very successful at this task. However, because of the algorithm employed (P-Progol) only a fraction of the available data could be used to discover the rules. For instance, for one tag (nn) there are in excess of 100,000 examples yet only 6000 were used for training. Even to deal with this small proportion of the data the ILP algorithm had to be extended by adding a caching facility [2].

The EVIL₁ algorithm described in this paper, is based on using many small samples of the dataset for training. Consequently, it is suited to problems with large datasets. Previously, EVIL₁ has been applied to the task of classifying chess endgames with approximately 10,000 training examples [16, 17]. However, this domain employed little background knowledge (only 2 predicates) and the data for this artificial domain was noiseless. In the part-of-speech problem, the data is noisy and far more background knowledge is required.

3.2 Methodology

The tagged corpus used is the Wall Street Journal, which is part of the Penn Treebank [12]. The corpus contains sentences such as “*Rudolf Agnew, 55 years old and former chairman of Consolidated Gold Fields PLC, was named a nonexecutive director of this British industrial conglomerate*”. Each word in the corpus has been manually assigned one of 38 part-of-speech tags. For instance, cd is a cardinal number, dt is a determiner and nn is a singular noun. (A list of tags is given in Table 1). The Wall Street Journal data was prepared for use with Progol by Cussens; details may be found in [2]. The data was partitioned into 2/3 training and 1/3 test data. Following Cussens, a simple grammar of English was used as background knowledge. This grammar comprised of some 120 Prolog clauses, an excerpt of which is shown in Appendix A.

The EVIL₁ algorithm was used as follows. A population of 10 learning agents are supplied with subsets of the entire training set and are allowed to induce a hypothesis using the *C-Progol* inductive logic programming algorithm. In each generation, each agent is supplied with 50 examples chosen at random (with replacement) from the full training set². The algorithm was run for 100 generations.

Three experiments were conducted. The first examined the effect of fitness proportionate selection. Two cases were considered: (1) the population for the next generation was chosen deterministically: if a new theory was superior to its parent, then it replaced the parent, otherwise the parent was carried forward; and (2) the members of the next gen-

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

eration were chosen stochastically with elitist fitness proportionate selection. In order to avoid confusion over which operator was responsible for phenomena, a crossover period of $\omega_c = \infty$ was used to disable crossover. The predictive accuracy on the test set was recorded in each case. This procedure was followed for one tag, nn, using only the first 6000 training examples, with 30 repetitions. This methodology allows comparison with the results obtained by Cussens for each tag³

Examples of input and output may be found in Appendix A. Figure 2 shows a scatter plot for the test set accuracy of the fitness-proportionate selection case (y -axis) against the deterministic selection case (x -axis). Points above the line $y = x$ reflect an improvement in predictive accuracy for fitness-proportionate selection. The distribution of points indicates that the use of fitness-proportionate selection is not advantageous, if anything it is slightly disadvantageous. However, when the accuracy is compared to that obtained using a greedy ILP strategy (73.8%), the accuracies are consistently over 76% regardless of whether deterministic or fitness-proportional stochastic selection was used. This would indicate that for the nn tag sampling the data into subsets is advantageous.

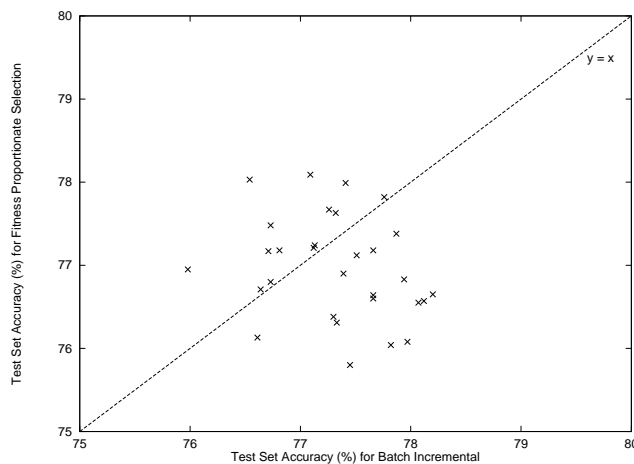


Figure 2: Comparison of accuracies for the nn tag between stochastic fitness-proportionate selection and deterministic selection.

The second experiment examined the effect of rule exchange between learners. Once again elimination rules are learned for the nn tag using only the first 6000 training examples, but at certain intervals, governed by the communication period ω_c , agents are selected to exchange parts of their logic program. In the control case $\omega_c = \infty$, no rule exchange occurs. In the treatment cases, $\omega_c = 3, 5$ or 10 generations. Rule-exchange was performed by crossover: new theories are constructed by exchanging rules using crossover. This procedure was repeated 30 times.

³Cussens also evaluated the tag theories as a complete tagging system together with statistical extensions. Although this provides a more accurate reflection of the value of the induced theories, this is not the aim in this work.

Tag	Data			Accuracy				
	Pos	Neg	Total	ILP	EVIL_1			
					$\omega_c = \infty$	$\omega_c = 3$	$\omega_c = 5$	$\omega_c = 10$
cc	1192	338	1530	59.1	77.65	78.24	77.65	77.65
cd	797	2327	3124	75.1	79.65	78.98	80.13	81.77
cln	8	0	8	n/a	n/a	n/a	n/a	n/a
cma	0	0	0	n/a	n/a	n/a	n/a	n/a
dt	11778	6770	18548	82.9	85.77	86.01	85.93	85.71
dlr	0	0	0	n/a	n/a	n/a	n/a	n/a
ex	182	660	842	88.1	94.68	95.04	94.68	94.33
fw	362	163	525	28.4	75.43	77.71	77.71	75.43
in	16448	25100	41548	82.0	77.60	77.51	77.40	77.90
jj	40041	35149	75190	62.7	74.67	76.13	75.88	74.66
jjr	3255	4182	7437	76.6	77.89	80.19	79.83	78.30
jjs	879	1068	1947	71.4	80.59	79.82	81.51	80.43
lpn	0	0	0	n/a	n/a	n/a	n/a	n/a
lqt	0	0	0	n/a	n/a	n/a	n/a	n/a
ls	5	4	9	n/a	n/a	n/a	n/a	n/a
md	209	102	311	94.1	78.10	79.05	78.10	78.10
nn	54837	50394	105231	73.8	78.34	78.81	78.81	78.81
nns	7826	13460	21286	79.4	80.90	82.20	82.00	80.95
np	16526	21672	38198	62.8	79.02	79.09	79.05	78.97
nps	5817	3483	9300	65.0	63.77	63.29	62.94	63.90
pdt	3231	470	3701	90.8	90.93	91.98	91.98	92.47
pnd	0	0	0	n/a	n/a	n/a	n/a	n/a
pos	1185	14318	15503	98.8	99.15	99.15	99.15	99.15
pp	511	168	679	83.7	91.19	90.31	90.75	91.19
ppz	152	511	663	82.1	91.86	91.86	92.31	92.31
rb	29352	13483	42835	64.7	71.99	73.92	73.80	74.51
rbr	5150	2855	8005	56.8	74.07	73.96	75.46	74.86
rbs	1064	720	1784	78.3	87.75	87.75	87.75	87.75
rp	6757	2414	9171	61.7	71.48	71.08	72.78	71.48
rpn	0	0	0	n/a	n/a	n/a	n/a	n/a
rqt	1043	0	1043	100.0	0.00	0.00	0.00	0.00
stp	11	1	12	0.0	n/a	n/a	n/a	n/a
sym	11	18	29	100.0	81.82	81.82	81.82	81.82
to	0	0	0	n/a	n/a	n/a	n/a	n/a
uh	195	33	228	82.4	89.47	81.58	88.16	81.58
vb	42673	29464	72137	92.4	95.06	95.35	95.35	95.32
vbd	27664	22701	50365	84.9	84.25	85.68	85.30	85.88
vbg	9701	8738	18439	67.9	78.10	77.47	77.96	78.71
vbn	28046	26665	54711	78.2	78.36	81.38	81.38	80.23
vbp	29465	10875	40340	89.2	92.32	93.50	92.32	92.32
vbz	22611	4745	27356	78.7	89.22	91.43	91.57	90.20
wdt	8954	3287	12241	92.5	97.21	97.30	97.48	97.48
wp	10	5	15	20.0	20.00	20.00	20.00	20.00
wpz	0	0	0	n/a	n/a	n/a	n/a	n/a
wrb	0	0	0	n/a	n/a	n/a	n/a	n/a

Table 1: Part-of-Speech tags; the distribution of examples; accuracy using ILP by Cussens [2]; accuracy using EVIL_1 with different crossover frequencies. n/a indicates insufficient data to learn rules.

Figure 3 shows scatter plots for the test set accuracy of rule exchange using crossover. The graphs indicate that there are a number of instances where the introduction of crossover increases predictive accuracy. Furthermore, the improvement is least significant when crossover is infrequent, i.e. $\omega_c = 10$. This would suggest that crossover is directly contributing to an increase in test set accuracy. The statistical significance of introducing crossover was examined using the paired t-test⁴. For crossover periods 3, 5 and 10, the introduction of crossover results in a statistically significant increase in predictive accuracy at the 95% confidence level.

The third and final experiment examined the effect of sampling from the entire training set instead of only the first 6000 examples. The EVIL_1 algorithm was run once for each tag with crossover periods $\infty, 3, 5, 10$.

The test set accuracies are shown in Table 1. Results indicate that when crossover is used with period $\omega_c = 5$ the greatest increase in accuracy is achieved. Compared with no crossover this increase is statistically significant ($P < 0.002$).

Finally, the performance of the greedy ILP strategy may be compared with EVIL_1. Results indicate that of the 34 tag concepts learned, EVIL_1 with crossover period $\omega_c = 5$ resulted in an improvement for 28 tags, statistical significance ($P < 0.005$). While these results appear very promising, there is a caveat. Because of the properties of the ILP algorithm he used, Cussens placed an upper bound on the size of training and test data. In the method reported here, no such upper bound was used; all the data was used (in some cases over 70,000 training examples). The increased accuracies in the last experiment could therefore be accounted for by changes in the underlying distribution in the data, i.e. Cussens' 6000 training and 3000 test examples were unrepresentative of the true distribution. However, it is perhaps unlikely that this should occur in so many cases.

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 allow complex spaces to be defined. In addition, the algorithm is inherently parallel; is sufficiently expressive to learn relational concepts; and produces comprehensible output.

The algorithm's learning properties were examined on the part-of-speech tagging problem. It was shown that using the evolutionary algorithm as a global strategy for an ILP algorithm led to an improvement accuracy for 28 out of 34 concepts. This result is statistically significant ($P < 0.002$). These results are consistent with results previously obtained on the chess-endgame (KRK) problem [16].

⁴It should be noted that when comparing classifiers, the paired t-test can give overoptimistic results [6]. The statistical significance of these results should be treated with appropriate caution

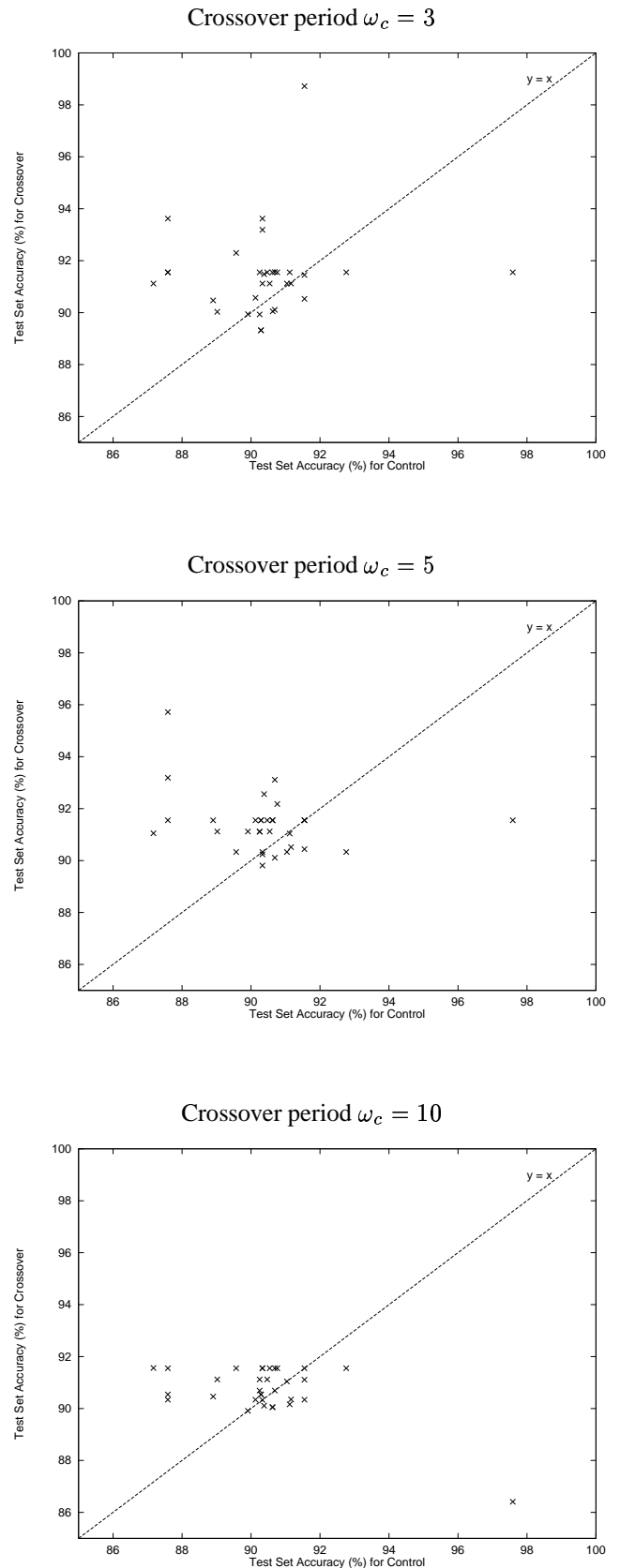


Figure 3: Test set accuracies for the nn tag for crossover periods $\omega_c = 3, 5, 10$ plotted against no crossover, $\omega_c = \infty$.

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 automatically triggering crossover by identifying local maxima.

Bibliography

- [1] Thomas Bäck. *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.
- [2] James Cussens. Part-of-speech tagging using progol. In *Inductive Logic Programming: Proceedings of the 7th International Workshop (ILP-97), volume 1297 of Lecture Notes in Artificial Intelligence*, pages 93–108. Springer, 1997.
- [3] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
- [4] 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.
- [5] Kenneth A. DeJong, William M. Spears, and Diana F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.
- [6] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1997.
- [7] T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.
- [8] A. Giordana and L. Saitta. Regal: an integrated system for learning relations using genetic algorithms. In *Proceedings of 2nd International Workshop on Multi-strategy Learning*, pages 234–249. Morgan Kaufmann, 1993.
- [9] David E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, Vol. 37:113–119, March 1994.
- [10] 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.
- [11] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [12] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19, 1993.
- [13] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.
- [14] Stephen Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13, 1995.
- [15] Gregory C. Peck and Atam P. Dhawan. Genetic algorithms as global random search methods: An alternative perspective. *Evolutionary Computation*, 3(1):39–80, 1995.
- [16] Philip Reiser and Patricia Riddle. Evolving logic programs to classify chess-endgame positions. In *Proceedings of Second Asia Pacific Conference on Simulated Evolution and Learning*, 1998.
- [17] Philip Reiser and Patricia Riddle. Evolving logic programs to classify chess-endgame positions. *Applied Intelligence*, 1999. (to appear).
- [18] 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.
- [19] 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.

Acknowledgements

The authors would like to express their gratitude to James Cussens for supplying the background knowledge and pre-processed data; Stephen Muggleton for making available Progol; Ross King for suggesting the problem domain; and members of Penn Treebank Project for supplying the linguistic data. The authors would also like to thank Jan Zytow and the anonymous reviewers for their valuable comments.

Appendix A. Examples of training data, background knowledge and induced theories

A sentence from the Wall Street Journal corpus together with two examples constructed for the highlighted word. The negative example is preceded by :-.

Rudolf Agnew, 55 years old and former chairman of Consolidated Gold Fields PLC, was named a non-executive director of this British industrial conglomerate.

```
:- rmv([in,nn,jj,cc,jj,nns,cd,cma,np,np],
      [np,np,np,cma,vbd,vbn,dt,jj,nn,in,dt,jj,jj,nn,stp],
      np).
```

```
rmv([in,nn,jj,cc,jj,nns,cd,cma,np,np],
     [np,np,np,cma,vbd,vbn,dt,jj,nn,in,dt,jj,jj,nn,stp],
     jj).
```

Excerpts of the grammar used as background knowledge (in Prolog).

```
%noun classes
```

```
%proper noun
```

```
prnoun([np|S],S) :- !.
```

```
prnoun([nps|S],S).
```

```
% noun phrase - used
```

```
nounp(L1,L2) :- noun(L1,X), !, (X=L2 ; snp(X,L2)). % "Pierre Vinken"
```

```
nounp([pp|S],S) :- !. % "us, him"
```

```
nounp(L1,L2) :- dtz(L1,L3), !, (snp(L3,L2) ; adjp(L3,L4), snp(L4,L2)).
```

```
nounp(L1,L2) :- cncy(L1,L3), !, cds(L3,L2). % "$ 20 billion "
```

```
nounp([cd|Y],L2) :- !, (X=Y ; cds(Y,X)), (X=L2 ; snp(X,L2)). % "20, 20%"
```

```
nounp(L1,L2) :- adjp(L1,L3), snp(L3,L2). % "green man"
```

A theory learned for the in tag.

```
rmv(A,B,in) :- sverb(B,C).
```

```
rmv(A,B,in) :- dt(A,C).
```

```
rmv(A,B,in) :- vpart(A,C), vp1(C,D).
```

```
rmv(A,B,in) :- noun(B,C), stp(C,D).
```

```
rmv(A,B,in) :- nounpl(A,C), cma(C,D).
```