Incrementally Learning the Rules for Supervised Tasks: the Monk's Problems

Ibrahim KUSCU
Cognitive and Computing Sciences
University of Sussex
Brighton B 1 9Q
Email: ibrahim@cogs.susx.ac.uk

December 7, 1995

Abstract

In previous experiments [4] [5] evolution of variable len th mathematical expressions containin input variables was found to be useful in findin learnin rules for simple and hard supervised tasks. owever, hard learnin problems required special attention in terms of their need for lar er size codin s of the potential solutions and their ability of eneralisation over the testin set. his paper describes new experiments aimin to find better solutions to these issues. Rather than evolution a hill climbin strate y with an incremental codin of potential solutions is used in discoverin learnin rules for the three Monks' problems. It is found that with this strate y lar er solutions can easily be coded for. Althou h a better performance is achieved in trainin for the hard learnin problems, the ability of the eneralisation over the testin cases is observed to be poor.

Keywords: Supervised learnin , hard learnin , Monk's problems, hill climbin , enetic pro rammin .

1 Introduction

In previous p per [4] genetic sed encoding schem h s een presented s potenti lly powerful tool to discover le rning rules for sever l simple supervised t s s. In nother p per [5] the model is pplied to more difficult supervised le rning pro lems such s three Mon s' pro lems nd p rity pro lems.

Com ined with genetic lgorithms the model c n successfully produce evolution of le rning rules. R ther th n se rching for gener l le rning lgorithm (s in the wor of Ch lmers [1]), the im is to see whether evolution would produce specific le rning rule for the pro lem in h nd. he represent tion schem is very simil r to the one used y Koz [3]. However, introducing prior nowledge into the represent tion of initi l solutions using pro lem specific functions is minim l, if ny t ll. In this str tegy potenti l le rning rules re encoded s r ndom m them tic l expressions t v ri le lengths. he expressions re m de up of r ndom num ers nd r ndom v ri les. he v ri les re to e inst nti ted to input v lues of tr ining set in typic l supervised le rning. By using LISP's "E L" st tement, the expressions re ev lu ted to cert in num ers nd y the help of squ shing-function this v lue is m pped to v lue in the r nge of output v lues of the supervised t s . he success of n expression in le rning the t s is determined y the num er of correct m ppings from the tr ining set nd y the degree of gener lis tion over the testing set.

he experiments showed th t the encoding str tegy nd evolution re useful to discover or re-represent the pro lem-specific-functions descri ing the le rning rules y using rel tively more gener l, fixed set of non-pro lem-specific functions. he model is lso helpful in solving h rd le rning pro lems (i.e. in the context of this rese rch h rd le rning pro lems re considered to e those supervised le rning pro lems where the le rning rule refers to the rel tionship *among* the input v lues r ther th n representing direct correl tion etween v lue(s) of input(s) nd output v ri le(s) [2] [7]) such s Mon 2 nd p rity pro lems. However, when the pro lems w s l rger nd more complex (such s Mon 2 nd 5- it nd higher p rity pro lems), the ility of the model in coding for good solutions nd effectively gener lising over the testing set w s not found to e s tisf ctory.

In this p per these issues re investig ted further. he experiments involve n incre- ment l encoding of n expression s potenti l solution. lthough the sic encoding str tegy (forming potenti l solutions s r ndom m them tic l expressions) is the s me

```
Problem  3: (jacketcolor = green and holding =  word) or
            (jacketcolor = (not blue) and body ha e = (not octagon))
```

he most difficult one mong these pro lems is the second pro lem since it refers to complex com in tion of different ttri ute v lues nd is very simil r to p rity pro lems. Pro lem one c n e descri ed y st nd rd disjunctive norm l form (DNF) nd m y e sily e le rned y ll sym olic le rning lgorithms such s Q nd Decision rees. Fin lly, pro lem three is in DNF form ut ims to ev lu te the lgorithms under the presence of noise. he tr ining set for this pro lem cont ins 5 percent miscl ssific tion.

he results of the comp rison h ve shown th t only B c prop g tion, B c prop- g tion with dec y, c sc de correl tion nd Q17-DCI h d 100 percent perform nce on Mon 2 pro lem. However, the success of B c prop g tion is pro ly is due to the conversion of origin l tr ining set v lues into in ry v lues which o viously this will directly effect the le rning rule representing the true c ses. he success of Q17-DCI is cle rly ttri ut le to one of its function which tests the num er of ttri utes for specific v lue. Mon 1 nd Mon 2 were rel tively e sy to le rn y most of the lgorithms.

2.0. T aining and Testing Sets

he tr ining nd testing sets used for the experiment in this

The Model

3.1 he n oding S hema

he potenti l le rning rules re encoded s simple m them tic l expressions r ther th n it represent tion. hey re t v ri le lengths. he expressions re produced r ndomly involving r ndom num ers (in some experiments re l num ers nd in others integers or the com in tion of the two h s een tried) nd num er of v ri les to e inst nti ted to the v lues of inputs from e ch p ttern in the tr ining set. he m them tic l oper tors include plus, minus nd multiplic tion (In ddition to these MOD nd division oper tors re lso tried. lthough their sence for the experiments to e descri ed here did not show ny notice le difference, it reduced signific ntly the comput tion l cost of processing individu ls). typic l expression for pro lem with two input v lues would loo li e this:

```
(((1 + *I1*) + (*I2* * *I1*)) - (( 0  - *I2*) - (*I1* * *I2*)))
```

his expression is r ndomly produced for pro lem with two input v lues. *I1* nd *I2* re the v ri les to e inst nti ted to the input v lues from the p tterns t e ch time of ev lu tion.

When gener ting the expressions v ri le p r meter c lled *percent ge* is used to impose how complex we w nt the expressions (i.e. longness or shortness of the expressions). It c n h ve v lues from 0 to 100. he higher the percent ge v lue the more complex the expression tends to e. In the experiments v ri le *percent ge* v lues re used depending on the complexity of the pro lem (in the r nge of 75 to 85).

Intern lly e ch of the expressions re represented s trees. he typic l structure of n expression would loo li e s in Figure 1.

```
he ex re  ion: (*I1* - ((*I2* + 1) * 0))
```

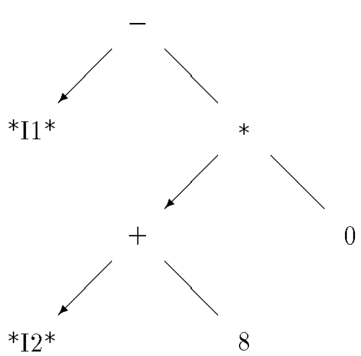

Figure 1: ree represent tion of n expression.

In order to l nce the eh vior of the expressions (i.e. the i s tow rd positive expressions) h lf of the expressions re given minus sign in front of them. his is to chieve, potenti lly n equ l ch nce of producing neg tive nd positive v lues when gener ting the expressions in the initi l popul tion.

3.2 Forming In remental Re resentations

n expression s potenti l solution to the pro lems is uilt increment lly s follows:

- Produce n expression r ndomly nd test it over the tr ining c ses. Repe t this for num er of times nd ret in the one which produces the highest success(c ll it E1).
- Next, cre te nother r ndom expression (c ll it E2). Com ine E2 with E1 y either (+) or (−) function nd test the com ined expression over the tr ining c ses. Repe t this for num er of times nd ret in the expression which produces higher success when it is com ined with E1 th n the success produced y E1 lone. (if recently dded expression does not contri ute to the level of success, repe t cre ting nd com ining until n expression which contri utes to the success is found or termin tion condition is re ched).
- hen cre te third r ndom expression (c ll it E3) nd com ine it with the previous two y using either of (+/−) functions g in nd test on the tr ining c ses. Repe t this for num er of times nd ret in the expression which produces higher success when E3 is com ined with E1 nd E2 th n the success produced y E1 nd E2 com ined only.
- Iter te ove increment l process until s tisf ctory level of success is re ched y ny com ined expression nd ret in th t expression s potenti l solution.

hus, in the experiments descri ed here modified represent tion of the expressions is used. his form of encoding introduces n explicit hier rchy to the represent tion of possi le solutions. he gener l structure of ny expression encoding for possi le solution would loo li e s follows:

```
(Squa hing- unction (( 1) +/- ( 2) +/- ...+/-( )))
```

com ined expression is ev lu ted nd the v lue o t ined is m pped to v lue in

4 Results

he perform nce of the model on the Mon 's pro lems is tested using oth the origin l

4.1 MONK 1:

```
 ( 1B 0.888889 0.927 19

(+ (+ (+ (- (- (- (+ (- (+
(- (  2 (+ *I1* *I *))
   (  2 (* *I8* *I3*)))
(- (  2 (|%| *I9* *I1 *))))
   (  2 (|%| *I15* *I1 *)))
(  2 (|%| *I6* *I11*)))
(  2 (- *I1 * *I *)))
(  2 (|%| *I10* *I1 *)))
(  2 (|%| *I6* *I3*)))
(  2 (* *I2* *I6*)))
(  2 (|%| *I13* *I3*)))
(  2 (+ (|%| *I5* *I3*)
     (- (|%| (* *I * *I2*) (|%| *I9* *I1*))))))))
```

4.2 MONK 2

```
( 28I991212*(*I2()%)051d(*-) j15.83980 d((|%|) j26.16062318..(|.+
```

```
( 2B 0.763958 0.83 312

(- (- (- (+ (- (+ (- (-
(  2 (- *I11* 0. 10278))
(  2 (|%| *I11* *I7*)))
(  2 (* *I8* *I11*)))
(  2 (* (|%| *I1* *I7*) (+ *I11* *I1 *))))
(  2 (* *I6* 0.0186 3)))
(  2 (- (|%| (- *I10* 0.  5533)
     (|%| (- (+ *I8* 0.177315) (- (- (|%| *I * *I7*)
     (* *I2* 0.92965 )))) (|%| *I13* *I7*))))))
(  2 (* *I2* 0.0151 9)))
(  2 (- (* (|%| *I7* *I15*) (|%| *I10* 0.3315 )))))
(  2 (- (- (|%| (- *I15* *I1*) (* *I5* *I1 *))
     (- (|%| (|%| *I10* *I8*)
     (* (- *I3* 0.277117) (+ *I * *I3*))))))))
   2B 6066)


( 2B 0.83925  0.961007
(- (+ (+ (- (- (- (- (- (- (- (-
(  2 (- *I11* 0. 177 9))
(  2 (|%| *I7* *I11*)))
(  2 (* (|%| *I7* *I13*)
     (|%| *I6* 0.025976))))
(  2 (- (|%| (|%| *I11* *I1*)
     (- (|%| (+ *I11* *I15*)
     (+ *I15* *I12*)))))
     ))
(  2 (* *I8* *I11*)))
(  2 (- (* (* *I * *I2*) (* *I9* *I11*)))))
(  2 (- (|%| (* (- *I3* *I7*) (|%| *I12* *I9*))
     (|%| *I5* *I1*)))))
(  2 (* *I8* 0.015916)))
(  2 (* *I15* 0.03257 )))
(  2 (* (* *I3* 0. 6 306)
     (- (+ (- (+ (|%| *I5* *I *)
     (|%| (- *I10* 0.882678)
     (* *I2* 0.098271))))
     (- (- (|%| *I13* *I8*)
     (- (* (* *I6* *I3*)
     (- (* (|%| *I10* 0. 56 5)
     (|%| *I12* *I5*)))))))))))
     )
(  2 (- (- (* (* *I6* *I13*) (* *I6* *I10*)))
     (* *I13* *I11*))))
(  2 (- (* (+ *I10* *I8*)
     (|%| (* (+ (* *I15* *I3*) (|%| *I8* 0.856525))
     (|%| *I7* *I10*))
     (+ *I1* 0.916587))))))
 2B 15699));;;
```

4.3 MONK 3

```
( 3O 0.981558 0.951792

(+
(- (  2 (* *I2* 0.195083)) (  2 (* *I1* 1.29916)))
(+ (  2 (- *I5*  .39331)) (  2 (* *I1* *I *))))
 3O 10 9))


( 3B 0.972093 0.957318

(+ (+ (+ (+ (+ (+
(  2 (- *I6* 0.181332))
(  2 (|%| *I1 * *I1 *)))
(  2 (* *I15* 0. 05052)))
(  2 (+ *I6* 0.18335)))
(  2 (- (* *I2* 0.0 8 06) (* *I6* 0.873582))))
(  2 (* *I13* 0.05 36 )))
     (+ (  2 (* (* *I2* *I13*)
        (- (|%| *I * *I3*) (|%| *I10* *I5*)))))))
```

4.4 arity roblems

4.4. 2-BIT-PARITY

```
((1.0 (+ (-
(* 0.852533 (- (* *I1* *I2*))) (* -0. 97317 *I1*))
(* 0.262082 *I2*))
P2 62))

(1.0 (- (+
(* -0.730121
   (- (* (* *I1* *I1*) (+ (- (|%| (- (+ *I1* *I2*))
   (- (- (* *I1* *I2*)) *I2*))) *I2*))))
(* -0.011222 (- *I2* (- (- *I2* *I1*)))))
(* -0.8 1233 (- (* (- (|%| *I1* *I2*) *I2*) *I2*))))
      P2 56))

(1.0 (-
(* -0.682833
    (- (- (* (- (* *I2* *I2*)) *I2*)
    (+ *I1* (- (* (+ *I2* *I2*) *I2*))))))
(* -0.8385 6 (- (* *I1* (- *I2* *I1*)))))
P2 3))

((1.0 (+ (-
(* -0.622615 *I2*) (* -0. 5  73 *I1*))
(* 0.896    (- (|%| *I2* (- (- (- (* *I1* *I2*))
   (- *I1* *I2*)))))))
P2 26))
```

4.4.2 3-BIT-PARITY

```
((1.0 (+ (+
(* 0.782908
   (- (+ (+ (- (* *I2* *I3*) (+ *I1* *I3*))
   (* (* *I3* *I3*)
   (- (|%| (* *I2* *I3*) (|%| *I1* *I3*)))))
   (- (- (|%| (* *I3* *I2*)
   (- (|%| (- *I2* *I1*) (+ *I1* *I3*)))))
   (+ *I1* *I2*)))))
(* 0.9  9 7
   (- (* (+ (- (- (* *I2* *I2*) (|%| *I2* *I2*)))
   (+ *I2* *I1*)) (+ *I3* *I1*)))))
(* 0.906881 (|%| (+ *I3* *I1*) (|%| *I3* *I2*))))
P3 632))

(1.0 (+ (- (- (+
(* 0.55 27  *I1*)
(* -0.571618 (* *I1* (- *I2* *I3*))))
(* -0.63217 (|%| *I3* (- (- (* *I2* *I3*) *I3*)))))
(* 0.1 732 (+ (- (- *I2* (* *I3* *I3*))) (+ *I1* *I1*))))
(* -0.6 95 9 (* *I1* (|%| (* (- *I1* (|%| *I2* *I3*))
      (- (- (- (+ (* *I1* *I3*) *I3*)) *I3*))) *I3*))))
P3 727))
```

4.4.3 4-BIT-PARITY

```
((0.950 11 (+ (+ (+ (+ (+
(* 3.07007 (  2 (- *I * *I3*)))
(* 2.86816 (  2 (- *I2* *I1*))))
(* 9.55089
   (  2 (- (* (- *I * *I3*)
   (- (- (|%| (|%| *I3* *I1*) (+ *I3* *I1*))
   (|%| *I1* *I3*))))))))
(* 0.171 25 (  2 (- *I3* *I *))))
(* 0.182571 (  2 (* *I3* *I1*))))
(* 6.52985
   (  2 (- (+ (- (* (|%| *I * *I1*) (|%| *I3* *I *)))
        (|%| *I2* *I *))))))
   P  9926))
```

4.4.4 5-BIT-PARITY

```
(0.9375 (- (- (+ (+ (- (- (+ (+ (+
(* 0.922868
   (|%| (- (- (|%| (- (+ (- (|%| (-
        (|%| (|%| *I * *I1*) (- *I3* *I2*)))
   (* *I5* *I3*))) (- *I2* *I5*)))
   (|%| *I1* *I3*))) (+ *I1* *I3*))
   (- (- (+ *I2* *I3*)
      (- (+ (+ *I5* *I2*) (- *I1* *I *)))))))
(* 0.081095
   (|%| (- (* (|%| *I3* *I2*) (|%| *I3* *I2*)))
   (- (+ (- *I * *I5*) (+ *I2* *I5*))))))
(* 0.112516
(* 0.389813
   (- (|%| (- *I5* *I *)
   (+ (* *I2* *I1*) (|%| (- (* (+ *I2* *I *)
   (+ *I * *I5*))) (- *I1* *I1*))))))))
(* 0.16357
   (- (* (- (- (+ *I5* *I1*) (- *I * *I3*)))
   (- *I2* *I *)))))
(* 0.216359 (- (* *I3* *I3*) (* *I3* *I2*))))
(* 0.3 0928
   (- (* (- (- (- (- (+ *I1* *I1*)
   (* (+ *I2* *I *) (|%| *I1* *I1*))) )
   (- (|%| (|%| (- *I1* *I *)
   (- (- (* *I3* *I *) (|%| *I3* *I2*))
   (*
```

```
(0.90625 (- (+ (- (- (+ (+ (+ (+
(* 0.9 2357
   (|%| (* (* (+ (|%| *I5* *I1*)
   (- (- (+ *I2* *I2*) (|%| *I5* *I5*))))
   (- (* *I2* *I3*) (|%| *I5* *I1*)))
   (* *I3* *I3*)) (- (+ (* *I * *I1*)
   (- (+ (- (- (+ (+ *I1* *I5*)
   (|%| (+ *I5* *I *) (- *I * *I5*))))
   (+ *I5* *I *)) (* (- (* (- *I5* *I2*)
   (- *I5* *I5*))) (- *I1* *I *))))))))
(* 0.532938
   (- (+ (- *I1* *I5*)
   (- (- (- *I * *I3*) (|%| *I2* *I *)) )))))
(* 0.72181
   (* (- (+ (- (|%| -  (+ (+ (*
```

onclusions

he rese rch descri ed in this p per is imed to see whether hill clim ing str tegy with n increment l encoding of r ndom m them tic l expressions c n improve the ility to code for the solutions nd gener lise over the testing set for simple nd h rd supervised le rning t s s. mong the hree Mon 's pro lems, solutions to Mon 1 nd Mon 2 were found e sily nd s tisf ctorily. lthough, the perform nce in le rning the rule for Mon 2 incre sed comp red to perform nce of the solutions found in previous experiments nd w s etter th n the perform nce of the most le rning lgorithms reported in [8], it w s not s good s the perform nce of c -prop g tion. In ll c ses, during the tr ining solution with perform nce higher th n 90 percent found for Mon 2 ut they ll showed poor gener lis tion over testing set. When tested with the p rity pro lems, simil r results re o served. he model showed incre sed perform nce in coding for up to 5 it p rity pro lems, ut when tested on incomplete p rity m ppings it showed poor gener lis tion ility. hus, the encoding str tegy together with hill clim ing is useful in finding solution for h rd le rning pro lems. However, it is not cle r whether the pro lem of gener lis tion is ttri ut le to the n ture of the h rd le rning pro lems or the encoding str tegy. In order to discover this, n n lysis of the solutions to p rity pro lems with incomplete d t sets is eing c rried out. he experiments lso showed th t the num er of individu ls processed in finding solutions re less th n wh t is required y the evolution ry ppro ch ut solutions found re longer nd more complex.

he experiments h ve shown cle rly the dv nt ge of increment l encoding. However, in this str tegy su expressions re produced r ndomly nd f r from eing optim l in terms size nd ility to gener lise. In future experiments in order to enh nce the increment l str tegy evolution c n e used to find etter su -expressions.

References

[1] D.J. Ch3999.6( )-. ndc n -prop g t d[(pro:-16000-110 ilit)999.0.37(t l)-9.654(ts)] J99.36050 d[