Can ILP Learn Complete and Correct Game Strategies?

Stephen Muggleton and Changze Xu

Computing Department, Imperial College London

Abstract. While there has been a long history of applying machine learning to game playing , our approach differs in attempting to provide a general approach to learn complete winning strategies for a group of combinatorial games and the first time to apply ILP to learn complete and correct game strategies. Instead of learning the winning moves under different game states, the learning problem we propose is to learn a classifier for P-positions, in which the next player has at least one minimax winning move. Combining such a classifier with a move generator produces a winning strategy. We report the predictive accuracy curves of learning the positions for winning strategies of a range of combinatorial games. In each of the six combinatorial games 100% accurate prediction is achieved after presenting at most 26 randomly sampled examples of play. These results were averaged over 10 independently sampled trials. We also report the predictive accuracy curves of learning the positions for the winning strategy of Nim using artificial neural network(ANN), support vector machine(SVM) and Case Based Reasoning(CBR). Even with 200 randomly sample examples, 100% predictive accuracy is not achieved in any case by ANNs, SVMs and CBRs.

1 Introduction

Automated Game Playing, Machine Learning and Logical Reasoning are each important sub-areas of research within Artificial Intelligence [13]. This paper combines these three aspects of AI. In combinatorial game theory[3], an impartial game is a game in which the allowable moves depend only on the position and not on which of the two players is currently moving. Nim is an impartial game in which two players take turns removing objects from different heaps. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap. The player to take the last object wins. Table 1 shows an example of a 3-heap Nim. The winning strategy of Nim is based on computing the xor sum of the number of objects in each pile [4] and was extended to impartial games by Sprague [14] and Grundy [6]. Sprague-Grundy theory [3] shows that any impartial game is isomorphic to a Nim game; in other words, despite appearances, all impartial games are mathematically equivalent to Nim. Later, Guy and Smith [7] applied this theory to obtain complete, closed-form solutions to a wide variety of impartial games. However, it is still non-trivial for humans to manually find an accurate mapping from an impartial game to a Nim game. It is well known [3,5] that each state of an impartial game must be classified to one of two types of positions : P-positions and N-positions and there are three theorems [2] which form the basis of winning strategies of impartial games : any move applied to a P-position turns the game into a N-position; there is at least one move that turns the game from a N-position into a P-position; the final position (when the game is over) is a P-position. Our aim is to use the ILP system Progol4.5[11] to learn the classifier for N-P positions of impartial games. Then we use the learned N-P classifier to construct the winning move generator of these games. This approach can be easily extended for partisan games(non-impartial combinatorial games) whose winning strategies are based on N-P positions such as the Northcott's game [3].

Heap1	**Heap2**	**Heap3**	**Moves**
1	2	4	Player1 removes 1 obj from Heap 3
1	2	3	Player2 removes 2 obj from Heap 2
1	0	3	Player1 removes 2 obj from Heap 3
1	0	1	Player2 removes 1 obj from Heap 1
0	0	1	Player1 removes the last obj and wins

Table 1. a 3-Heap Nim game starting wtih 1, 2, 4 objects.

2 ILP Representation of Games

An impartial game in a P-position is a positive example. An impartial game in a N-position is a negative example. There is a set of mathematical operations {xor, mod, ×, /, +, -} regarded as general background knowledge. There is some specific background knowledge that encodes the game states. A classifier for N-P positions(target hypothesis) H which entails all the positive and none of the negative examples. A winning strategy is a function that takes as input the current state of the game, player(for partisan game) and outputs a winning move that the current player can play.

2.1 Learning Schema

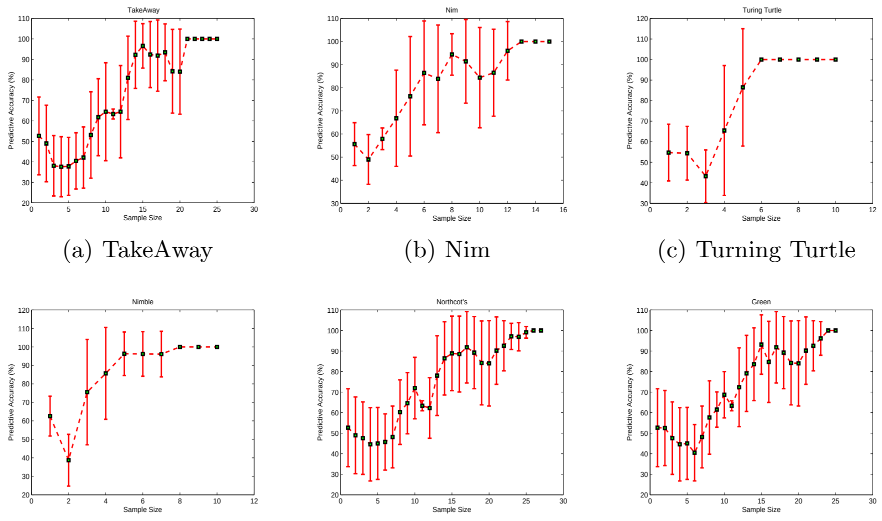
```
Given
A set $I^+$ of P-positions and A set $I^-$ of N-positions
General background knowledge ((a set of mathematical functions) GB
Apecific background knowledge for each impartial game SB
A space of N-P classifiers H
Find an N-P classifier p-Pos $\in$ H such that
$\forall\ i^+ \in I^+, \mathbf{GB} \cup \mathbf{SB} \cup$ p-Pos $\models i^+$ and $\forall\ i^- \in I^-, \mathbf{GB} \cup \mathbf{SB} \cup$ p-Pos $\not\models i^-$
Construct the winning move generator
Input : CState-current game state     Player-current player
Output : Action-winning move     AState-the game state after taking the action
If p-Pos(CState)
Return an Action $\in$ LegalActions(Player) with the least change to CState.
Else
    Forall Action $\in$ LegalActions(Player)
     If play(Player,CState,Action)
        $\leftarrow$updateState(Player,CState,Action,AState),p-Pos(AState)
     Return Action and AState.
```

3 Experiments

Experiments for learning the N-P classifier for the winning strategies of six different combinatorial games are performed in this section. We will analyse the relationships between sample size and predictive accuracy of each game.

3.1 Learning N-P Position of Impartial Games by ILP

We use Progol4.5 [10,11], as the reference system. We use the minimax algorithm to generate examples assuming that both players are playing winning strategies. We fix the sample size to 50 where 32 of them are positive examples and 18 of them are negative examples. We use the qsample program in Progol4.5 which is based on the sampling algorithm [8] to randomly generate training and testing examples. If N examples are chosen from the total sample dataset as training examples, then the remaining 50-N examples are test examples. For each game, we conduct experiments by sampling 1 to 27 training examples and repeat the same experiment 10 times for each size of training examples and calculate the mean predictive accuracy and the standard deviation. For all impartial games, we use the same set of math functions {xor, mod, ×, /, +, -} as general background knowledge. Fig.1 shows the relationships between sample size and predictive accuracy of games: TakeAway, Nim, Nimble, Turning Turtles, Northcott's, Green Hackenbush respectively. For more details of the game rules see [3]. In general the predictive accuracy increases with the increase of sample size for each of the six games. As we can see from the six graphs, at most 26 examples are required for a predictive accuracy of 100% in each case.



(a) TakeAway (b) Nim (c) Turning Turtle

(d) Nimble (e) Northcott's (f) Green Hachenbush

Fig. 1.

3.2 Experiment to Determine Performance on Nim of ANNs, SVMs and CBRs

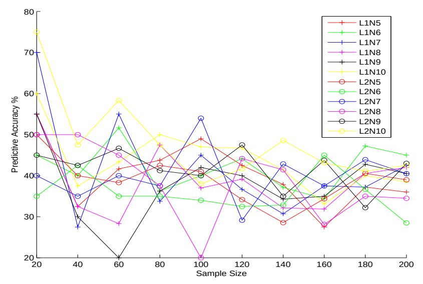
We use Matlab 2011B as the reference system and use the ANN toolbox (contains the ANN method) and the Bioinformatics Toolbox (contains the SVM method) in Matlab and implemented a CBR system by Matlab to learn the N-P classifier of a 3-heap Nim. We use 200 randomly generated examples (100 positive and 100 negative) as the total sample dataset. Examples are represented in two forms: Decimal Form and Binary Form. An example "3 4 5 1" in the Decimal Form, which means a positive example of a 3-heap Nim with 3,4,5 objects, is represented as "00011 00100 00101 1" in the Binary Form (the last digit represents the class of the example: 1≡Positive and 0≡Negative). We randomly choose 20,40,60,80,100,120,140,160,180,200 examples (half positive and half negative) from the total sample dataset as the subsample and use 10-fold cross validation to find the average predictive accuracy for each size of subsample. Fig.2 shows the relationships between sample size and predictive accuracy of Nim games under six different kinds of configurations. In general SVM has higher predictive accuracy than ANN under each sample form and for each machine learning technique, learning under Binary Form has higher predictive accuracy and CBR has extremely high predictive accuracy under Binary Form but extremely low predictive accuracy under Decimal form. According to the Fig.2, the best results are achieved with SVMs using kernel functions ′linear′ and ′poly′(with max exponent 3) under Binary Form which makes the predictive accuracy tend to 90% and with the CBR using similarity measure ′L2′ under Binary Form which makes the predictive accuracy up to 98% . Even though the predictive accuracies in these cases are very high, they still cannot reach 100% which is the basis to construct a winning move generator. The settings of the experiments are shown in Table 2 .

Fig	System	Sample Form	Parameter	Value
2(a)	ANN	Decimal	L/N/Train/Transfer/R	2. . . 3/5. . . 10/trainscg/tansig/e-4
2(b)	ANN	Decimal	L/N/Train/Transfer	2. . . 3/5. . . 10/trainscg/tansig/e-4
2(c)	SVM	Binary	Kernel	linear/poly/quad/rbf
2(d)	SVM	Binary	Kernel	linear/poly/rbf
2(e)	CBR	Decimal	Similarity Measure	dice|| jaccard|| naive|| L2
2(f)	CBR	Binary	Similarity Measure	dice|| jaccard|| naive|| L2

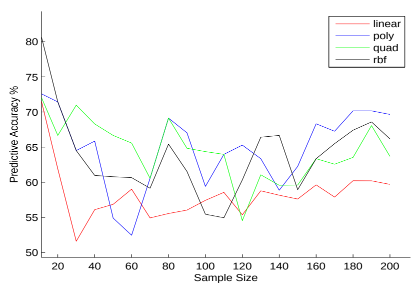
Table 2. L means Layers, N means Neuron,Train means training Function, Transfer means Transfer Function, R means the learning rate and more detail about the meaning of the ANN Values at www.mathworks.com/help/toolbox/bioinfo/ref/svmtrain and the SVM Values at www.mathworks.com/help/toolbox/nnet/ and the CBR Values at [9]

4 Related Work, Conclusions and Future Work

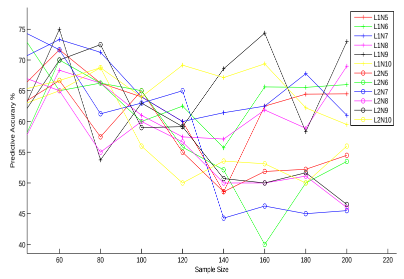
In this paper, four machine learning approaches : ILP, ANN, SVM and CBR for learning N-P positions of combinatorial games have been demonstrated. Experiments have shown that ILP is able to learn the N-P classifier for the winning strategy of each of six different combinatorial games given with up to 26 examples and sufficient background knowledge. Even with 200 randomly sampled



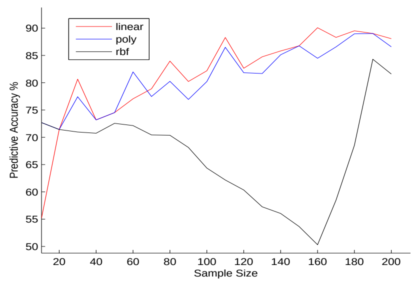
(a) Learn 3-heap Nim using ANN



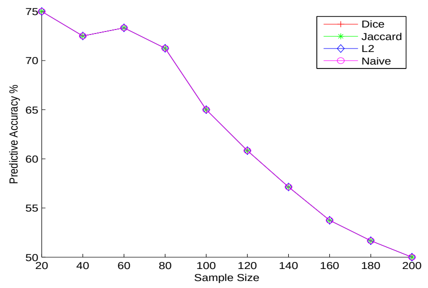
(b) Learn 3-heap Nim using SVM



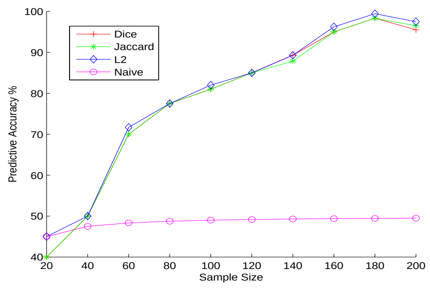
(c) Learn 3-heap Nim using ANN under Binary Form



(d) Learn 3-heap Nim using SVM under Binary From



(e) Learn 3-heap Nim using CBR



(f) Learn 3-heap Nim using CBR under Binary From

Fig. 2.

examples, 100% predictive accuracy is not achieved in any case by ANNs, SVMs CBRs. Another disadvantage of ANNs, SVMs, CBRs compared with ILP systems is that they are unable to give the players useful hints to play the Nim game as the learned N-P classifier is not human readable. The proposed method can be applied for any partisan games whose winning strategies are based on P and N-positions. Mihai Oltean uses the Multi-Expression Programming-a genetic programming variant to compute the winning strategy for Nim [12]. The idea is to read the game tree, check N and P-positions during the traversing of the game tree and count the number of configurations that violates the rules of the winning strategy. However, the Genetic Programming is not guaranteed to converge and the approach is only tested on a single game. By contrast, our approach has been demonstrated to produce correct solutions across a variety of combinatorial games. M. Bain and S. Muggleton [1] applied the ILP system Golem to learn the optimal strategies of certain the Chess endgame King-and-Rook-against-King but were only able to learn a complete strategy for depths 0,1 and 2. Future work will aim to extend our ILP approach to learn strategies across a broader range of combinatorial games, including impartial game under Misre play(the player that is forced to take the last stone loses) and complex partisan games and apply multi-clause learning to learn game strategies.

References

1. M. Bain and S. Muggleton. *Learning Optimal Chess Strategies.* Oxford University Press, 1994.
2. Elwyn R. Berlekamp. Blockbusting and domineering. *J. Comb. Theory Ser.A*, 49(1), 1988.
3. Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning ways for your mathematical plays*, volume 1. A K Peters/CRC Press, London, 2001.
4. Charles L. Bouton. Nim, a game with a complete mathematical theory. In *The Annals of Mathematics*, 2, pages 35–39, Princeton, 1902. Annals of Mathematics.
5. Martin Gardner. *Hexaflexagons and other mathematical diversions: the first Scientific American book of puzzles & games.* University of Chicago Press, 1988.
6. P. M. Grundy. Mathematics and games. *Eureka*, 2:6–8, 1939.
7. Richard K. Guy and Cedric A.B. Smith. The g-values of various games. pages 514–526, Princeton, 1956. Proceedings of the Cambridge Philosophical Society.
8. D.E. Knuth. *The art of computer programming*, volume 1. Addison-Wesley, 1997.
9. T. Warren Liao, Zhiming Zhang, and Claude R Mount. *Similarity measures for retrieval in case-based reasoning systems*, volume 12. 1998.
10. S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
11. S.H. Muggleton and J. Firth. CProgol4.4: a tutorial introduction. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, pages 160–188. Springer-Verlag, 2001.
12. Mihai Oltean. Evolving winning strategies for nim-like games. *World Computer Congress*, 9(2):353–364, August 2004.
13. S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson, New Jersey, 2010. Third Edition.
14. R. P. Sprague. ber mathematische kampfspiele. *Tohoku Mathematical Journal*, 41:438–444, June 1936.