Learning Attributes/Rules for KRK End Game Chess (ILP)

M.Arunothia, 13378

Under the guidance of Mr. Ashudeep Singh

Submitted to Prof. Amitabha Mukherjee for partial fulfilment of the course requirements for CS365A, IITK

Abstract

This project is an attempt to understand the applications of Inductive Logic Programming (ILP) in solving End Game Chess(King-Rook-King).Machine Learning gives the results without giving explicitly the rules to arrive at them. While, Logic programming works completely on pre-defined premises and deduction rules and learns nothing from the examples available. ILP is the bridge between these two - it learns from examples and returns rules in the language of logic. This project uses this bridging nature of ILP to study some configurations in K-R-K End Game Chess.

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Contents

List of Figures

List of Tables

1 Introduction

1.1 End Game Chess

- End Game Chess (or Chess Endgame) is the game of chess when only few pieces are left on board.
- In this project we will be dealing with King-Rook-King End Game. (i.e) the board is left with only the pieces - White Rook, White King and Black King.
- Read More

1.2 Inductive Logic Programming (ILP)

Given background Knowledge B, postive examples E^+ and negative examples E^-, ILP comes with a heuristic (or rule) H such that

$$\begin{gathered} B \cup H \models e, \forall e \in E^+ \\ B \cup E^- \models \neg H \\ B \cup H \text{ is consistent.} \end{gathered}$$

- Read More

1.3 Prolog

Prolog is a logic programming language.

- Read More

1.4 Progol

Progol is one of the methods of implementation of ILP.

- Read More

2 Motivation

Search heuristics and some traditional machine learning methods have already been proved best in providing winning algorithms for playing chess. The motivation in this project is not to make the machine play chess. It is to use the machine to give us rules and ideas of different board configurations and in some sense give us intuitive rules to play an optimal game. This motivation is best captured by Inductive Logic Programming. Logic Programming is the way to get rules written down for anything and it is inductive here, because all we have is examples and some rough idea about the attributes.

3 Data

- Data Base Source
- Format
  Every line in the data has 7 attributes seperated by comma, they are -
  - White King file (column)
  - White King rank (row)
  - White Rook file
  - White Rook rank
  - Black King file
  - Black King rank
  - Optimal depth-of-win for White in 0 to 16 moves, otherwise draw.
- The data has been formatted to suit progol environment.

4 Methodology

4.1 Inverse Resolution

- A method to Inductive Reasoning
- Involves the inversion of resolution operator.
- The idea is, in resolution, we club clauses by cancelling literals that exist in both its positive and negative form. In the inverse resolution process, we try to find those cancelled literals to get our rule clause.
- It is non-deterministic and heavily depends on the background knowledge provided.
- Read More

4.2 Progol

- Direct application of inverse resolution can lead to computational explosions.
- Progol deals with this complexity by using *Mode Directed Inverse Entailment* (MDIE) approach.

4.2.1 MDIE

- We define head modes and corresponding body modes.
- Predicates, whose rules we want to estimate are given by the head modes and the formulation of these rules will use the predicates given by their corresponding body modes.
- For every example, the most specific heuristic h is estimated using inverse entailment.
- We now have a heuristic space H to which all h belong.
- A general-to-specific search is then applied over H to obtain the least general heuristic that along with background knowledge entails all positive example and no negative example.

4.2.2 Head and Body Modes in this Project

The modes are nothing but the predicates (attributes) used in the code. Refer Table1, Table2 from the results section for further details.

4.3 Work included after Poster Presentation

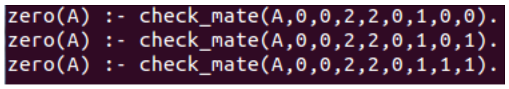
The project has been extended to solve special case draw configurations. See Results section.

Definition 1. *Special Case Draw Configuration considers only those draw configurations where Black King has a chance to attack the White Rook in the next move, leaving both ends with only their kings left. Notice, draw can happen under other circumstances also - like repeating loops, etc.*

5 Results

5.1 Check-Mate Configuration

The rule obtained for Check-Mate configuration in terms of the attributes defined is shown in the Table 1 and Figure 1.



```
zero(A) :- check_mate(A,0,0,2,2,0,1,0,0).
zero(A) :- check_mate(A,0,0,2,2,0,1,0,1).
zero(A) :- check_mate(A,0,0,2,2,0,1,1,1).
```

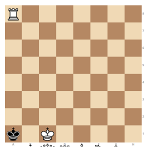
Figure 1: Results - Check Mate (KRK) Rules Learnt

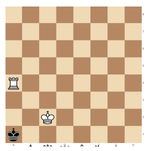
- The Table 1 should be read as
  For a configuration to be check-mate in K-R-K, the attribute from column 1 should take the value given in column 2 and vice-versa.

Attribute	**Value**
Minimum File/Rank difference between White Rook and Black King	0
Distance from edge for Black King	0
Maximum File/Rank difference between White King and Black king	2
Minimum File/Rank difference between White King and White Rook	2
Distance from edge for White Rook	0
Is White Rook on same edge as Black King?	1
Minimum File/Rank difference between White King and Black King	0/1
Is black King on the corner?	0/1

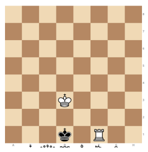
Table 1: Results - Check Mate (KRK) Rules Learnt

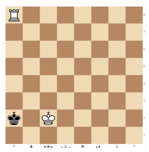
- The last two attributes are conditional over each other, (i.e)
  - When the Black King is in the corner, The Minimum File/Rank difference between the White King and the Black King can take both the values 0 and 1.





  - When the Black King is not in the corner, The Minimum File/Rank difference between the White King and the Black King should take the value 0.





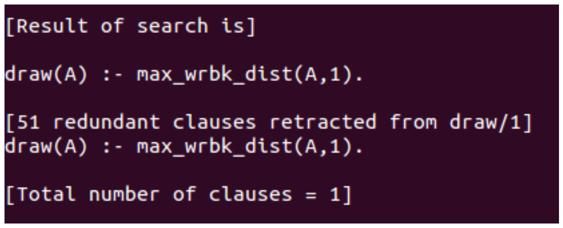
- zero(A) -
  - 'A' is the configuration (position) vector that uniquely determines a configuration.
  - zero(A) - Returns true if it is a check mate configuration.
  - zero(A) - Returns false if it is not a check mate configuration.
- check mate(A,$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$) -
  - 'A' is the configuration (position) vector that uniquely determines a configuration.
  - a_i gives the i^{th} attribute's value from the table
- The Figure 1 shows the clause obtained from the code.

5.2 Special Case Draw (def.1) Configuration

The rule obtained for special case draw (def.1) configuration in terms of the attributes defined is shown in the Table 2 and Figure 2.

Attribute	Value
Maximum File/Rank difference between White Rook and Black King	1

Table 2: Results - Special Case Draw (def.1) (KRK) Clause Learnt



```
[Result of search is]

draw(A) :- max_wrbk_dist(A,1).

[51 redundant clauses retracted from draw/1]
draw(A) :- max_wrbk_dist(A,1).

[Total number of clauses = 1]
```

Figure 2: Results - Special Case Draw (def.1) (KRK) Clause Learnt

- draw(A) -
  - 'A' is the configuration (position) vector that uniquely determines a configuration.
  - draw(A) - Returns true if it is a draw configuration.
  - draw(A) - Returns false if it is not a draw configuration.
- max wrbk dist(A,a) -
  - 'A' is the configuration (position) vector that uniquely determines a configuration.
  - 'a' gives the maximim file/rank distance between the white rook and the black king.
- Figure 2 gives the resultant clause from the code.

6 Insights

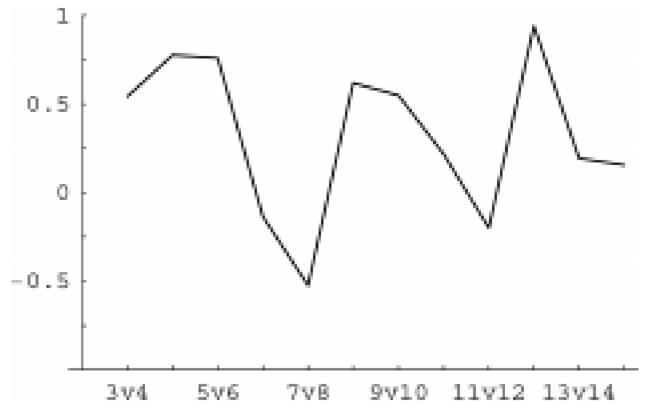
6.1 Traditional Approaches to Game Solving

- The traditional approach for any zero-sum game is Min-Max Search along with suitable pruning strategies to reach some depth where a heuristic can be estimated.

- The approach of ILP attempted in this project is different because it is more about finding strategic rules than about making the machine play an optimal game.

6.2 Highlights from the paper [1] *Learning long-term chess strategies from databases*

- The aim here is to find the optimal set of moves for the White King starting from a given configuration of the KRK board.
- The main approach in this paper is centered in a stage-wise categorization of the End Game Chess.
- The idea is that - the set of attributes that are important in different stages can be drastically different.
- The stage boundaries are detected using the correlation plot measuring the similarity between adjacent gain vectors for the end game.
- The mathematical representation of the gain vectors is discussed in detail in the paper. Read More
- The intuition is that - adjacent gain vectors will tend to be independent whenever there is a stage boundary, (i.e.) correlation will tend towards minima.
- The correlation plot for KRK end-game taken from the paper [1] is shown



- Therefore, the two local minima in the plot show the stage boundaries for the KRK End Game.
- In the paper, the author identifies these stages as

  - Far (The aim here is to take the Black King to the Edge)
  - Middle (The aim here is to take the White King near the Black King to create a lock.)
  - Close (The aim here is to mate using the White Rook)

- This idea has been clubbed with search heuristics and decision trees in the paper, an approach very different from the ILP approach used in this project. But, some ideas from this paper can be used with the ILP approach to attain rules for optimal moves.

7 Conclusions

7.1 Check-Mate Configuration

- The rules obtained for check-mate condition is very intuitive. It can easily be understood and contemplated by humans.

7.2 Special Case Draw (def.1) Configuration

- The clause
  draw(A) :- max wrbk dist(A,1)
  says that for this special case of draw (def.1), the White Rook should be just one step away from the Black King which was the natural expectation.

- The more important fact is that the converse is also true, (i.e) If the White Rook is just one step away from the Black King then in an optimal game of chess, this will surely lead to a draw.

7.3 Future Improvements

- This method could be extended to find the rule for a general draw (or any) configuration. The challenge would be in defining suitable attributes.

- Once we have every configuration mapped to a clause of attributes, we can extend this approach to learn the rule (or strategy) for the next optimal move for the player (both black and white). Notice, here the background knowledge will comprise of the clauses of the configurations that we obtained previously.

- Once this is done, this approach could be clubbed along with the stage-wise categorisation mentioned in the paper *Learning long-term chess strategies from databases* [1] to obtain the optimal moves from fewer and well defined set of attributes.

8 Source Code

You can access the code written for this project here :

Source Code

References

[1] Aleksander Sadikov Ivan Bratko. *Learning long-term chess strategies from databases.* Springer Science + Business Media, LLC 2006, 2005.

[2] Sam Robert. *An Introduction to Progol.* 1997.

[3] archive.ics.uci.edu. machine-learning-databases, 1994. [Online; accessed 25-March-2015].

[4] *Induction as Inverted Deduction : Lecture Notes (COM3250/6170).* The University of Sheffield, 2010-2011.