Towards the Automatic Synthesis of Interpretable Chess Tactics

Abhijeet Krishnan
Chris Martens

Explainable Agency in Artificial Intelligence Workshop
March 1, 2022

P O E M
Principles of Expressive Machines

NC STATE UNIVERSITY

Presenter

Abhijeet Krishnan

5th year PhD student

POEM Lab

NC State University

Advisor: Dr. Chris Martens

Contents

- Overview
- Motivation
- Methodology
- Experiment
- Results
- Limitations + Future Work

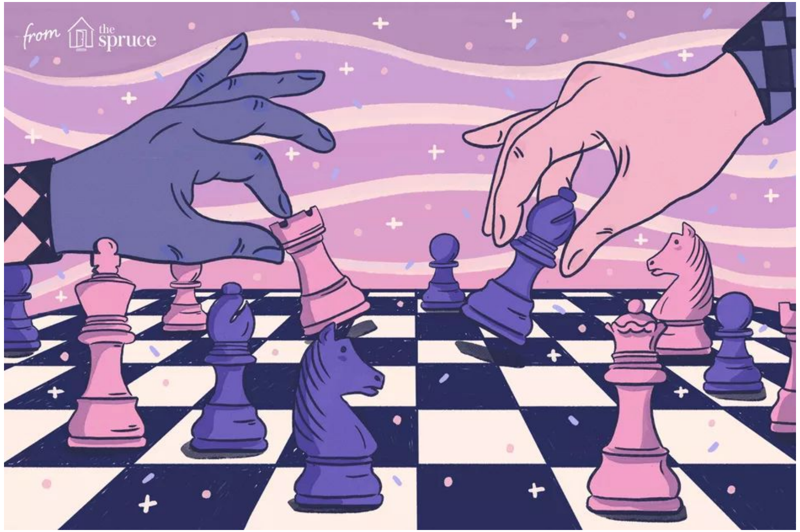


Illustration: Alison Czinkota. © The Spruce, 2018 [1]

Overview

What? a method to learn *interpretable* chess tactics

Why? want to *understand* engine moves to improve our play

How? model *chess tactics* as symbolic programs, learn them using *inductive logic programming* (ILP) and define *metrics* to measure their utility

And? identified learned tactics as resembling a beginner player

Motivation

- Superhuman AI exist for many games
  - *Starcraft 2* (AlphaStar)
  - *Dota 2* (OpenAI 5)
  - Chess (Stockfish 14)
  - Go (Leela Zero)
  - Poker (DeepStack)
  - ...

DeepMind

OpenAI

Stockfish

DeepStack

Motivation

- Not just faster hardware!

"[AlphaStar] demonstrated strategies I hadn't thought of before, which means there may still be new ways of playing the game that we haven't fully explored yet."
- Dario "TLO" Wünsch, top professional SC2 player on his games with AlphaStar [4]


[3]

Motivation

- Could inform how humans play

"In essence I have become a very different player in terms of style than I was a bit earlier, and it has been a great ride."

- Magnus Carlsen, 5x world chess champion on AlphaZero's influence on him [6]

Motivation

Could explanations of superhuman agents for games improve human play?

Related Work

1. Strategy synthesis
2. Explainable RL
3. Patterns in Chess

Related Work: Strategy Synthesis

- automated game analysis
- evolutionary approaches to learn rule-based agents for games like
  - *Neverwinter Nights* (Spronck, Sprinkhuizen-Kuyper, and Postma 2004)
  - Hanabi (Canaan et. al. 2018)
  - μRTS (Mariño et. al. 2021)
  - ...
- We
  - use ILP to learn rules from limited background knowledge
  - measure similarity of rules to reference engine

Related Work: Explainable RL

- Explainability using surrogate model
  - decision trees (Bastani, Pu, and Solar-Lezama 2018)
  - programmatic policies (Verma et. al. 2019)
  - ...
- We -
  - introduce collection of chess tactics as a surrogate model learned using ILP
  - attempt to incorporate domain knowledge to make model more interpretable
  - allow for surrogate model to be *incomplete*

Related Work: Patterns in Chess

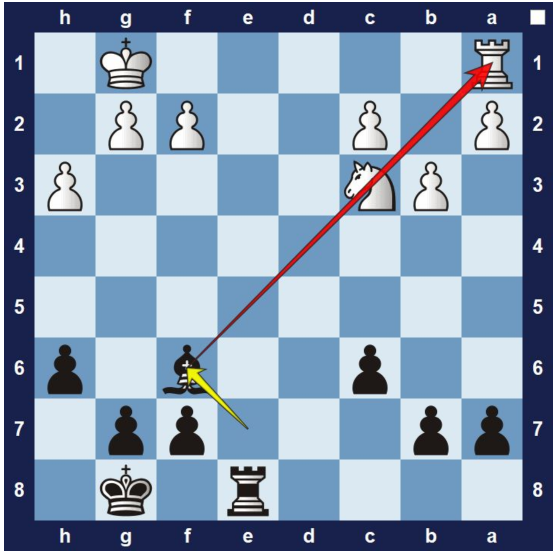
- Application of pattern-based rules to play chess in the
  - middle-game (Berliner 1975)
  - endgame (Huberman 1968)
- Heuristics to evaluate a position in classical engines
- Quality of rules measured using win rates
- We
  - define metrics to measure *goodness* of learned rules using a reference engine

Methodology

1. Chess Tactic + Model
2. Learning via ILP
3. Tactic Utility Metrics

Methodology: Chess Tactic Model

- *A chess tactic is a maneuver that takes advantage of short-term opportunities* (Seirawan 2005)
- E.g., fork, pin, skewer, x-ray, windmill, deflection
- Important concept in chess training and education[8]



Example of a pin in chess [7]

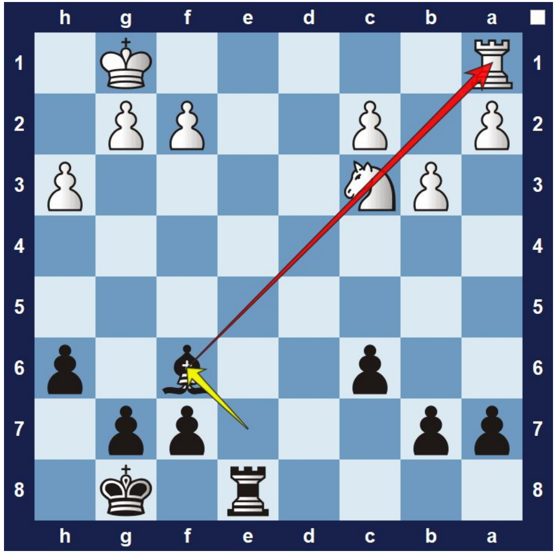
Methodology: Chess Tactic Model

- Model chess tactic as a *pattern-action* rule
- If *pattern* detected in current position, suggest move(s) *action*
- Implemented as a first-order logic rule in Prolog

```
tactic(Position) ←
    matches(Position),
    !,
    suggested(Move1,Move2,··· ,MoveN),
    legal(Position,Move1),
    legal(Position,Move2),
    ⋮
    legal(Position,MoveN).
```

Prolog-like representation of our chess tactic model

Methodology: Chess Tactic vs. Model



Example of a pin in chess [7]

```
pin (S1,P1,(X1,Y1),S2,king,
    (X2,Y2),S2,P3,(X3,Y3),(X4,Y4),Pos1) ←
  sliding piece(P1,(X1,Y1),Pos1),
  make_move(S1,P1,(X1,Y1),(X4,Y4),Pos1,Pos2)
  sliding piece(P1,(X4,Y4),Pos2),
  stale(S2,P3,(X3,Y3),Pos2),
  threat(S1,P1,(X4,Y4),S2,P3,(X3,Y3),Pos2),
  in_line (S2,king,(X2,Y2),S2,P3,
        (X3,Y3),S1,P1,(X4,Y4),Pos2).
```

The pin tactic model

Methodology: Learning using ILP

- Tactic model learned using inductive logic programming (ILP)
- symbolic ML technique

$$E \quad \cup \quad B \quad \rightarrow \quad \mathrm{T} \ \ (\textit{induce})$$

E	B	T
parent(Mary,Vicky). parent(Mary,Andre). parent(Carrey,Vicky). parent(Carrey,Andy).	mother(Mary,Vicky). mother(Mary,Andy). father(Carrey,Vicky). father(Carrey,Andy).	parent(X,Y) :- mother(X,Y). parent(X,Y) :- father(X,Y).

[9]

Methodology: PAL System

- PAL system (Morales 1992) to learn chess tactic model
- Patterns and Learning
- Proposed by Eduardo Morales in his PhD thesis
- Uses *rlgg* algorithm + heuristics to construct a suitable rule from given examples

Methodology: Tactic Utility Metrics

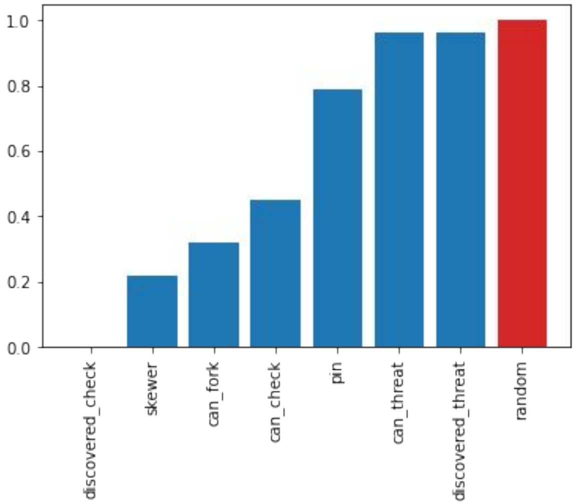
- Want to measure the *goodness* of a learned tactic
- Introduce two metrics -
  - Coverage: how *general* the tactic is
  - Divergence: how *well* a tactic approximates a reference policy
- Coverage = fraction of positions in which tactic matched
- Divergence = rank-weighted sum of the difference in q-values of a reference engine move vs. tactic suggestion

Experiment

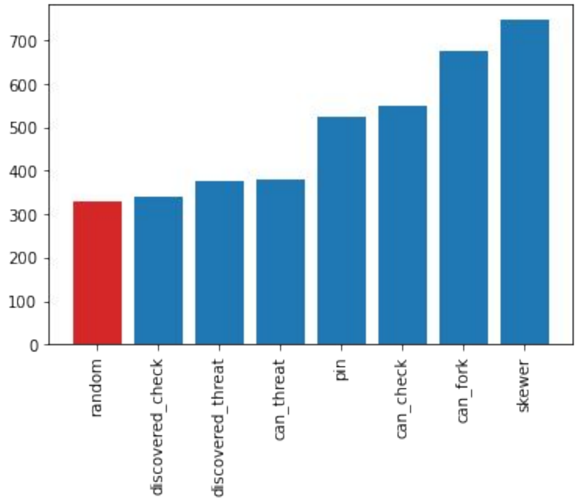
- How do the tactics learned using ILP score as per our metrics?
- Take 7 tactics from those learned in original PAL paper
- Measure coverage and divergence for each tactic
- Use both a strong (Stockfish 14) and a weak (Maia-1100) reference engine to measure divergence
- Database of positions from collection of online games on lichess.com
- Use a “random” tactic as a baseline
  - make a random legal move in the given position

Results

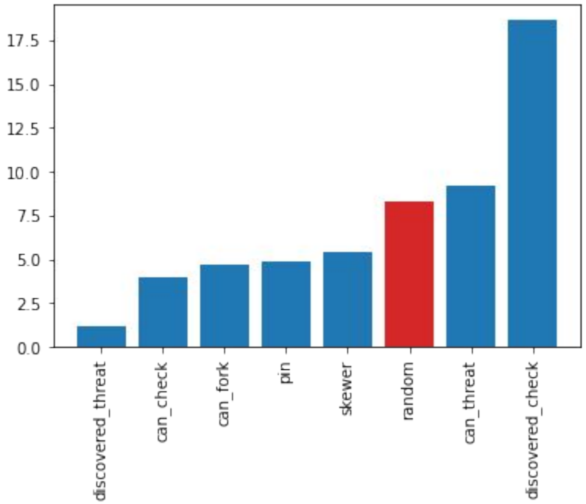
Coverage



Divergence (strong engine)



Divergence (weak engine)



Limitations + Future Work

- *Interpretability not verified with user study* - measure ease of learning and applying the tactics in real games
- *PAL system does not attempt to minimize divergence* - Improved chess tactic learning algorithm which minimizes divergence
- *PAL system requires manual guidance towards target concepts* - Improved chess tactic learning algorithm which can automatically learn from a training set of examples
- *Chess tactic model can only represent a single move* - extend the model to represent an entire game tree

Thank You!

Get in touch at

- akrish13@ncsu.edu
- abhijeetkrishnan.me

Sources

[1] https://www.thesprucecrafts.com/chess-strategy-tactics-4102130

[2] https://franz.cgsociety.org/pqpw/personal-robot-04

[3] https://liquipedia.net/starcraft2/TLO

[4] https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[5] https://svw.no/en/chess-blog/dubai-expo-fide-world-championship-match/

[6] https://www.newinchess.com/media/wysiwyg/product_pdf/872.pdf

[7] https://chessfox.com/pins/

[8] https://www.chess.com/article/view/importance-of-tactics-part-ii

[9] https://wiki.ubc.ca/Course:CPSC522/Inductive_Logic_Programming

Appendix: Tactics Learned

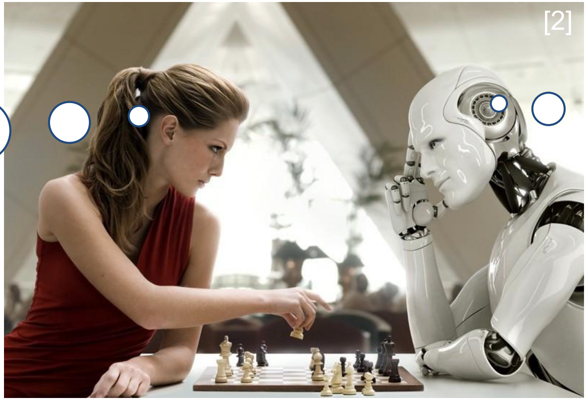
1. can threat
2. can check
3. can fork
4. discovered check
5. discovered threat
6. skewer
7. pin

Appendix: Why ILP?

- *Prior work on learning chess rules uses ILP* - good for learning chess rules
- *Prior work in explainable RL using programmatic policies* - good for providing explanations
- *Build on existing knowledge* - previously learned tactics can be used to learn new tactics

Motivation

After I move here, my opponent will most likely recapture, in which case I won't immediately recapture but play the intermediate move of f4, which will block off the dark-squared bishop allowing my pawn complex to become more powerful in the long run. This gives me excellent attacking opportunities along the queen-side, and given the low synergy of my opponent's pieces, I should almost certainly be advantageous in this position. There might be a possibility that with a sacrifice, my opponent generates some counterplay. Let me calculate the line after the bishop sac...



0100110001100101011001010110
110001100001001000000001101100
0110111101101111011010110110
010101100100001000000001100010
0110010101100001011101010111
0100011010010110011001110101
01101100001000000011101000110
1111011001000110000101111001
001011100010111000101110

Learning from our robot overlords