Chess Endgame Classifier using Machine Learning

Sutanu Gayen
Dept of Computer Science and Engineering,IIT Kanpur

Advisor : Prof Amitabha Mukerjee

April 22, 2012

1 Introduction

Chess has always been an attraction for mankind for it's apparently simple rules but difficult decision making required at different stages of the game.Psychologists have extensively studied chess to look into human cognitive processes and perceptions of the board associated with the game.In the last part of previous century AI scientists were very eager to build a chess engine that can exibit typical human-like chess intelligence. They were successful performance-wise but the human-like perception or decision-making were not at all present in those engines.Realising the non-triviality of the problem,Alexander Kronrod, a Russian AI researcher, said 'Chess is the Drosophila of AI.'Some of them tried to build engines that depends on learning but the then tools for learning approach were limited.After the explosion of machine learning research in last few decades that gave birth to algorithms like SVM,chess AI was revisited.In this paper a machine learning approach towards king-pawn endgames is presented.

2 Classical approaches to chess AI

Heuristics has always been the bread and butter of AI game researchers and to build a chess engines they also adopted a similar approach.From a give borad position,they constructed game tree consisting of branches leading to all possible half-moves(ply),searched the tree brute-force using min-max algorithm to select the best move based on some heuritics.They also used α-β,δ etc prunning techniques;fail-high,null-move etc reductions to reduce the potential branches for checking.But even then the amount of computation required were gigantic to come up with best move as evident from following case studies.

2.1 Case study : IBM deep blue

Finally efforts of AI researchers were successful when IBM deep blue defeated garry kasparov in 80's.But the huge amount of computational resources taken up by the 1997 version is evident from a survey paper[1]. It had 480 chess chips each searching 2-2.5 million chess positions/s reaching about 1 billion chess positions/s. It had non-extended search 12-ply deep and extended search 20-ply deep.The chess chips were divided into 4 parts: move generator,smart-move stack,evaluation function,search control.

2.2 Case Study : Academic chess engines

The following diagram taken from a paper[2] by Newell,Shaw,Shanon describes early chess AI engines and their characteristics :

	TURING	LOS ALAMOS Kister, Stein, Ulam, Walden, Wells	BERNSTEIN Roberts, Arbuckle, Belsky	NSS Newell, Shaw, Simon
• Vital statistics				
Date	1951	1956	1957	1958
Board	8 × 8	6 × 6	8 × 8	8 × 8
Computer	Hand simulation	MANIAC-I 11,000 ops/sec	IBM 704 42,000 ops/sec	RAND JOHNNIAC 20,000 ops/sec
• Chess program				
Alternatives	All moves	All moves	7 plausible moves Sequence of move generators	Variable Sequence of move generators
Depth of analysis	Until dead (exchanges only)	All moves 2 moves deep	7 plausible moves 2 moves deep	Until dead Each goal generates moves
Static evaluation	Numerical Many factors	Numerical Material, Mobility	Numerical Material, Mobility, Area control, King defense	Non-numerical Vector of values Acceptance by goals
Integration of values	Minimax	Minimax (modified)	Minimax	Minimax
Final choice	Material dominates Otherwise, best value	Best value	Best value	1. First acceptable 2. Double function
• Programming				
Language		Machine code	Machine code	IPL-IV, interpretive
Data scheme		Single board No records	Single board Centralized tables Recompute	Single board Decentralized List structure Recompute
Time	Minutes	12 min/move	8 min/move	1-10 hrs/move (est.)
Space		600 words	7000 words	Now 6000 words, est. 16,000
• Results				
Experience	1 game	3 games (no longer exists)	2 games	0 games Some hand simulation
Description	Loses to weak player Aimless Subtleties of evaluation lost	Beats weak player Equivalent to human with 20 games experience	Passable amateur Blind spots Positional	Good in spots (opening) No aggressive goals yet

2.3 Case Study : State-of-the-art engines

We took a look at contemporary fruit 2.1 chess engine that is used in gnuchess till now.It had features like Null-move prunning,nullmove reduction,verification search,verification reduction,history prunning,history thresold, delta margin,quiescence check plies,evaluation percentage etc and it uses classical search techniques with material value associated with the pieces.

3 Drawbacks of classical Approaches

As evident from the above case studies,the classical approaches were very computation intensive and independent of the given board position.Also the possible moves checked at each move(millions) were far larger than those of human beings(< 100).Also psychological studies have shown that human chess players percieve board not consisting of individual pieces but dynamic complexes that emerges out of individual pieces and signifies attack,defence,supports, threats etc.If you ask GM's how they come up with such complexes they donot have a clear cut answer but it is based on their intuition develoed out of huge experience.They have a large encyclopedia of chunks wich are treated like a single entity.This prompts us to take a machine learning approach in the context of chess.This approach investigates the junction between human-like and machine-like processing which is bread and butter of AI.In a classical paper by Linhares [3],he points out a set of wishlist that should be part of such an intelligent engine :

- it should evaluate large number of moves and search the game tree exhaustively only rarely.
- it should concentrate it's attention to important pieces in chess relation.
- it should have familiarity with real board positions while having difficulty interpreting implausible board position.
- it should have a short term memory with small chunks whereas a large encyclopedia of chunks in long term memory.
- it should exhibit bottom-up parallel processing strating from an arbitrary position.
- it should exhibit top-down processes triggered by expectations to come up with coherent and meaningful description of the board
- it should construct dynamic complexes out of pieces,relations among them and empty squares.

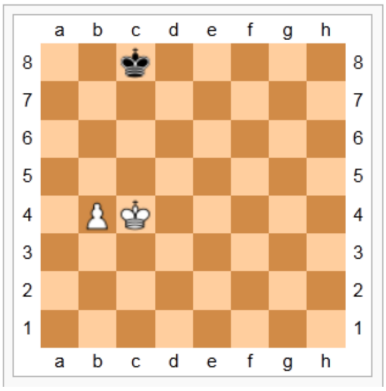
He also points out how copycat like higher cognitive representation if introduced into chess to come up with active symbols satisfies those wishlist.

4 King-pawn Endgames

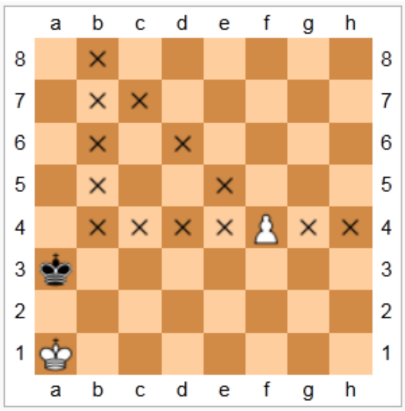
King pawn vs king engames are prevalent in chess.GM's quit if they see any move leading to such endgame and even good ameatuer players tell win/draw from looking at an arbirary positions by means of some rules described below:

4.1 Some standard rules

- rule of square : if you draw two imaginary squares from the position of pawn with length of side as the distance to promotion in both sides of the paws and if the opponent king can enter any of the square,it's a drawn game

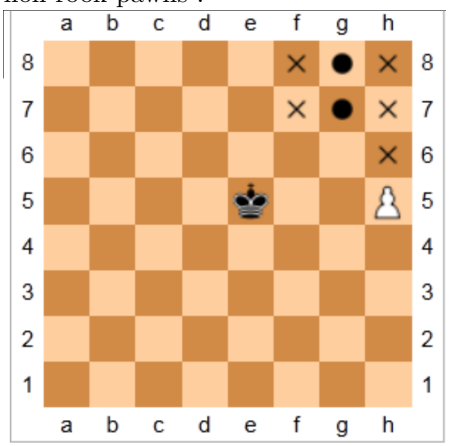


- rook pawns :



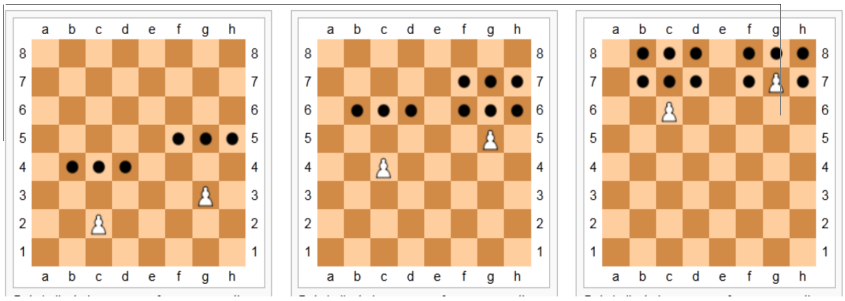
if opponent king can take any of the cross positions it's draw , if it's own king can take any dot positions ,it's a win.

- non rook pawns :



if white king takes any of the dot positions, it can guide pawn towards promotion

- taking opposition :



if distance between the kings is two,the colour that moves second wins.

- With only one exception , if black gets in front of or next to next square its a draw.
- White wins if at least any two of the followingconditions are met:
  - his king is in front of the pawn
  - he has the opposition
  - his king is on the sixth rank

4.2 past work : Experiments with AQVAL

In the 70's University of Illinois came up with some predicate based predictor program AQVAL. In a paper[4] by Michalski and Negri they used proposed distinguish between different levels(3) of learning depending on what the program has to know and what it should be able to do.The AQVAL program had the following features :
AQ7 : infers an optimized description of one decision class in relation to other classes, based on given event sets

AQ8 : determines an optimized description of each decision class seperately constrained by degree of generalization

AQ9 : optimizes a given set of DVL formulas according to a certain optimality functional

SYM-1 : determines symmetry in variable-valued functions

It produced a decision tree that had nonterminals as predicates of positions terminals as ¡value,action¿ ordered pair where value was win/draw/undefined.The program search the tree depth first till a defined leaf was found.Knowledge was represented by predicates as well as order of them.The researchers used a set of 17 predicates and found some interesting patterns like : "White wins when it is white's turn ,the pawn has a rank and is not a rook pawn and the king is on the aquare immediately behind the pawn and on the same coloumn".The accuracy of their approach is depicted below in 2 experiments(images are taken from the same paper):

	Drawing Positions			Winning Positions		
	Correct	Incorrect	Undecided	Correct	Incorrect	Undecided
First Formulas	86%	5%	9%	86%	4%	10%
Second Formulas	83%	12%	5%	79%	12%	9%
Both Formulas Together	80%	4%	16%	80%	4%	16%

Learning Events	Drawing Positions			Winning Positions		
	Correct	Incorrect	Undecided	Correct	Incorrect	Undecided
200 random events	82%	8%	10%	92%	5%	3%
200 random events + 54 selected events	85%	6%	9%	83%	4%	13%

But needless to say they didn't had the luxury of using modern machine learning techniques like SVM which apperaed in 1995.

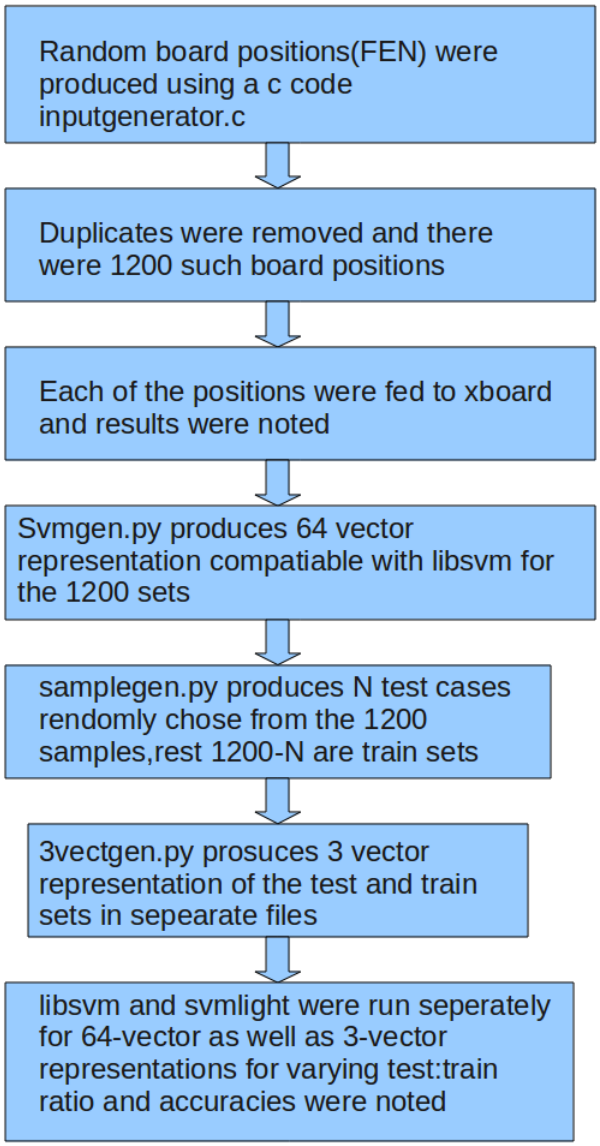
4.3 Our approach

Our target was to essentially follow the footprints of AQVAL and use machine learning tool SVM to classify new board positions as win/draw by analyzing patterns of known win-draw positions.We played 1200 such endgames in xboard and noted win-draw results.We represented a board position as 64-dim vector representing each cell as blank/black king/white king/white pawn.Also we represented it as a 3-dim vector with the dimensions specifying the cell number of each of the pieces in some order.We feed such vector to SVM and see if it can come up with any higher cognitive representation by statistical analysis on train data as described by Linhares.

5 Methodology and Results

5.1 Methodology

The following flow chart shows methodology we adapted :



5.2 Results

The following table shows the accuracy of predictions of our approach for different sets of data on a single run.

test:train	train(win:draw)	test(win:draw)	64-dim vector accuracy	3 dim-vector accuracy
600:600	376:224	374:226	61.83	63.17
700:500	424:276	326:174	64.4	67.4
800:400	507:293	243:157	60.5	61.75
900:300	547:353	203:96	65.67	70.33
1000:200	612:388	138:62	68	71
1100:100	686:414	64:36	63	67
1150:50	722:428	28:22	52	60
1175:25	733:442	17:8	68	64
1190:10	742:448	8:2	80	80
1195:5	748:447	2:3	40	40

5.3 Interpretation of results

Performance of our approach may be poor till now but that doesn't mean the approach is wrong.Think of the people who came up with the standard rules described above.They did so by exposure of a lot of endgames.So number of train data is a determinant of the accuracy of classifier.Also we can train with all 1200 samples and check accuracy of prediction with specific kind of random inputs not present in the train set like with win positions by rule of squares or by taking the oppposition or rook pawn rule.Due to lack of time this was not done but this will definitely provide the potential of this approach.

5.4 Shortcomings and Improvements

- the endgame has an upperbound of 64*63*62 positions but we used only 1200 such due to lack of time.More data will result in more accurate classifier.
- Any engine that can just run a number of board positions like batch processes and record the win/draw results in a file would automate the data collection thereby reducing time substantially
- PCA or any such tool may be used to come up with core set of components of vectors from 64-dim representation that do not lose meaningful representation of the board and then feed to svm as the vector representation we used is really sparse
- boosting based SVM classifier can be done using polling techniques
- we used RBF in svm but properly self-defined kernel functions can improve the results.

6 Credits

- All chess board images are taken from wikipedia.org
- We used python extensively at different parts of our pproject
- We used Xboard gui two machines mode to obtain results
- We looked at the GNU chess engine fruit notes and source code to know about their algorithm
- We used libsvm as the support vector machine .

- I also tried svmlight but result was more accurate for above.
- Finally a special thanks to Prof. Amitabha Mukerjee and Ankit Gupta for their guidance.

References

[1] Feng-Hsiung Hsu, *IBMS Deep Blue Chess grandmaster chips*, IEEE, Pages 70-81,Volume:19,Issue:2,Mar/Apr 1999.

[2] Allen Newell,J.C. Shaw,H.A. Shanon, *Chess Playing Programs and the problems of complexity* IEEE, Pages 320-335,Volume:2,Issue:4,Oct 1958

[3] Alexandre Linhares, *An Active Symbols Theory of Chess Intuition* Springer, Pages 131-181,Volume:15,Number:2,Oct 2007

[4] R.S.Michalski ,Pericles Negri *An Experiment on Inductive Learning in Chess End Games* Department of computer science,University of Illinois at Urbana-Champaign