Predicting Human Decision-making in Games

Who? Charles Zheng

From? Stanford University

When? May 4, 2016

History of Chess

Shatranj (≈500 AD)

Chess (1450 AD)

Shogi (≈1500 AD)

Doubutsu Shogi (Animal Shogi)

2009, Madoka Kitao


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
LET'S CATCH THE LION!


A
B
C
1
2
3
4
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!

A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!


A
B
C
1
2
3
4
どうぶつしょうぎ
LET'S CATCH THE LION!

What is a game?

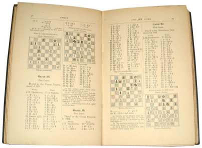
- Set of states $s \in S$.
- Set of winning states for player 1, $W_1 \subset S$.
- Winning states for player 2, $W_1 \subset S$.
- Each state has a set of legal actions $\mathcal{A}(s)$.
- Player 1 and player 2 take turns choosing the action.
- A *transition function* $P_a(s, s')$ determines the next state s' resulting from taking action a in states s.
- In *deterministic games*, the transition function equals one for one s' and zero otherwise.

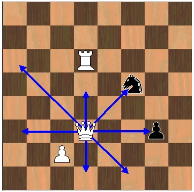
Recorded Game Data

- List of moves made by both players: "1. Pawn from e2 to e4, 2. Pawn from e7 to e5."
- Convert to list of *state-action* pairs, (s_i, a_i).
- s_i is state of game at the beginning of turn i, a_i is the move selected at turn i.

The prediction problem

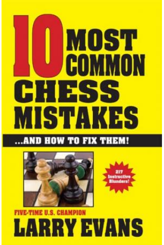
- *Database* of games from players $p = 1, \ldots, m$ with:
  - Date played
  - Player 1 and player 2 identities
  - List of moves $\rightarrow$ player, state, action pairs (p_i, s_i, a_i)
- In a *new game*, can we predict which move a given player p will make, given the game state s?
- In other words: given (p, s), guess $a \in \mathcal{A}(s)$.





Motivation

- First step for building superhuman AI player
- Detecting computer-aided cheating (Regan 2013)
- Objectively evaluating professional players (Matej 2011)
- Automatic commentary
- Writing a book about 'most common chess mistakes'



- Use machine learning to learn about human learning (?)

Section 2

Methods

Features

- In order to apply machine learning, need numeric features for the state s.
- Let $x_1(s), \ldots, x_p(s)$ denote the features, $\vec{x}(s)$=feature vector.
- Borrow features from chess programming: material, board control, king safety, etc.?
- Or try to do generic *feature selection* or *representation learning*?

Features

We use a minimalistic featurization. The 3x4 board is converted into a 136-length binary vector.

x_1	Player 1 has a king at (1,1)?
x_2	Player 1 has a rook at (1,1)?
x_{23}	Player 2 has a bishop at (1, 3)?
x_{132}	Does player 1 have two rooks in hand?

We also consider *second-order interactions*: e.g. $x_4 x_{23} =$ Does player 1 have a pawn on (1,1) and player 2 have a bishop on (1, 3)?

Policy model

- Fit the model

$$\Pr[A = a|s] = \frac{\exp[\beta_a^T \vec{x}(s)]}{\sum_{a \in \mathcal{A}} \exp[\beta_a^T \vec{x}(s)]}.$$

  where $\mathcal{A}$ is the set of *all possible moves* in the game (not just legal moves in s).

- When *predicting*, choose the action with the highest predicted probability among *legal* actions

$$\hat{A}(s) = \max_{a \in \mathcal{A}(s)} \beta_a^T \vec{x}(s).$$

Policy and values

- A policy $\pi(s, a)$ specifies a probability distribution of *actions* for each state $s \in S$.
- The value of a state $V^{\pi_1,\pi_2}(s)$ is the probability that player 1 wins if player 1 uses policy π_1 and player 2 uses policy π_2.

$$V^{\pi_1,\pi_2}(s) = \begin{cases} 1 & \text{if } s \in W_1 \\ 0 & \text{if } s \in W_2 \\ & \sum_{a \in \mathcal{A}} \pi_i(s, a) \sum_{s'} P_a(s, s') V^{\pi_1,\pi_2}(s') \end{cases}$$

where $i = 1$ if it's player 1's turn and $i = 2$ if it's player 2's turn.

Evaluation functions

- In a deterministic game, let $s'(s,a)$ denote the state with probability 1 resulting from (s,a)
- Suppose player 1 knows player 2's policy. Then it would be rational for player 1 to choose a which maximizes the
$$V^{\pi_1,\pi_2}(s'(s,a)).$$
- However, humans are not perfectly rational nor omniscient. Perhaps players have an intuitive *evaluation function* E which approximates the true value function,
$$E(s) \approx V^{\pi_1,\pi_2}(s).$$
- This is also how chess engines work–idea first proposed by Shannon (1949)

Evaluation model

- Suppose a player does have a mental evaluation function E. Should they always choose a which maximizes $E(s'(s,a))$?
- Even in perfect information games, there is an advantage to being unpredictable!
- This leads to a multinomial model of player choice:

$$\Pr[A = a|s] = \frac{\exp[E(s'(s,a))]}{\sum_{a' \in \mathcal{A}(s)} E(s'(s,a'))}$$

Note that $E(s)$ need not be a probability: it could be a real number. Real-valued $E(s)$ may be more realistic, anyways.

How to fit the evaluation model?

- For fixed player p, let $\{(s_i, a_i)\}_{i=1}^n$ denote all the state-action pairs for that player in the database.
- Fit a logistic evaluation model, minimizing the loss

$$-\sum_{i=1}^{n} \vec{x}(s'(s_i, a_i))^T \beta - \log(\sum_{a \in \mathcal{A}} \exp[\beta^T \vec{x}(s'(s_i, a))]).$$

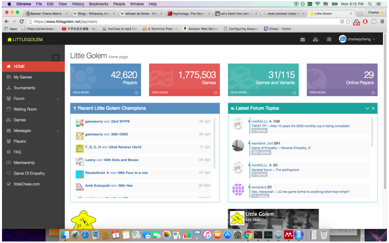
- Loss function is convex!
- For large feature models, recommended to use L1 and L2 regularization.
- To predict, choose $a \in \mathcal{A}$ maximizing $\beta^T s'(s, a)$.

Section 3

Data

Doubutsu Shogi on LittleGolem

- Data obtained from `littlegolem.com`, a free turn-based game site, using `scrapeR`
- 85 players, 727 games, 17094 turns (state-action pairs)
- Oct 2012 to Apr 2016
- Usernames have been anonymized



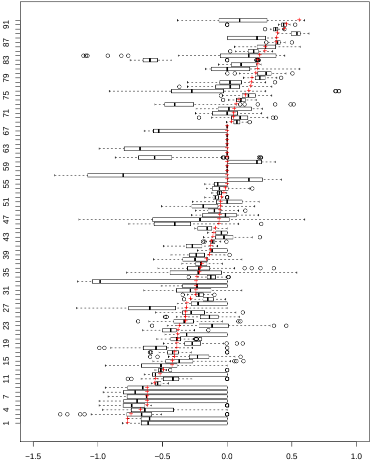
Ranking the players

- θ_i parameterizes the skill level of player i

$$\Pr[i \text{ wins} | i \text{ vs } j] = \frac{\exp[\theta_i - \theta_j]}{1 + \exp[\theta_i - \theta_j]}.$$

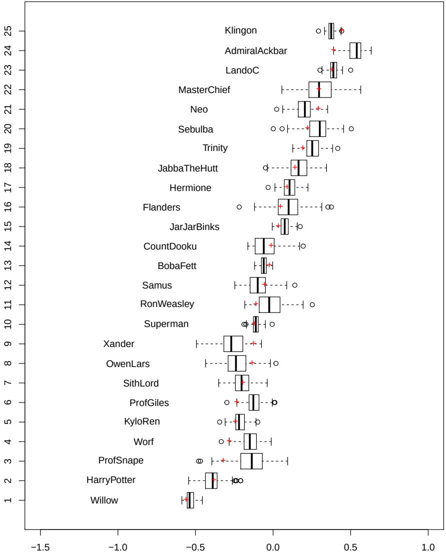
  Bradley-Terry model
- A spread of $\theta_i - \theta_j = 1$ means a 73% win rate for i.
- Fit using glmnet: add some regularization
- Bootstrap to get 'confidence' intervals

All players



91
87
83
79
75
71
67
63
59
55
51
47
43
39
35
31
27
23
19
15
11
7
4
1
−1.5
−1.0
−0.5
0.0
0.5
1.0

Players with > 5 games.



How close are we to perfect play?

The game is a theoretical loss for player 1. We looked at games where both players are above a given skill threshold... does the win % approach the theoretical value?

Group	# of games	% win for P1
All	701	0.48
$\theta > 0$	119	0.44
$\theta > 0.2$, > 5 games	19	0.31
$\theta > 0.3$, > 5 games	10	0.40
Theoretical		0

Move prediction

First step: lump all of the players into one group. How well can we predict?

Method	No. params	Accuracy
Markov model	186	0.441
Policy, order-1	25k	0.565
Policy, order-2	1.7m	0.559
Eval, order-1	136	0.370
Eval, order-2	9316	0.559
Eval, order-3	419k	0.439

Note: regularization was done in an ad hoc way.

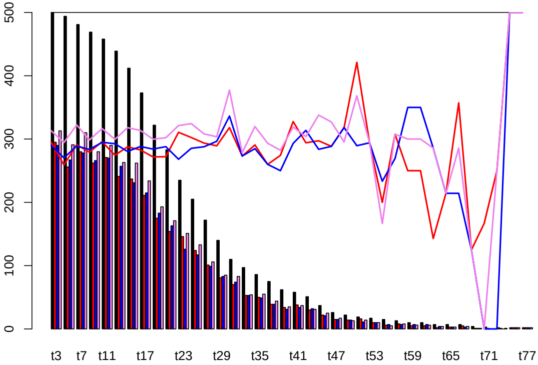
Move prediction

What happens if we combine the policy and evaluation models?

$$\hat{\Pr}[a|s] = 0.5\hat{\Pr}_{policy}[a|s] + 0.5\hat{\Pr}_{eval}[a|s]$$

Method	No. params	Accuracy
Policy, order-1	25k	0.565
Eval, order-2	9316	0.559
Ensemble	34k	0.611

Move prediction accuracy by turn



Policy (red), Eval (blue), Ensemble (violet)

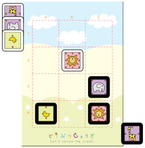
Validating the evaluation function

- The learned eval function should estimate some monotonic function of the probability of a win.
- Take game states near the ends of games:
  - if player 1 won: $E(s)$ should be high.
  - if player 1 lost: $E(s)$ should be low.
- This gives a somewhat independent validation, since the win vs lose information was *not* used to fit the evaluation model!
- We apply this for the second-order model (9316 params, 0.559 accuracy)

Validating the evaluation function

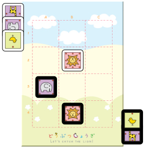
Win

12.02



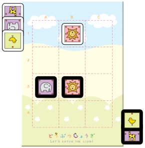
6.32

14.29



Lose

13

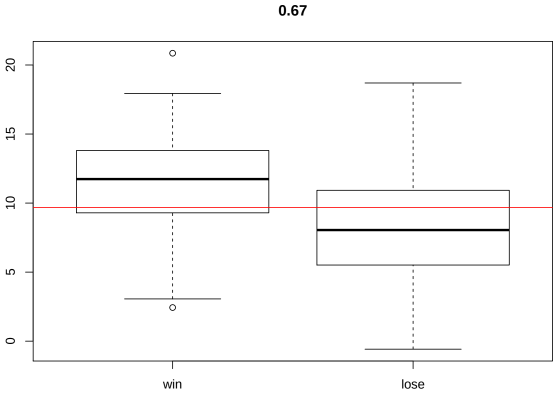


6.13

7.13

Validating the evaluation function

Game states 3 or 4 moves from the end of the game, on Player 1's move.



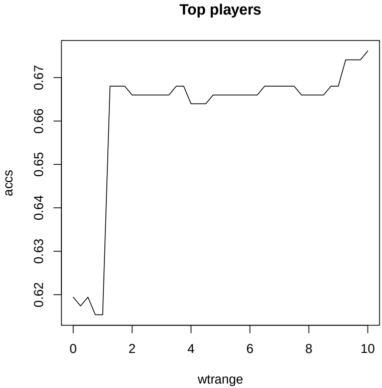
Personalized Prediction

- Data (p_i, s_i, a_i) where p_i is the player
- Let P be a set of 'similar' players.
- We would like to predict moves made by players in P.
- Fit the evaluation model by minimizing

$$-\sum_{i=1}^{n} w_i(\vec{x}(s'(s_i, a_i))^T\beta - \log(\sum_{a\in\mathcal{A}} \exp[\beta^T\vec{x}(s'(s_i, a))])).$$

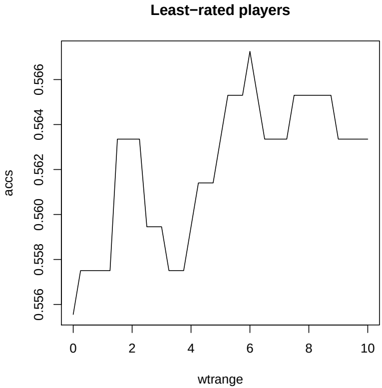
- Set $w_i = 1 + w$ if $p \in P$ and $w_i = 1$ for $p \notin P$. (Then normalize so $\sum w_i = n$).
- w controls the extra weight given to players in P.

Personalized Prediction



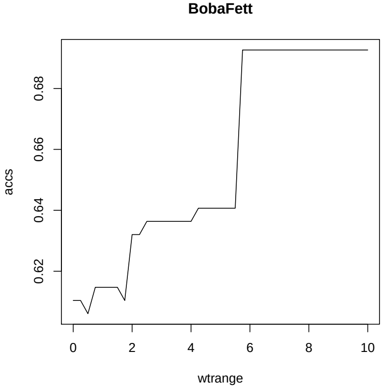
Players with $\theta > 0.2$ and more than 5 games (6 players).

Personalized Prediction



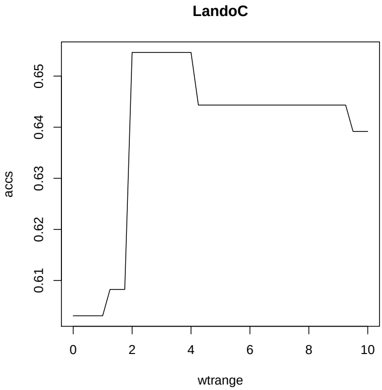
Players with $\theta < 0.2$ and more than 5 games (16 players).

Personalized Prediction



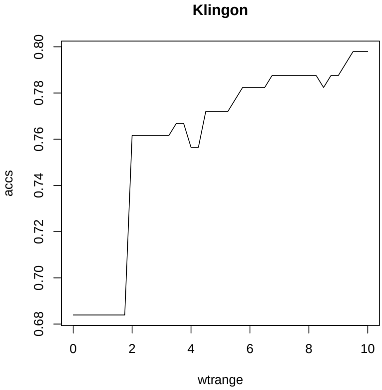
BobaFett, 78 wins - 118 losses, $\theta = -0.02$.

Personalized Prediction



LandoC, 108 wins - 7 losses, $\theta = 0.38$.

Personalized Prediction



Klingon, 98 wins - 7 losses, $\theta = 0.44$.

Conclusions

- Two approaches for prediction problem: policy model and evaluation model.
- Policy model works well for the most part, but may fare worse in new situations.
- Evaluation model may be more parameter-efficient.
- Ensemble works better than either model.
- Second-order evaluation model could likely be improved, based on validation results.
- Hypothesis: better players are more predictable than worse players?

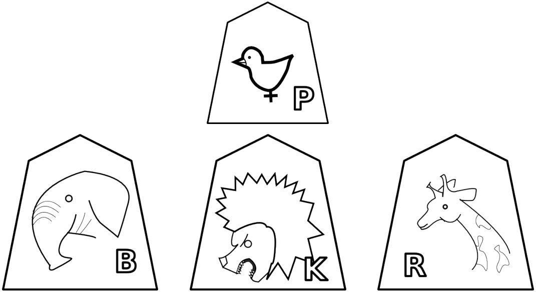
Future play

As opposed to future 'work'.

- More games! More complex games and different types of games!
- Decision trees might be very, very good!
- 'Regularize' the evaluation functions using the assumption of self-consistency

$$E(s) \approx \min_{a \in \mathcal{A}(s)} \max_{a' \in \mathcal{A}(s')} E(s'(s', a')).$$

- Track player improvement over time.
- Our predictive models should be better than using a computer player to predict human moves, right?



Thanks!