Artificial Intelligence as Structural Estimation: Economic Interpretations of Deep Blue, Bonanza, and AlphaGo

Timothy Schwieg

February 2, 2019

Introduction

- Artificial Intelligence and Machine Learning are often seen as black box techniques with no interpretability.
- Implementations of these methods is often an implementation of a dynamic structural model.
- Case studies for three famous Game AIs: All board games of perfect information
- Chess: Deep Blue (Calibrated Value Function)
- Shogi: Bonanza (Estimated Value Function - Harold Zurcher)
- Go: AlphaGo (Conditional choice probability and condition choice simulation)

Why Board Games?

- Perfect information but extremely large state spaces.
- Cannot be solved via backwards inductions with modern computing power.
- One of the best known and successful applications of machine learning.
- Humans can interpret the results and quantify success and failure much more easily.
- Economically, there is a single valued best response function to any state space.

Rules

- Two players take turns making moves in discrete time. Let a_t denote the actions of the moving player at time t.
- State transition is deterministic: $s_{t+1} = f(s_t, a_t)$
- Action state is finite, determined by "legal moves" $a_t \in \mathcal{A}(s_t)$
- State space is finite, and can be one of four outcomes, continue, draw, win, loss. Defined from the perspective of player 1.

$$u_1(s_t) = \begin{cases} 1 & \text{if } s_t \in S_{win} \\ -1 & \text{if } s_t \in S_{loss} \\ 0 & \text{otherwise} \end{cases}$$

How does deep blue work?

- Three components: Evaluation function, search algorithm, databases.
- Evaluation function is a linear combination of observable characteristics of the state space s_t.

$$V_{DB}(s_t,\theta) = \theta_1 x_{1,t} + \theta_2 x_{2,t} + ... + \theta_K x_{K,t}$$

Where θ is a vector of K parameters, and x_t is a vector of K observable characteristics of s_t

- Published version has $K = 8150$.
- Chosen by expert knowledge - multiple grand masters manually adjusted the parameters of the model.

Search Algorithm and Databases

- We would like to use backwards induction from end-game positions, but this is computationally infeasible.
- Instead of searching completely, it searched possible moves for a fixed number of moves, as well as pruning moves that are known to be bad.
- Databases were all possible five-piece position end-games solved as well as some six-piece end-games.
- Opening book - Collection of the optimal way to play each of the common openings.
- Four grand-masters constructed an opening book of 4000 openings by hand.

What move should I make?

- If we had the optimal value function, then we would never need to look forward to solve a game, because this value function would have embodied this already.
- However our value function can only ever be an approximation to this optimal one.
- Approximation errors are handled by the search, which conducts backwards induction on the truncated version of the game tree.
- How should we construct the approximation of the value function?

The evaluation function is a Calibrated Value Function

- Obtain a value function that at each point generates behavior consistent with ideal play.
- It should never suggest a move that the value function itself can beat.
- Differences between calibrating this model and a macro model lie only in the goals and magnitude
- Many more parameters
- More interested in prediction rather than testing fit. (This motivates the larger number of parameters)
- At the end of the day, these parameters are still chosen to produce output that is consistent with some goal.

What is shogi?

- Commonly called Japanese chess or Game of Generals
- Pieces can return to the board: $|S|$ does not decrease over time.
- Many pieces have significantly less mobility than in chess
- There are more types of pieces
- The board is larger
- These factors all combine to generate a state space of

$$|S_{shogi}| \approx 10^{71}$$

How to approximate the value function?

- Kunihoto Hoki factorized he state space into 50 million variables.
- Based on relative positions of clusters of three pieces.
- This is combined into a linear valuation function

$$V_{BO}(s_t|\theta) = \sum_{k=1}^{K} \theta_k x_k$$

Want to choose the action that maximizes this value function in some future $t + L$ turn.

$$a_t^* = arg \max_{a \in \mathcal{A}(s_t)} \{V_{BO}(s_{t+L}|\theta)\}$$

How big is too big?

- This is much too large to ever try manual calibration for predictive accuracy.
- Instead this value function is approximated from professional play.
- Only 50,000 games with 100 moves on average. (Figure out how he handled this)
- Discrete choice regression is then used to estimate the values of θ.
- In effect, we are estimating a dynamic discrete choice problem.

Who is this Harold Zurcher?

- Harold Zurcher - Want to predict the actions of the superintendent for maintenance of Madison Metropolitan Bus Company
- Nested Fixed Point algorithm comprises of finding θ values that make the model's predictions fit the observed data, and then solves the model to find the corresponding policy function.
- First portion is solved via Maximum likelihood estimation, and the second using value function iteration. (NFXP algorithm)

Bonanza is Harold Zurcher

- The goal of bonanza is to estimate the policy function, which ideally would be the best response function. Why is it not?
- Bonanza follows a two-step procedure as well:
- Step One: Estimate θ via logit-regressions. This is the same as Rust (1987).
- Step Two: Estimate the optimal strategy for the truncated game tree with the value function on the leaves, assuming the opponent has the same value function.
- While the second step differs from Rust (Two players) in both cases, there is a unique optimal strategy that both models are solving. See Igami (2017,2018) for proof of how the NFXP extends to these games.

Go

- Chess and Shogi utilized parameterizations of the state spaces to construct approximations of the optimal value function.
- Very difficult to do this for Go.
- State space is huge: 19×19 board, with $|S| \approx 10^{171}$.
- Very difficult to articulate what makes good play versus bad play
- One advantage is that Go becomes simpler in the end game.
- Moves are just placing a stone in an open space, and the final positions are unambiguous.

AlphaGo

- Combines four different estimation techniques in an ensemble.
- Supervised Learning of a Policy Network
- Reinforcement Learning
- Value network
- Monte Carlo Search Trees

Supervised Learning of the Policy Network

- Deep neural network to predict professional players' moves from a state.
- Policy function using a Convolution Neural Network and 4.6 million weights (parameters)
- Learned from online database of 160,000 games leading to 256 million states. This is virtually nothing compared to the state space size.
- Goal is to estimate the probability of victory based on the logit.

$$\Pr(a_t = j | s_t, \theta) = \frac{\exp(y_j(s_t, \theta))}{\sum_{j' \in \mathcal{A}(s_t)} \exp(y_{j'}(s_t, \theta))}$$

Where $y_j(s_t, \theta)$ is the latent value of choosing action j in state s_t for the parameters θ.

Reinforcement Learning of the Policy Network

- The policy network is only estimating human play. It correctly predicts human play out of sample 55.7% of the time, which is very high considering how little data are observed.
- However we are not interested in predicting human play, as much as predicting the optimal play.
- We seek a policy function based on different parameters θ' that performs better when playing against θ.

$$\Pr_{win}\left(\sigma(s_t,\theta'),\sigma(s_t,\theta)\right) > \Pr_{win}\left(\sigma(s_t,\theta),\sigma(s_t,\theta)\right)$$

Reinforcement Learning Continued

- This does not imply that the new strategy is dominant, as we are only playing it against what we previously approximated as optimal.
- This could only ever be fully addressed by solving explicitly for the optimal strategy.
- To try to combat this, the new strategy $\sigma(s_t, \theta')$ plays against many policies randomly sampled form previous rounds.
- Is this satisficing?

Evaluation Function

- This is constructed from the existing policy function that is an approximation to the optimal policy function
- Sample many games using the policy function to determine play
- Pick many random states and record the winners of each of the game.
- This generates an arbitrarily large synthetic dataset.
- This can then be used to predict the probability of winning from any location in the state space s_t.
- The policy function implies outcomes in games against itself, and then can be made explicit through simulation.
- Estimated through another deep neural network with 49 channels, 15 layers, and 192 kernels.

Monte Carlo Search Trees

- Can simulate how good a position is by simulating entire games starting from a position randomly, and counting how many of them are wins and how many are losses.
- This is very easily parallizable, and can be done in real time to evaluate a single position on the board.
- This was the state of the art for Go computing before AlphaGo, and is implemented in AlphaGo.
- Both the Policy and the Value functions are combined in MCTS to determine the play that is used by AlphaGo.

Hotz and Miller (1993)

- Wish to circumvent the curse of dimensionality in NFXP. Don't want to have to solve a dynamic program at each iteration of θ.
- Want to estimate the policy function directly from the data
- If our data is large enough, we can use the observed choice probabilities conditioned on the state to non-parametrically obtain the optimal policy function.
- Trades away the computational curse of dimensionality for the data curse instead.
- In this context, neural networks are ideal, as they can approximate any arbitrary function provided they have enough layers and nodes (Hornik, Stinchombe, White 1989)
- They also show that there is a one-to-one mapping between policy functions and value functions.

Policy network is first-stage CCP Estimator

- First-stage CCP Estimator exists to capture the actual choice patterns in the data as flexibly as possible.
- In this context, the policy network is attempting this exactly.
- The supervised-learning policy network has an out of sample move-prediction accuracy of 55%, and close to 90% in its top-five predictions.
- This can be compared to the parametric fit of the logit, which sits at a measly 27%.

Reinforcement Learning is a Counter-factual with long-lived players

- Reinforcement learning is policy function iteration.
- A policy (strategy) is assumed, and the value function is evaluated. This value function implies a new policy function, and this process is iterated.
- Consider the strategy found in the Policy network (stage 1) to be the best strategy arrived at so far in part of the life of an infinitely lived agent that encompasses all the top Go players.
- Improvements in this strategy can be considered to be aging of this agent, as he discovers new strategies that are better than the previous strategy.

Value Network is Second-Stage CCS Estimates

- HMSS (1994) Calculate the value function by simulating the first-stage CCP Estimators, these simulations imply a value function as well as its underlying structural parameters.
- This is exactly how the value network is generated by AlphaGo.
- AlphaGo is interested in beating human champions rather than studying the strategic iterations, so it ignores many dynamic-game interactions.
- BBL (2007) extend HMSS (1994) to dynamic games using a moment-inequality-based estimation. This would only become relevant for using structural methods to study the interaction of Go Engines (players).

Implicit Assumptions

- Both Bonanza and AlphaGo implicitly type-1 extreme value errors when they use logit-style discrete choice models.
- Both also assume that the error is iid across actions, players, times and games.
- This forces them to ignore several important aspects of games

1. Consideration Sets and Selective Search
2. Cross-sectional Heterogeneity
3. Inter-temporal Heterogeneity
4. Strategic interactions
5. Time/Physical Constraints

Opportunities for Future Research

- Since in games of perfect information AI is so dominant, making it "human" may be a new goal
- The Econometric Toolbox has evolved since 1987 and 1993, in particular for strategic interaction (BBL) and heterogeneity.
- Heterogeneity may be increasingly important in games of incomplete information where beliefs must be formed.
- Structural interpretability: This does not mean that will suddenly have interpretations of Neural Networks, but rather the object for which the neural network is an approximation of. (AlphaGo's Policy Network is the CCP estimate of the average professional player's strategy under the assumptions of homogeneity)
- Use of Convolution Neural Networks for non-parametric estimation of CCPs.