Machine Learning in Board Games

Neural Networks, Genetic Algorithms and Propositional Nets

Jeff Zurita
Why Board Games?

- Board games are a good proving ground for AI because
  - Complex Problems
  - Clearly defined inputs and results
  - Game environment is easily amenable to computer simulation
  - Like the Turing Test, board games are a good example of human cognition which AI would like to replicate
Game Over – You Lose!

- But aren’t board games a solved problem?
- Minimax (and other tree pruning methods, alpha beta, Negamax, SSS*, etc) and evaluation functions make this an engineering problem, not an AI problem
- Two words: Deep Blue!
Game Over – Not so fast!

• One word: Go
• The Humans Strike Back!
  • Arimmaa
  • Octi
  • Havannah

• Human Expertise vs. Machine Learning
  • That’s what AI is for
  • Humans can make mistakes!
Case Study: Samuel’s Checkers

- Program: Arthur Samuel’s Checkers (1956) [1].
- Game: Checkers
- Implementation: Polynomial evaluation function
- Learning Method:
  - “rote learning”: Iterative deepening of search tree based on past board positions
  - "learning by generalization": Polynomial Coefficients changed
- Results: Moderate strength player
  - IBM Stock jumped 15 points
Samuel’s Learning Method

- Evaluation function had 39 features, each individually weighted. Sixteen features use at any particular time.
- Learning program compared value of the current position with a future position
  - If the difference was positive, weights would increase
  - If negative, weights would decrease
  - If the gap could not be closed, features would be swapped.
- Many games played against a static version to arrive at optimal weight values

Reproduced from [2]
Case Study: TD-Gammon

- Program: TD-Gammon (1992) [3]
- Game: Backgammon
- Implementation: MLP evaluation function
- Training Method: TD(λ) aka Temporal Difference Learning
- Results:
  - Strong player
  - Revised expert opinions on game opening
The Perceptron

- Inputs: 1D Vector of board and piece positions
- Outputs: 1D, 4-element Vector containing predicted winner and whether the win was a gammon
Neural Network approximation of a function

\[ Y = \sum f([w_0] \ast [x_i]) \]

\[ Y = \sum (f([w_{1ij}] \ast f([w_0] \ast [x_i]))) \]
TD-Gammon Learning Method - TD(\(\lambda\))

- The "Temporal Difference" \(dt\) is defined as:
  \[ dt = Y_{t+1} - Y_t \]
  where \(Y\) is the neural network evaluation function of the game board at turn \(t\). If we knew the real evaluation function, then \(dt = 0\).

- At the end of the game \(Y_T\) is known. Then adjust weights matrices to minimize \(dt\) with a limiting learning parameter \(\alpha\).

  \[ W_{t-1} = W_t + \alpha(Y_{t-1} - Y_t)\nabla_w Y_T \]

- The parameter \(\lambda\) is the decay parameter, which limits the amount of weight change according to time (turns) away from the game end.

  \[ W_{t-1} = W_t + \alpha(Y_{t-1} - Y_t) \sum \lambda^{t-T} \nabla_w Y_T \]
Case Study: Blondie24

- Game: Checkers
- Implementation: MLP evaluation function
- Training Method: Genetic Algorithm
- Results: Success!
Blondie24: Evaluation Function

**Inputs:** 1D Vector of board and piece positions

**Output:** Value of the board passed to minimax (alpha beta) evaluation

Reproduced from [5]
Blondie24: Training by Genetic Algorithm

1. Generate a pool of random MLPs
2. Use MLPs in a round of games between each other
3. Rank the MLPs based on win/loss scores
4. Delete the lowest performing MLPs, keep the best performing MLPs as the basis for the next generation
5. Create a new generation of MLPs by mutating (random variation) of the weights and biases of the successful MLPs
6. Repeat

Reproduced from [5]
Blondie24: Results

- Resulting program was tested against humans in an online gaming site.
- Blondie24 ranked 18 of 44,000 registered users
- Able to draw a Master rated checkers player.
Case Study: Hex Player

- Program: Hex Player [8]
- Game: Hex
- Implementation: MLP evaluation function
- Training Method: Genetic Algorithm
Hex Player – Implementation Notes

- Board represented as a linear vector of $n^2$ elements
  - MLP included two hidden layers
- Population of 12 nets, 6 survivors in each generation
- Mutation governed by a log-normal process

$$w'(i) = w(i) + \sigma \times N(0,1)$$

where $\sigma$ is constant depending on the total number of weights

Reproduced from [8]
Hex Player: Results

- Learning was observed...
- ...but it was not human rated.

Lessons learned:
- Lots of computer time needed!
- Many tradeoffs in board size, population size, number of games, etc.
- Troubleshooting alpha beta is hard.
GA Limitations in Hex Player

- Hex has a much larger search space than Checkers
- Hex has no Piece Differential
- Hex is a *divergent* game, as is Go
  - GA applied to Go also resulted in a single strategy [9]
- Board representation is an important factor
  - Need to capture positional relationships between pieces
  - Blondie24 (checkers) [5] and Blondie25 (chess) [7] included preprocessing neural nets for key areas of the game board
Conditions Required for Temporal Displacement Learning

- Conjectures by I. Ghory [10]
- These reasons likely apply to the GA method as well.

1. Smoothness of board evaluation function.
2. Divergence rate of boards at single-ply depth.
Complexity of Select Games

Reproduced from [10]
The Frontier: Monte Carlo Tree Search

- An adversarial search method in which selected nodes are played-out and the results become part of the evaluation.
- Allows for a heuristic game playing: only legal moves and winning conditions are needed.
- Machine learning methods can be applied to the node selection step, as explained in reference [13].

Repositories from [11] and [12].
The Frontier: General Game Playing

- State machine
  - Implement the rules of a game as a state machine
  - Implemented commercially in *Zillions of Games* [14]

- Stanford GGP: Logical Proposition Nets
  - Decompose game into logical propositions
  - Marking a proposition assigns boolean values – equivalent to a game state
  - Compute legal moves
  - Compute consequences (aka new game states)
  - Compute goal achievement (aka winning)

See references [15] through [17] for online course materials
References


[9]. A downloadable version of Litho, the genetically developed Go player, s is available in this forum discussion: http://lifein19x19.com/forum/viewtopic.php?f=18&t=5368
References (cont’d)

- [12]. Monte Carlo Tree Search (MCTS) research hub, available at: http://mcts.ai/about/index.html
- [13]. “Evolutionary Learning of Policies for MCTS Simulations”, J. Petit and D. Helmbold. (Made available by personal communication with the authors).
- [17]. Coursera site for online GGP course: https://www.coursera.org/course/ggp