IntelliChess
By
Usman Hassan
Faculty Advisor: Lawrence Saul

Abstract:
IntelliChess is a chess engine that learns to play from the final outcome of games. IntelliChess learns chessboard evaluation functions to compare different boards and based on its experience, makes the best move. IntelliChess uses temporal difference to use rewards at the end of the game and to learn a value function for the different board positions encountered during the game.

Related Works:
Chess is a complex problem with many levels to explore it at. There have classically been two main categories of approaches taken to solve the problem of machine chess. The first is the more traditional, well-established and to some extent saturated approach of a brute force searching algorithm, looking just to maximize value over the next 6, 7, or 8 moves and picking the one that gives the maximal expected benefit. The other is of using a reinforcement learning technique to learn an evaluation function and therefore a decision rule for playing the game. The complexity of the reinforcement learning algorithm applied varies, and yields a variety of levels of play.

Neuro Chess by Thrun, and the works by Michael Gherrity and Iman Ghory, all focus on the reinforcement learning approach. There is a common underlying theme between their ideas and those that lie at the core of IntelliChess.

System Overview:
IntelliChess implements a reinforcement learning algorithm, in order to learn from final outcomes of games. It uses a Temporal Difference TD-lambda approach to implement an appropriate learning scheme. IntelliChess learns by example and by gaining experience and then applies what it has learnt when it plays.

IntelliChess is trained on over 20,000 thousand expert level games and basically learns how best to value a chess board. It gets a positive (+1) reward if the game is won, and a negative (-1) reward if the outcome is a loss. A neutral (0) reward is received by the system if the game turns out to be a draw.

Board Parameterization:
IntelliChess uses a board “featureization” (parameterization) scheme to reduce the dimensionality of the problem. It extracts from a chess board, values for features which are deemed to be key to the value of a board. There were two different parameterization schemes that were employed, one using 38 features and the other using 50. The two showed differing results, and playing styles.

There were two main categories for the features that were used to parameterize the board. The first category included:
- Positional & Structural: Whether the queen is developed or not, or whether the pawn structure is strong or not.
- Value Based: The total value of pieces on the board, or the value of pieces in the centre squares (center 4 and 16) and value of pieces in a certain radius around opponent’s King.

Learning Algorithm:
IntelliChess learns the weights of a linear weighted valuation function for the chessboard. A TD-lambda approach is used since the timing of the rewards is delayed and uncertain. The actual TD algorithm that is used is as below:
\[ V(s) = \alpha \cdot R(s) + \gamma \cdot V(s') \]

Different values for the learning rate alpha were employed, including an alpha-decaying scheme. Different discount factors (gamma) in the range from 0.9 - 1 were tried and the lambda parameter was also varied between 0 and 1. With the 38 feature vector the parameters that gave quickest convergence and worked best, were:
- gamma of 0.97
- alpha of 0.000001
- lambda of 0.0001

Results:
The resulting evaluation function was judged on the basis of its play against other computers such as GNU Chess and also against myself. While it was not the strongest engine and was unable to win against GNU Chess, it was playing reasonably competitive, and was winning pieces and board position at times, however it would commit some meaningless blunders as well.

Given an existing board position, IntelliChess extracts values for a pre-defined set of features in the feature vector. It calculates the value of the board by taking a linear weighted function of the feature vector (i.e. dot product of feature vector and weight vector). The output is a single number, which is a measure for the value of the board position.
- Learning stage: The weight vector is learnt through a reinforcement learning TD-lambda approach.
- Playing stage: The pre-learnt weight vector to value boards and decide its best move.

The methodology used by IntelliChess to figure out the best next move, is to iterate over all possible next states, i.e. the boards that can possibly be reached by making one legal move. It values each one of these boards, and then selects the move that leads to the one that gives the highest valuation.

Board Valuation

Pick Max Value Move

IntelliChess

Senior Project Poster Day 2005, CIS Dept. University of Pennsylvania