Literature Survey

Title: Using Genetic Algorithms to evolve a Neural Network based Draughts Player

Student: Thien Nguyen

Academic Advisor: Stefan Dantchev

Themes

- Machine Learning
- Neural Networks
- Genetic Algorithms
- Draughts (Checkers)

Definition of Terms

Genetic Algorithm - An evolutionary method that solves optimisation problems. This is based on Darwin's theorem of perpetually evolving populations of solutions.

Neural Network - A computational model based on the operations of "interconnected processing elements, which process information by their dynamic state response to external inputs." [1]

Draughts - In this project we will be using the British Draughts (American Checkers) rules.

For the sake of clarity these rules will be enforced:

- Jumps are enforced
- Multiple jumps are enforced
- In the event that a piece performs a multiple jump, if it lands on a promotion row (where a piece is promoted into a king), then its move is terminated.

Ply - a ply refers to one turn taken by one of the players.

Important issues of identified themes from results of reading.

Evolutionary approach to the game of checkers *M. Kusiak, K. Waledzik, J. Mandziuk, Warsaw U. of Technology* [2]

This paper mentions 25 heuristics that that can be used together to create a linear combination. We could potentially use these heuristics in unison to create the evaluation function.

These heuristics can be grouped depending on the state of the game. For instance, some heuristics would be more useful during the early-mid game, whereas other heuristics would be only useful for the end game. The paper suggests that this performs the strongest out of their options. The paper goes into further detail, suggesting that a 3-phase heuristic performs "unsurprisingly" the best out of the tested heuristics.

The population will consist of coefficients of these heuristics. Selection is managed by tournaments. The winners of two tournaments are coupled and crossbred. Crossover is managed by a random partition of its coefficients, with random assignments from each parent. The offspring would replace the weakest specimen in the population. Mutation occurs here are simple math operations on the coefficients.

The Fitness Function becomes a little intricate here, and can only be best described using the original source:

> The game was partitioned into several disjoint stages according to the number of moves already performed. During the first phase of the algorithm a heuristic that would be able to assess correctly situations close to the end of the game was to be obtained. In order to achieve this, alpha-beta algorithm with no heuristic was used to assess a number of randomly generated positions close to the leaves of the game tree. All positions beyond the depth of alpha-beta algorithm were considered a draw.
> Subsequently, each specimen assessed the same positions and its fitness was calculated according to a formulae $n/\sum(h_i - a_i)^2$, where n denotes the number of test situations, h_i is the assessment of the i -th test situation by the heuristic specimen and a_i- by the alpha-beta algorithm. Once the initial stage had ended, worst fitted fraction of the population was replaced by new random specimens. New board situations closer to the root of the game tree were generated and they were assessed by the alpha-beta algorithm with the fittest specimen of the previous phase used as its heuristic function.
> The process continued until the root of the game tree was reached. In other words, the evaluation function of HG was evolved step-by-step starting from positions achieved far from the initial position and

> moving backwards. In each phase a constant fraction of all the test boards came from the stage of the game closest to the beginning (i.e. the stage considered most recently). In the first step positions obtained in between 82 and 86 moves were considered.

Apply genetic algorithm to the learning phase of a neural network *S. Perez, UC Irvine* [3]

This paper discusses comparisons between the possible approaches that would improve the quality of weights on the neural network: Back Propagation and Genetic Algorithms. The comparisons would be measured using a Balance Scale.

The test data used was split into two sets. One set is used to feed into the neural network as training data, with its performance tested. The set is then used as test data for comparison. Hidden layers are necessary for the comparison in order to make the measurements fair.

The paper suggests that the GA approach proposes better results "as expected":

> As expected, the GA approach gives better results than the back-propagation method with almost all the iterations used (using 6% and 50% as mutation and crossover probabilities). Only when the number of iterations is very small, the back-propagation method gives better precision performance.

The paper suggests possible downfalls to a genetic algorithm approach, where its search space is dramatically larger in the event that the margin of the weights aren't known. This is not relevant as we should know what they are.

> What would have happened if the margin of the weights was not known? The search space for the genetic algorithm would be much larger that the one that we were supposed to search, and the solution not only will converge slowly, but also with less precision.

Evolving Neural Networks to Play Checkers without Relying on Expert Knowledge *Kumar Chellapilla, UC San Diego; David B. Fogel, Natural Selection, Inc.* [4]

Fogel's paper describes an implementation of an evolutionary NN that plays checkers. This may propose a problem wherein the project would need to be sufficently distant from the methods described in the paper.

The paper describes a perspective on an approach that can be used to solve the problem described. It consists of a min-max algorithm with alpha-beta pruning. The algorithm determines the quality of its potential moves, with it choosing the best move out of it.

Therefore, its evaulation function consists of a neural network: A possible board move is the input, and the output is a score value of how good the board is.

It uses an evolutionary algorithm (which is assumed to be a genetic algorithm), with the neural network as the population. The individual weights of the neural

networks were measured against each other using a tournament, where each neural network plays against each other in a simulated game of checkers. The best neural networks stay on.

They used 250 generations, of 150 games per generation, with a population of 15 neural networks. The bot was then used to play against players on the gaming site `zone.net`, where it achieved an ELO of 1914.

It mentions that some pitfalls of the bot were dependent on the ply, where calculating a large ply would be computationally infeasible, even though it would be necessary for some end-game sequences. It also suggests that a min-max strategy may not suggest the best move (although it is the accepted standard.)

Proposed Direction of the Project

The approach taken would follow the a similar framework as described in *Chellapilla* and *Fogel's* paper.

The neural networks would take as input a board, and outputs a value that determines the effectiveness of that board. The number of hidden layers are dependent on the heuristics, proposed by *M. Kusiak* and *K. Waledzik,*. This is left for experimentation during the implementation of the project.

The genetic algorithm would be used to improve the quality of the weights of the neural networks. We will need a population of neural networks (where they have varying weights), with a round-robin style tournament as the evaluation function. Finalists are chosen as the basis for the next generation.

Experimentation of techniques would be conducted in these areas, and I could imagine that they would be tested independently of each other.

- Effectiveness of the min-max algorithm
- Genetic Crossover Algorithm
- Genetic Mutation Algorithm
- Effectiveness of the Neural Network
- When/where to perform the genetic algorithm

An interface is also needed in order to test the AI. There are multiple options for this; It could be built for the web, with a `node.js` backend, which would interact with the AI, or we can simply use a program built locally for it.

In order to test it's performance, we could, like *Chellapilla* and *Fogel's* paper, make an account for the AI at a popular gaming site and act as the intermediary for it. This would provide a lot of benefits, such as a relative ELO amongst other players on the site.

References

[1] "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989

[2] "Evolutionary approach to the game of checkers" by M. Kusiak, K. Waledzik, and J. Mandziuk (Warsaw U. of Technology), ICANNGA, 2007

[3] "Apply genetic algorithm to the learning phase of a neural network" by S. Perez (UC Irvine), 2008

[4] "Evolving Neural Networks to Play Checkers without Relying on Expert Knowledge" by Kumar Chellapilla (UC San Diego), David B. Fogel (Natural Selection, Inc), IEEE Transactions on Neural Networks, Nov. 1999