Reinforcement Learning, Looking for New Backgammon Strategies

Student Name: Fatema Alkhanaizi

Supervisor Name: Rob Powell

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

Abstract —

A Context/Background

TD-Gammon of Tesauro (**?**, **?**) had demonstrated the impressive ability of machine learning techniques to learn to play games. TD-Gammon used reinforcement learning techniques with a Neural Network (NN) that trains itself to be an evaluation function for the game of backgammon, by playing against itself and learning from the outcome (**?**). However, the monolithic nueral network soon reached its limitation an outcome studied by Boyan () and a modular nueral network becomes more suitable to over come this limitation. The newest software for Backgammon build on top of the modular architecture such as eXtreme Gammon (**?**) and GNUBG (**?**).

B Aims

The aim of this project is to study the influence of including a hybrid of known backgammon strategies such as Priming and Back games as part of the nueral network architecture and to find a combination of strategies to maximize the performance.

C Method

Initially the nueral network from Tesauro's TD Gammon will be implemented and trained. This network will be referred to as monolithic nueral network. Then, multiple Modular Nueral Networks that include hybrid of backgammon strategies will be implemented and trained.

D Proposed Solution

Keywords — Backgammon; Reinforcement Learning; Modular Nueral Network;

I INTRODUCTION

This section briefly introduces the project, the research question you are addressing. Do not change the font sizes or line spacing in order to put in more text.

Note that the whole report, including the references, should not be longer than 12 pages in length (there is no penalty for short papers if the required content is included). There should be at least 5 referenced papers.

A Backgammon Game

Game rules followed and game set up.

B TD Gammon

first implementation, limitations

C Searching Algorithm

depth of lookup to choose the best action for the current turn (1-ply, 2-ply ... etc)

D Learning Method

E Nueral Network architecture

F Research Questions

II DESIGN

This section presents the proposed solutions of the problems in detail. The design details should all be placed in this section. You may create a number of subsections, each focusing on one issue.

A Requirements

B Algorithms

B.1 Reinforcement Learning

- Define Reinforcement Learning components in term of Backgammon
  - Temporal difference learning
  - value function with nueral network (backprobogation)
  - after state value function

B.2 Depth search algorithm

C Nueral Network Architecture

For the Monolithic Nueral Network, it will be based on Tesauro's TD Gammon implementation. A fully-connected feed-forward nueral networks with a single hidden layer. The architecture consists of one input layer with 296 inputs units which consists of raw inputs representing the checkers configuration on the board and expert features mentioned in table-3).

The modular nueral networks that will be implemented for this project consist of a combination of the following networks:

1. One network for racing game; the checkers cannot be hit anymore by another checker or the checkers layout is close to this outcome

2. One network for back/holding game positions; the player has some checkers on the bar or two or more checkers in 18, 19, 20 or 21 point
3. One network for priming game; if the player has a prime of 4-5 fields (a long wall of checkers)
4. One default network for all other positions

Initially all networks will have the same layout as the monolithic nueral network, however the networks don't necessarly need to have the same layout. Different layouts will be tested to configure each nueral network e.g. racing game network does not need inputs for all fields since most checkers will be beared off or close to being beared off.

C.1 Designer Domain Decomposition Architecture

C.2 Meta-Pi Networks

- This network will replace the gating program. It will be used to determine the best nueral network to be triggered based on a give input. The benefit of this approach is that it could discover that some conditions in the gating program that used to trigger one nueral network might be more fitting for another network to handle such input; reduce the stiffness introduced by hard coding the triggers for the sub-networks. This will allow the agent to develop a more flexible strategy and eventually better decisions.

D System Components

Python 3.6 will used as the language for this project. The nueral networks will be implemented using tensorflow package with GPU support. The figure below shows the expected structure of the project.

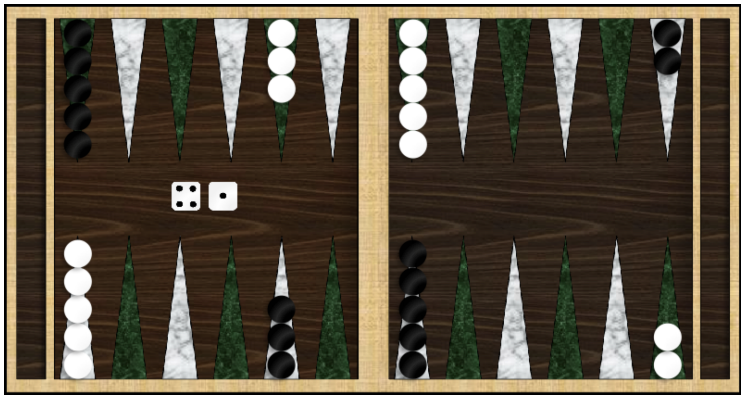

Figure 1: Backgammon board setup

D.1 Game

The user interface for the game is not the focus of this project, so a pre-existing interface written by ... will used. The implementation will be refactored so it can be used in the project. This module will hold the game setup and define the rules and constraints of the game e.g. take an action, find legal moves, game board setup ... etc.

D.2 Agents

There are 3 types of agents will be implemented for this project, all agents will implement get_action method:

- **A human agent**, an interactive agent which takes user inputs either from the command line or by capturing the user clicks though a GUI.
- **A random agent** picks a random move from the list of legal moves based on the dice role. This agent will be mainly used for testing.
- **AI agent** uses a modular nueral network to determine the action for the current turn. A list of legal moves is obtained from the game module and the best action is picked after running the outcome of each move through the network. The search algorithm implemented is greedy and at a single depth, 1-ply; the action with the maximum output is picked.

D.3 Modnet

This module defines the operations for extracting features from the game board, testing, training, reading and writing three kinds of modular nueral networks. The architecture of those networks is explained in section-C. This module heavily depends on Subnet module. A game-specific gating program is implemented in this class which determines which subnetwork applies to a given set of extracted features.

D.4 Subnet

This module includes the Nueral Network implementation using tensorflow. It provides routines for storing and accessing model, checkpoints and summaries generated by tensorflow.

E Training

F Monolothic Nueral Network

This network will be trained on 1 million games.

G Modular Nueral Network

H Meta-Pi Network

I Testing and Evaluation

As the networks are training, in every 5000 game, a test will be ran against the random player to check the current performance of the networks architecture implemented.

J Extensions

K References

The list of cited references should appear at the end of the report, ordered alphabetically by the surnames of the first authors. The default style for references cited in the main text is the Harvard (author, date) format. When citing a section in a book, please give the relevant page numbers, as in (Budgen 2003, p293). When citing, where there are either one or two authors, use the names, but if there are more than two, give the first one and use "et al." as in , except where this would be ambiguous, in which case use all author names.

You need to give all authors' names in each reference. Do not use "et al." unless there are more than five authors. Papers that have not been published should be cited as "unpublished" (Euther 2006). Papers that have been submitted or accepted for publication should be cited as "submitted for publication" as in (Futher 2006) . You can also cite using just the year when the author's name appears in the text, as in "but according to Futher (2006), we ...". Where an authors has more than one publication in a year, add 'a', 'b' etc. after the year.

References

Budgen, D. (2003), *Software Design*, 2nd edn, Addison Wesley.

Euther, K. (2006), Title of paper. unpublished.

Futher, R. (2006), Title of paper 2. submitted for publication.

Table 1: List of Functional Requirements

ID	Requirement	Priority
FR1	A module for Backgammon game must be implemented. This will include the actual board setup with the rules and constraints of the game e.g. legal moves and the dice role	High
FR2	An AI agent should be created such that it uses a nueral network to evaulate legal moves/actions to play backgammon and a greedy search algorithm to pick the best legal move/action	High
FR3	A module for the nueral network should be created. It should support the funtionalities required for the nueral network e.g. updating weights through back-propagation, saving and restoring the network meta-data	High
FR3	A module for the training and testing the nueral network should be created	High
FR3	Monolithic Nueral Network should be implemented and trained based on Tesauro's TD Gammon	High
FR4	Modular Nueral Network that includes Racing Game strategy should be implemented and trained	High
FR5	Modular Nueral Network that includes Racing and Holding Game strategies should be implemented and trained	High
FR6	Modular Nueral Network that includes Racing and Priming Game strategies should be implemented and trained	High
FR7	Modular Nueral Network that includes Racing, Holding and Priming Game strategies should be implemented and trained	High
FR8	depth (n-ply) search algorithm should be implemented and incoporated into AI agent in FR2	Medium
FR9	Textual User interface for a Human agent to play against the AI agent from FR2 should be implemented	Medium
FR10	Graphical User interface for a Human agent to play against the AI agent from FR2 should be implemented	Low
FR10	Create a hueritic function for deciding when to include the doubling cube	Low

Table 2: List of Non-functional Requirements

ID	Requirement	Priority
NFR1	Trained networks should return the outcome from forward propagation within 40ms	Medium
NFR2	The AI agent should pick a legal move/action within 1s. In other word the search algorithm for the best move/action should return a value within 1s	Medium

Table 3: Expert features for each player

Feature name	Description
x_token	1 if the player is playing with x token perspective, 0 otherwise
o_token	1 if the player is playing with o token perspective, 0 otherwise
bar_pieces	number of checkers held on the bar for a player
pip_count	pip count for a player
off_pieces	percentage of pieces that a player has beared off
hit_pieces	percentage of pieces that are at risk of being hit (single checker in a position which the opponent can hit) for a player