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* —** These instructions give you guidelines for preparing the design paper. DO NOT change any settings, such as margins and font sizes. Just use this as a template and modify the contents into your design paper. Do not cite references in the abstract.

The abstract must be a Structured Abstract with the headings **Context/Background**, **Aims**, **Method**, and **Proposed Solution**. This section should not be no longer than a page, and having no more than two or three sentences under each heading is advised.

A Context/Background

There are many strategies that a professional backgammon player follows to better his chances of winning the game. In 1992, limitations of Monolithic Nueral Network - a generic strategy, one-size fit all Solution .

B Aims

study the effect of including hybrid of Backgammon strategies to the learning network.

C Method

the introduction of those strategies will be part of the NN architecture.

D Proposed Solution

***Keywords* —** Put a few keywords here.

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 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.

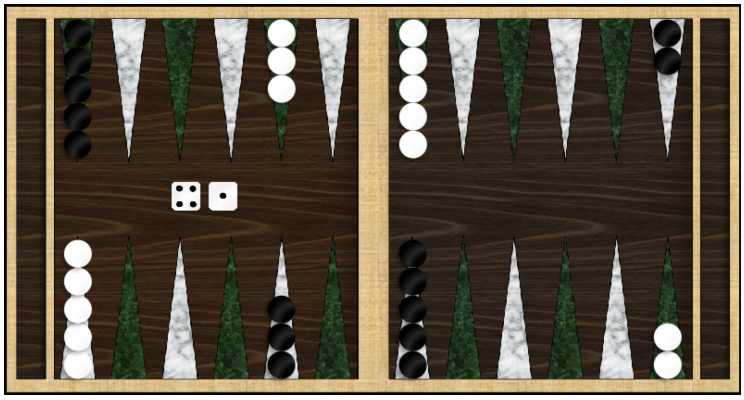

Figure 1: Backgammon board setup

B.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 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.

B.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.

B.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.

B.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.

C Nueral Network Architecture

The first implementation for this network will be based on TD-Gammon of Tesauro. For the function evaluation, eligibility traces are going to be included as part of the gradient computations. The network compromises of 3 layers: an input layer, a hidden layer and an output layer. For the basic implementation of Tesauro's TD-Gammon, the following expert features were included to the input layer:

Table 1: Expert features could to the raw inputs for each player

Feature name	Description
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

C.1 Designer Domain Decomposition Architecture

- strategies - racing game, - list of conditions to switch to each network aka gating program - learning the strategy is not done explicitly but it is left to the agent to figure out the strategy based on the current board configuration. This is handled by the gating program. It triggers a certain nueral network based on the extracted features from the game board)

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 Training

A Monolithic Nueral Network will be used as a bench reference for all other networks that will trained. This network will be trained on 1 million games. The implementation will be based on TD-Gammon of Tesauro.

E 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.

F Figures and Tables

In general, figures and tables should not appear before they are cited. Place figure captions below the figures; place table titles above the tables. If your figure has two parts, for example, include the labels "(a)" and "(b)" as part of the artwork. Please verify that figures and tables you mention in the text actually exist. make sure that all tables and figures are numbered as shown in Table **??** and Figure 1.

G 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.