Building A Backgammon AI

Virgile Audi
vaudi@g.harvard.edu

CS182 - Fall 2016

Introduction

Backgammon is one of the oldest board games known to men. It is a 2 player zero-sum game, where playing pieces are moved according to the roll of 2 dice, and a player wins by removing all of his/her pieces from the board before his/her opponent. Players can stop their opponent's progression by capturing left alone pawns.

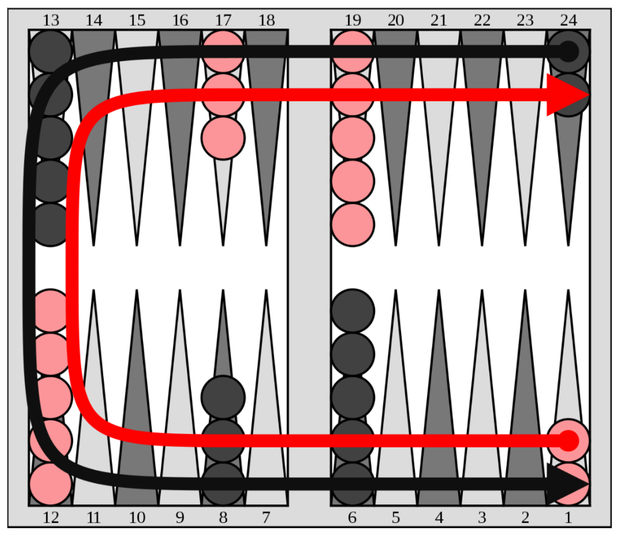

Figure 1. Board and Rule of Backgammon

The goal of this project was inspired by the approach taken to create the AlphaGo AI i.e. **train an AI by using both expert learning topped with Reinforcement Learning.**

Problem Formulation

For the scope of this project, we will use the framework of Markov Decision Process:

- States $s \in \mathcal{S}$: All possible distributions of pawns combined with a roll of two dice
- Set of actions $a \in \mathcal{A}$ given by the possible moves of pawns given the board and dice

The objective is to choose the optimal move at each step. We can evaluate the number of boards in backgammon to 18,528,584,051,601,162,496, which would make evaluating a value for every state completely infeasible. We therefore resorted to approximations by representing the board as a feature vector such as:

- Number of left alone pawns in each quadrant
- Number of enemy pawns imprisoned in each quadrant
- Number of safe pawns

To chose the optimal move is therefore equivalent to finding weights for each feature so that the best move leads to board with highest score **sc = $\underline{\mathbf{w}}\cdot\underline{\mathbf{f}}$**

Backgammon is an easy game to play, but not easy to play well.
Can an AI learn to play it well?

Models and Algorithm

A. Expert Learning using Multiclass Logistic Regression

Trained using a dataset of more than 30000 moves observed during backgammon championships. Only moves used by the winner of a given game were used. Classes correspond to the move to take i.e.

$$\max_{a \in \mathcal{A}} \frac{\exp(\mathbf{w}\cdot\mathbf{f}(a))}{\sum_{a \in \mathcal{A}} \exp(\mathbf{w}\cdot\boldsymbol{f}(a))}$$

Trained using Stochastic Gradient Descent and a cross entropy loss with regularization:

$$w_i \leftarrow w_i - \eta \frac{\partial L(y, \hat{y})}{\partial w_i}$$

B. Genetic Algorithms and Tournaments

The goal was then to use Reinforcement Learning to improve on the weights. To do so, I used a genetic algorithm with the following pseudo-code:

```
Step 0: Initialize the weights using
the results of the logistic regression

While it < max_it:
  1.  Generate a contestant by
      perturbing slightly each
      weight: ŵ_i = w_i + U(-.5,.5)
  2.  Play a tournament i.e. first
      to 15 points
  3.  If current champion wins:
          Continue
      Else:
          Update the weights:
          w_i = (1-a)w_i + aŵ_i
          where a depends on the
          amplitude of the win
```

C. Approximate Q-Learning

The last approach taken was to use approximate Q-learning to update the weights:

$$w_i \leftarrow w_i + \alpha\big(r + \gamma \max Q(s', a') - Q(s, a)\big) f_i(s, a)$$

Here s' represent the board and dice after the move made with a, an opponent's move given a roll of two dice and new roll of dice.

Results

In order to asses the results, two methods were used:

1. Evaluating the accuracy of predictions on held-out test data of "pro" moves

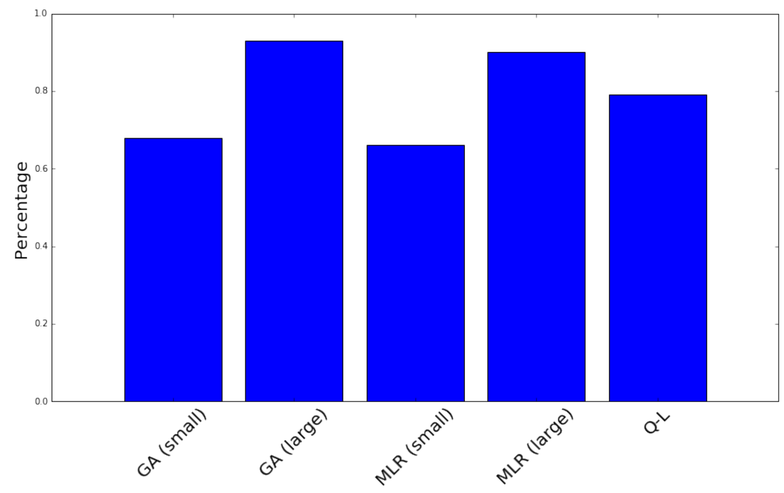

Figure 2. Accuracy on held-out test data

2. Playing tournaments opposing AI's at different levels of training and/or between different methods of training: logistic vs genetic for instance

	GA (L)	MLR (S)	MLR (L)	Q-L
GA(S)	(8-15)	(15-5)	(15-6)	(16-1)
GA(L)		(15-0)	(16-0)	(15-0)
MLR (S)			(1-15)	(8-15)
MLR (L)				(15-2)

Table 3. First-to-15 Results
(S=Small features and L=Large features)

Bibliography

- *Mastering the game of Go with deep neural networks and tree search*, Deep- Mind in Nature January 2016
- *Temporal Difference Learning and TD-Gammon*, Gerald Tesauro, 1995
- *Why Did TD-Gammon Work*, J. B. Pollack and A. D. Blair, 1996
- *Self-Play and Using an Expert to Learn to Play Backgammon with Temporal Difference Learning*, Marco A. Wiering 2010