Directed Study – A2
Jéssica Pauli de C. Bonson (B00617515)

Survey the Tesauro's approach to Backgammon

Backgammon is a two-player game where the players start with their checkers in pre-defined positions at the board, and the goal is to move all of them off the end of board first, winning the game. The movement is realized by rolling dices and moving the checkers in opposite directions along the track as allowed by the dice roll.

While the checkers move along the board it's possible to hit or block the opponent's checkers. A checker is hit when it`s alone in a position and a opponent checker lands on it, a hitted checker it's moved to the begining of the board and must re-enter the board before other checkers can be moved. Blocking configurations are formed when there are two or more checkers in the same position, prohibiting the opponent to move her checkers to that position.

Additionaly, if a player wins while the opponent hasn't taken any checkers off the board, she receives double the normal points for winning (a "gammon"). It's also possible to happen a more rare situation, where the player wins before the opponent has taken checkers off and has checkers in the farmost quadrant, called "backgammon".

Another complexity is added by the use of a "doubling cube", which is used to double stakes of the game. When a player offers to double the stakes, the opponent can either accept it or resign the match. If the player accepts, she gets the exclusive right to make the next double.

So, the total points of winning a match is given by the current value of the doubling cube multiplied by 1 for a regular win, 2 for a gammon, and 3 for a backgammon. All those components in the game leads to number of possible complex strategies at the expert level of play.

The backgammon game has some interesting dynamics that influence the machine learning. [1] points some features useful regarding self-play and random initial conditions: 1) The game always terminate, because it's impossible to draw, so even a network playing randomly would eventually end the match; 2) The nondeterminism generated by the dices rolls leads to a greater exploration of the search space than in a deterministic game, the exploration also benefits from the fact that every position can be achieved by some sequence of moves of any other position.

[2] highlights that in backgammon one dice roll or an improbable sequence are able to greatly reverse the chances of victory of a player, so in almost every situation, there is a nontrivial chance of winning and a nontrivial chance of losing, the authors also argue that it facilitates the learning process, since it provides potential to learn from the consequences of the current move and it, along with the nondeterminism, avoids the creation of simple strategies with suboptimal moves.

In [3] Tesauro states that due to the high branching ratio resulting from the probabilistic dice rolls, hundreds branches per ply, deep searches methodologies are not feasible. The author also criticizes the option of using heuristics and features developed by human experts to work around the problem of the branching ratio, since it leads to problems such as: there may be a large number of features required, and it`s hard to code all of them; the features may interact with each other in complex and unanticipated ways; and some of the expertise being emulated may contain errors. The author prefers the approach where the learning program is able to has its own sense of positional judgement by learning from experience in playing against itself, as occurs in TD-Gammon, this way the program may have less complex strategies, but it will avoid situations where the learner may not know what to do.

Another point cited by Tesauro in [3]:

"In backgammon the game-theoretic optimal value function is a real-valued function with a great deal of smoothness and continuity, in the sense that a small change in position leads to a small change in expected outcome. Such a function is presumably easier to learn than the discrete (win, lose, draw) value functions of deterministic games, which contain numerous discontinuities where a small change in position can make a huge difference in its game-theoretic value."

The TD-Gammon, developed by Tesauro in [1] and improved in [3], is a neural network that is able to learn to play backgammon solely by playing against itself and learning from the results. The main component of TD-Gammon is a neural network that utilizes a standard multilayer perceptron architecture, that has been shown to be able to approximate any nonlinear function to arbitrary accuracy [3]. The training is realized using the TD learning methodology, its main idea is to base learning on the difference between temporally successive predictions, leading the learner's current prediction for the current input pattern to be a more closely match to the subsequent prediction at the next time step.

The learning procedure works as follows [3]: The network observes a sequence of board positions (input vectors x1, x2, ..., xf), during the game resulting in a final reward signal z at the end of the match. Each time step in the sequence corresponds to a move made by one side (a “ply”). For each input pattern xt there is a neural network output vector Yt indicating the network’s estimate of expected outcome for pattern xt. Yt has four components, representing the four possible results of the match: White winning, Black winning, White gammoning, and Black gammoning. There is also possible to win due to backgammon, but since it's a very rare situation the author prefered to don't consider it.

At each time step (ply) the TD(λ) algorithm is applied to change the network’s weights according to the formula below, which is basically a junction of a temporal difference method for temporal credit assignment with a gradient-descent method for structural credit assigment.

$$w_{t+1} - w_t = \alpha(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

In the equation α is the learning rate, w is the vector of weights that parameterizes the network, ∇wYk is the gradient of network output with respect to weights, and λ is a heuristic parameter that controls the temporal credit assignment. When λ = 0, no feedback occurs beyond the current time step, while when λ = 1, the error feeds back without decay arbitrarily far in time. Intermediate values of λ provide a smooth way to interpolate between these two limiting cases.

Also, at each time step the neural network itself is used to select moves for both White and Black sides. The neural network scores every possible legal move, and then selects the one with maximum expected outcome for the side making the move. This self-play training paradigm is used even at the start of learning, when the network’s weights are random, and hence its initial strategy is a random strategy.

At the end of each game, a final reward signal z is given, based on the resulted outcome. Once again the formula is used to change the weights, but now the difference (z − Yf) is used instead of (Yt+1 − Yt). The proposed backgammon learning system uses no intermediate goals.

In [1] Tesauro performs three initial experiments of increasingly complexity applying TD learning to backgammon. For all experiments the parameter settings were α = 0.1 and λ = 0.7.

The first one uses TD to learn a simple case of disengaged bearoff positions, where both sides have their checkers in their home quadrant and are able to remove them from the board. The heuristic here is to always select the move that takes the maximum number of pieces off the board. The codification in the network encoded the number of Black checkers at locations 1-6, number of White checkers at locations 19-24, number of White and Black checkers off the board, and it also encoded the player to move. The initial board configurations were a uniform random distribution of all 15 checkers for each side in the home board positions. The reward signal was just a "win" or "lose". The results in the experiments showed that during training the absolute prediction error always decreased, but the move-selection performance reached a maximum and then decreased thereafter. The network was able to achieve around 92% of correct moves.

The second experiment developed in [1] considered the more complex training of general racing situations covering the entire board, also, in this case "gammons" could occur. Codification represented the number of White and Black checkers at locations 1-24, number of checkers off and player to move. The initial board configuration was an uniformly random distribution of the White and Black checkers according to a randomly picked divider location on the board. The reward signal had four components, representing the possible outcomes: W wins, W gammons, B wins, B gammons (p1, p2, p3, p4). The network estimated probabilities were used to estimate expected payoff, and then it's selected the move that maximizes the expected payoff. For example, the expected payoff to White is given by p1 + 2p2 − p3 − 2p4. The performance measure used was the fraction of time the network agreed with a human expert on the best move, the network was able to achieve around 82% of correct moves.

The third experiment extends the TD learning to play the full game, starting from the default initial positions in backgammon, and being able to hit and block. This case is much more complex than the other ones, since the optimal strategy for one side depends more critically on what strategy the opponent is using. The codification encoded for each one of the 24 locations, if there was one, two, three or beyond three White or Black checkers in that position. It was also encoded the number of checkers on the bar, off the board, and the player to move. The reward signal was the same one used in the previous experiment. The network trained in this approach was evaluated by comparing results against Gammontool, and this way it was demonstrated that the TD network had a better performance than EP network, a network that used features with imbued human expert knowledge. While the EP networks emphasized blocking, the TD networks developed a style emphasizing running and tactical play, emphasizing racing rather than blocking.

In the preliminary experiments cited, the network codification just used raw data, a knowledge-free approach with no lookahead, in [3] Tesauro presents three more complex versions of TD-Gammon.

The TD-Gammon version 1.0 used a set of hand-crafted features, such as the strength of a blockade and the probability of being hit, and a simple 1-ply search, in which every top-level move is scored by the neural net, and the highest-scoring move is selected. This network was trained for 300,000 self-play games and reached the level of a competent advanced player which was clearly better than any other previous backgammon program [3].

The version 2.1 made use of the increased computer power available to train for longer sessions (1.5 million games), and to use a greater depth search, a 2-ply search. The 2-ply search algorithm works as follows [3]:

"First, an initial 1-ply analysis is performed and unpromising candidates are pruned based on the 1-ply score. (This is commonly known as forward pruning.) Then, the remaining toplevel candidates are expanded by an additional ply. The 1-ply expansion of the surviving candidates involves making a 1-ply move decision for each of the opponent's 21 possible dice rolls, and computing a probability-weighted average score (weighting non-doubles twice as much as doubles) for each of the resulting states."

This network reached the level of a high-level expert, being competitive with the world's best human players.

The versions 3.0 and 3.1 of the algorithm are improved by being able to do a simplified 3-ply search, which is similar to the 2-ply search, except that a depth-2 expansion of the top level moves is realized, rather than a depth-1 expansion. This network has a clear advantage over top humans in piece movement decisions, and a slight advantage in cube decisions.

Both versions 2.1 and 3.1 are able to do double and take/pass decisions. The doubling algorithm isn't included in the self-play, but can be obtained by feeding the neural network's estimates into a heuristic doubling formula, as explained in [3].

Finally, [2] realizes a study that proposes that the great performance of the preliminary TD-Gammon in backgammon [1] isn't just due to the TD learning, but also to properties of the game, such as nondeterminism and others described in the begining of this survey. To make a point about it, in [2] is developed a feed-forward neural network which learning is performed through hill-climbing, a simple algorithm which consists of doing the champion network play against a slighly mutated version of itself, and changing the weights if the challenger wins. The learning starts with an initial champion of all zero weights, and the moves are performed by rolling the dice, generating all legal moves, converting them into the proper network input, and picking the position with the highest evaluation. The best network evolved could win about 45% of the time against PUBEVAL, which the author believes to have a similar level of Tesauro's 1992 network. The author concludes by stating that, due the results in [2], the reinforcement and temporal difference methods are not the primary cause for success in TD-Gammon, rather it is the dynamics of backgammon combined with the power of co-evolutionary learning.

References:

[1] G. Tesauro, Practical issues in temporal difference learning, Machine Learning 8 (1992).
[2] J.B. Pollack, A.D. Blair, Co-evolution in the successful learning of backgammon strategy, Machine Learning 32 (1998).
[3] G. Tesauro, Programming backgammon using self-teaching neural nets (2002).