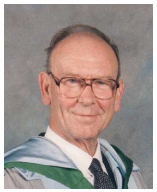
COMP3411: Artificial Intelligence

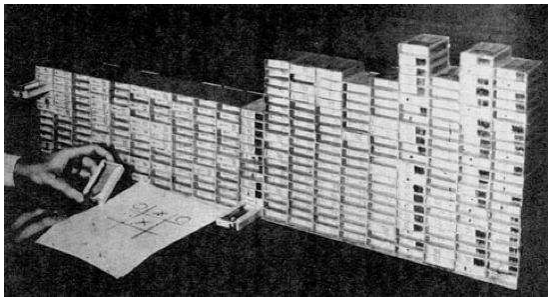
15. Learning Games

Outline

- 1959 Checkers (Arthur Samuel)
- 1961 MENACE tic-tac-toe (Donald Michie)
- 1989 TD-Gammon (Gerald Tesauro)
- 1997 TD-leaf (Baxter et al.)
- 2009 TreeStrap (Veness et al.)

MENACE

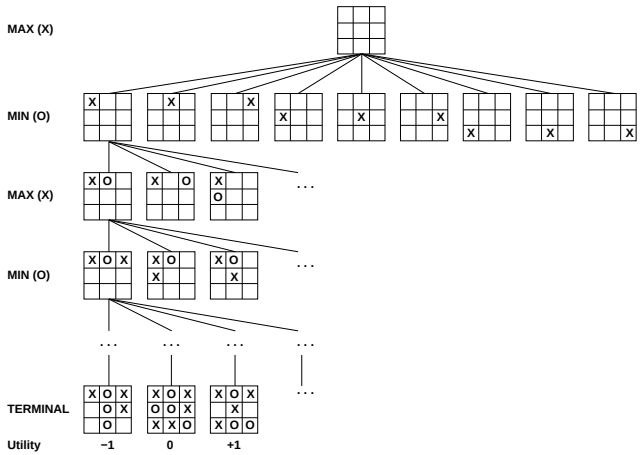




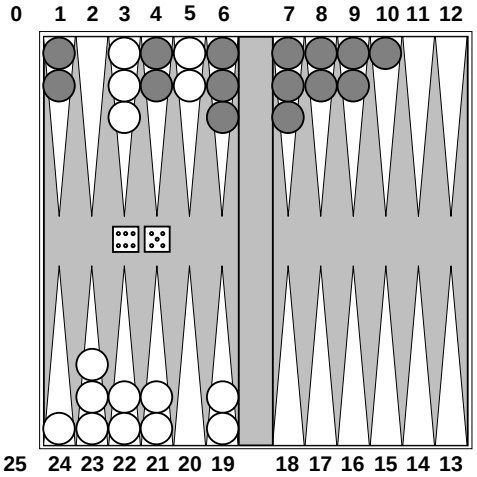
Machine Educable Noughts And Crosses Engine
Donald Michie, 1961

MENACE

Game Tree (2-player, deterministic)



Backgammon



Backgammon Neural Network

Two layer neural network

- 196 input units
- 20 hidden units
- 1 output unit

Board encoding

- 4 units $\times$ 2 players $\times$ 24 points
- 2 units for the bar
- 2 units for off the board

The input s is the encoded board position (state),
the output V(s) is the value of this position (probability of winning).

Chess

Heuristic Evaluation for Chess

- Material weights
  - Queen = 9, Rook = 5, Knight = Bishop = 3, Pawn = 1
- Piece-Square weights
  - some (fractional) score for a particular piece on a particular square
- Attack/Defend weights
  - some (fractional) score for one piece attacking or defending another piece, etc.
- Other features, such as Pawn structure, Mobility, etc.
- There are no hidden nodes. V(s) is a linear combination of input features, composed with a sigmoid function, to produce a value between 0 and 1 (probability of winning).

Game Play

- Backgammon: At each move, roll the dice, find all possible "next board positions", convert them to the appropriate input format, feed them to the network, and choose the one which produces the largest output.
- Chess: Alpha-Beta search, using the value function V(s) to evaluate the leaf nodes.

Backpropagation

$$w \leftarrow w + \eta(T - V)\frac{\partial V}{\partial w}$$

V = actual output
T = target value
w = weight
η = learning rate

Q: How do we choose the target value T ?

In other words, how do we know what the value of the current position "should have been"? or, how do we find a better estimate for the value of the current position?

How to Choose the Target Value

- Supervised Learning
  - learn moves from human games (Expert Prefecences)
- Temporal Difference Learning
  - general method, does not rely on knowing the "world model" (rules of the game)
- methods which combine learning with alpha-beta search (must know the "world model")
  - TD-Root (Samuel, 1959)
  - TD-Leaf (Baxter et al., 1998)
  - TreeStrap (Veness et al., 2009)

Temporal Difference Learning

We have a sequences of positions in the game, each with its own (estimated) value:

(current estimate) $V_k \rightarrow V_{k+1} \rightarrow \ldots \rightarrow V_m \rightarrow V_{m+1}$ (final result)

TD(0): Use the value of the next state (V_{k+1}) as the training value for the current state (V_k).

TD(λ): use T_k as the training value for V_k, where

$$T_k = (1-\lambda) \sum_{t=k+1}^{m} \lambda^{t-1-k} V_t + \lambda^{m-k} V_{m+1}$$

T_k is a weighted average of future estimates,
λ = discount factor ($0 \leq \lambda < 1$).

TD-Gammon

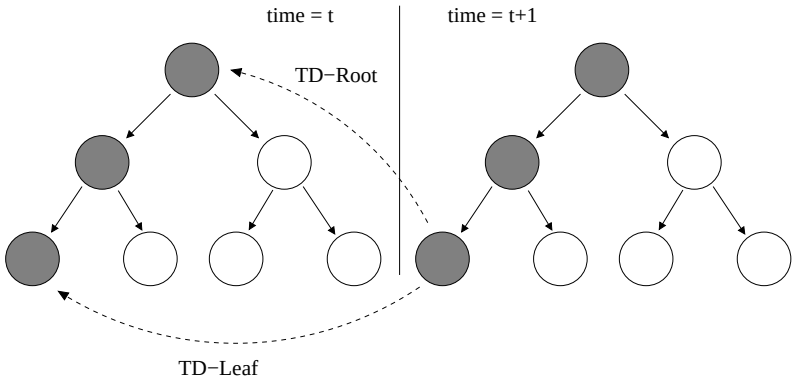
- Tesauro trained two networks:
  - EP-network was trained on Expert Preferences (Supervised)
  - TD-network was trained by self play (TD-Learning)
- TD-network outperformed the EP-network.
- With modifications such as 3-step lookahead (expectimax) and additional hand-crafted input features, TD-Gammon became the best Backgammon player in the world (Tesauro, 1995).

Discussion

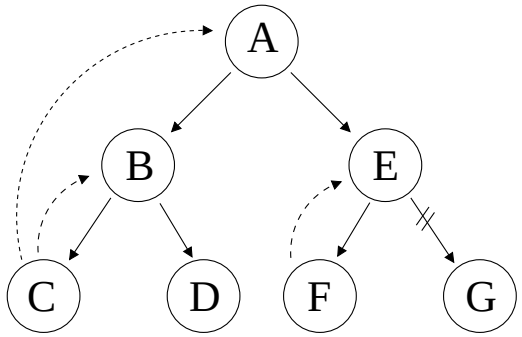
- Random dice rolls in Backgammon force self-play to explore a much larger part of the search space than it otherwise would.
- Humans are bad at probabilistic reasoning?
- Evolutionary Computation can also produce a surprisingly strong player, but a gradient-based method such as TD-learning is better able to fine-tune the rarely used weights, and exploit the limited nonlinear capabilites of the neural network.
- For deterministic games like Chess, direct TD-Learning performs poorly. Methods which combine search and learning are more effective.

Checkers Program (Arthur Samuel)

TD-Root and TD-Leaf



TreeStrap



Node E is **not** on the line of best play, but it is still updated, towards Node F. If Node G is pruned, E can still be updated, provided its current estimate is on the wrong side of the upper/lower bound.

Summary

- Games can be learned from human expert preferences, or from self-play (or a combination)
- TD-Learning is a general method, which does not rely on knowing the world model
- TreeStrap is more powerful, because it also refines the value of moves which were not chosen; but it relies on knowing the world model