Reinforcement Learning
– TD Gammon

slides adapted from:
http://www.cmpe.boun.edu.tr/~ethem/i2ml
And Bryan Pardo (Northwestern University)
Basic RL Model

1. Observe state, $s_t$
2. Decide on an action, $a_t$
3. Perform action
4. Observe new state, $s_{t+1}$
5. Observe reward, $r_{t+1}$
6. Learn from experience
7. Repeat

Goal: Find a control policy that will maximize the observed rewards over the lifetime of the agent
Learning Value Functions

- Q-learning
- SARSA
- TD learning
TD-Learning Algorithm

1. Initialize $V^\pi(s)$ to 0 $\forall s$
2. Observe state, $s$
3. Perform action according to the policy $\pi(s)$
4. Observe new state, $s'$, and reward, $r$
5. $V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
6. Go to 2

$\gamma$ = future returns
discount factor
$\alpha$ = learning rate
Getting the Policy

$$\pi^*(s) = \arg \max_a \ (R(s, a, s') + V^{\pi}(s'))$$

$$\pi^*(s) = \arg \max_a \ Q(s, a)$$

How do you tell which action to take from each state?
TD Gammon

[Tesauro, 1995]

Learn to play Backgammon

Immediate reward

- +100 if win
- -100 if lose
- 0 for all other states

Trained by playing 1.5 million games against itself
Now approximately equal to best human player
Backgammon

SITUATIONS: configurations of the playing board (about $10^{20}$)

ACTIONS: moves

REWARDS: win: $+1$
lose: $-1$
else: $0$

Gerald Tesauro,
IBM Watson Research Center
Tesauro, 1992–1995
Minimax Algorithm: An Optimal Strategy

Choose the best move based on the resulting states’

MINIMAX-VALUE…

MINIMAX-VALUE(n) =
    if n is a terminal state
        then Utility(n)
    else if MAX’s turn
        the MAXIMUM MINIMAX-VALUE
        of all possible successors to n
    else if MIN’s turn
        the MINIMUM MINIMAX-VALUE
        of all possible successors to n
Take 1 or 2 at each turn
Goal: take the last match

MAX wins = 1.0
MIN wins/ MAX loses = -1.0
Take 1 or 2 at each turn
Goal: take the last match

MAX wins

\[= 1.0\]

MIN wins/
MAX loses

\[= -1.0\]
Take 1 or 2 at each turn
Goal: take the last match

MAX wins

-1.0

MIN wins/
MAX loses

1.0

W

W
Take 1 or 2 at each turn
Goal: take the last match

MAX wins

MIN wins/
MAX loses

\[ W \rightarrow 1.0 \]

\[ \triangle \rightarrow -1.0 \]

\( +1 \)
Take 1 or 2 at each turn
Goal: take the last match

MAX wins = 1.0
MIN wins = -1.0
MAX loses

MIN wins/
MAX loses
Take 1 or 2 at each turn
Goal: take the last match

MAX wins
\[ W = 1.0 \]

MIN wins/
MAX loses
\[ W = -1.0 \]
Take 1 or 2 at each turn
Goal: take the last match

MAX wins

\[ \text{\( \neg W \) = 1.0} \]

MIN wins/
MAX loses

\[ \text{\( \neg W \) = -1.0} \]
Expecti Minimax

- White’s (max’s) turn, After rolling a 5 and a 6
- Possible moves (5-10,5-11), (5-11,19-24),(5-10,10-16) and (5-11,11-16)
- [1,1], [6,6] chance 1/36, all other chance 1/18
Expecti minimax value

\[
\text{EXPECTI-MINIMAX-VALUE}(n) =
\begin{array}{ll}
\text{UTILITY}(n) & \text{If } n \text{ is a terminal} \\
\max_{s \in \text{successors}(n)} \text{MINIMAX-VALUE}(s) & \text{If } n \text{ is a max node} \\
\min_{s \in \text{successors}(n)} \text{MINIMAX-VALUE}(s) & \text{If } n \text{ is a min node} \\
\sum_{s \in \text{successors}(n)} P(s) \cdot \text{EXPECTIMINIMAX}(s) & \text{If } n \text{ is a chance node}
\end{array}
\]
Evaluation Functions

Suppose we have 100 secs, explore $10^4$ nodes/sec
$\rightarrow 10^6$ nodes per move

Standard approach (Shannon, 1950):
- evaluation function = estimated desirability of position

if TERMINAL-TEST(state) then return UTILITY(state)
BECOMES
if CUTOFF-TEST(state, depth) then return EVAL(state)
How humans play games...
An experiment (by deGroot) was performed in which chess positions were shown to novice and expert players...

- experts could reconstruct these perfectly
- novice players did far worse...

Random chess positions (not legal ones) were then shown to the two groups
- experts and novices did just as badly at reconstructing them!
NeuroGammon

Action selection by 2–3 ply search
Modifying the Weights

\[ w_i \leftarrow w_i + \Delta w_i \]

\[ \Delta w_i = \text{LearningRate}(\text{DesiredOutput} - \text{ActualOutput})x_i \]

This is the difference between what we wanted the output to be and what it actually was.

If the desired and actual are equal, then this is 0 and the weight won’t change.
Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do

  1. Input the training example to the network and compute the network outputs
  2. For each output unit $k$
     \[
     \delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)
     \]
  3. For each hidden unit $h$
     \[
     \delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{h,k} \delta_k
     \]
  4. Update each network weight $w_{i,j}$
     \[
     w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}
     \]
     where
     \[
     \Delta w_{i,j} = \eta \delta_j x_{i,j}
     \]
TDGammon

Action selection by 2–3 ply search

TD error

$V_{t+1} - V_t$
Table 1. Results of testing TD-Gammon in play against world-class human opponents. Version 1.0 used 1-ply search for move selection; versions 2.0 and 2.1 used 2-ply search. Version 2.0 had 40 hidden units; versions 1.0 and 2.1 had 80 hidden units.

<table>
<thead>
<tr>
<th>Program</th>
<th>Training Games</th>
<th>Opponents</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDG 1.0</td>
<td>300,000</td>
<td>Robertie, Davis, Magriel</td>
<td>−13 pts/51 games (−0.25 ppg)</td>
</tr>
<tr>
<td>TDG 2.0</td>
<td>800,000</td>
<td>Goulding, Woolsey, Snellings, Russell, Sylvester</td>
<td>−7 pts/38 games (−0.18 ppg)</td>
</tr>
<tr>
<td>TDG 2.1</td>
<td>1,500,000</td>
<td>Robertie</td>
<td>−1 pt/40 games (−0.02 ppg)</td>
</tr>
</tbody>
</table>
Figure 3. A complex situation where TD-Gammon’s positional judgment is apparently superior to traditional expert thinking. White is to play 4-4. The obvious human play is 8-4*, 8-4, 11-7, 11-7. (The asterisk denotes that an opponent checker has been hit.) However, TD-Gammon’s choice is the surprising 8-4*, 8-4, 21-17, 21-17! TD-Gammon’s analysis of the two plays is given in Table 1.

<table>
<thead>
<tr>
<th>Move</th>
<th>Estimate</th>
<th>Rollout</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-4*, 8-4, 11-7, 11-7</td>
<td>+0.184</td>
<td>+0.139</td>
</tr>
<tr>
<td>8-4*, 8-4, 21-17, 21-17</td>
<td>+0.238</td>
<td>+0.221</td>
</tr>
</tbody>
</table>