Some game background

• Backgammon is two player, alternating move, zero-sum game
• We’ll say more about games later in the semester
• For now:
  • This class of problems is very similar to MDPs
  • Assume other player makes best (worst) move for him (us)
  • MDP-like algorithms will converge to minimax optimal strategy

• AI history: Before RL existed as a term in AI, Samuel’s made a checkers player that learned from experience – used ideas similar to what we call RL today
How most (computer) game players work

- Construct a game tree of possible moves
- Most interesting games cannot be searched to the end of the game (unless we are already very close to the end)
- Players construct a partial game tree
- Use evaluation function to estimate result of searching to game end
- Evaluation function ~ value function
- Tune this with RL

Some Backgammon background

- Backgammon has dice, so has randomness
- Large state space: $10^{20}$
- High branching factor: several hundred (much higher than chess)
- Deep search is impractical – can only do very shallow searches
Previous approaches

- Neuro-gammon viewed backgammon as supervised learning
- Trained NN on database of expert games
- Achieved “strong intermediate” level of play

Limitations:
- Experts may not be optimal
- Experts may be contradictory
- Nothing to enforce consistency
- Expert games may see only a fraction of the state space

A note about $\lambda$

- Last time we talked about Monte Carlo evaluation and TD
- What if wanted to interpolate between these in some way?
- TD($\lambda$) is an approach that does updates based upon multiple steps
  - TD(1) = Monte Carlo evaluation
  - TD(0) = standard TD algorithm presented last time
  - TD($\lambda$) – (0<\lambda<1) combines both, with lower values closer to standard TD, and higher values closer to Monte Carlo
- Picking good $\lambda$ = more data efficiency, but doesn’t change the fixed point
- RP:
  - Not always clear how to tweak $\lambda$.
  - Would rather focus on better features/better algorithm than tweak $\lambda$. 
Training in TD-Gammon

- Initial feature representation was a raw encoding of board positions
- NN was simple by today’s standards – 40 hidden nodes
- Main training paradigm was “self play”
- TD-Gammon played both sides
- Achieved “strong intermediate” play with after 200K games (used 1-ply search)
- Parity with neuro-gammon, but neuro-gammon had carefully engineered features (Tesauro is a good backgammon player)

TD-Gammon 2.X

- Added 2-ply search
- Expert features from neuro-gammon
- 1.5M games of self play
- Played at master level!
Building on TD-Gammon

• Quite difficult to replicate this success in other domains
• For other games, NN diverged or just didn’t play well (e.g. chess, go)

• What’s special about backgammon?
  • Tesauro’s expert features
  • Possible to do well with linear, suggesting an “on ramp” for the NN
  • Smoothness introduced by randomness
  • Maybe people aren’t very good at backgammon?