Lecture 13: Reinforcement Learning

AI For Traditional Games
Prof. Nathan Sturtevant
Winter 2011

UNIVERSITY OF DENVER

Learning in practice

- Neural networks and perceptrons give a mechanism for performing learning
  - Still need an approach to performing the learning in practice
- Three approaches:
  - Dynamic Programming
  - Monte-carlo (another MC!)
  - Temporal-Difference [combination of DP and MC]

Side note: Reinforcement Learning

- These are often discussed in the context of reinforcement learning
  - Very different assumptions than in games
- RL may not have model of the environment
  - In games we always know the environment
- In RL, may get payoff at every time step
  - In games we (usually) only get a payoff at the end
- In RL the adversary is implicit in the environment
  - In games we have an explicit adversary
- In RL the future is discounted
  - In games a discount isn't needed

Performing learning

- Suppose we have an initial value function
  - Where can we get training examples?
- Expert games (eg early work by Tesauro)
- Play against expert
  - May not learn to play better than the expert
  - Expert may be too strong and always win
    - Can cripple the expert by forcing it to take some percentage of random moves

Performing learning

- Self play
  - *Really* helps to have a game that converges
  - Program may not learn strong play if it gets stuck in a local minimum
    - May always play the same way and not train on other types of play
  - Helps if good and bad situations can both be seen

Learning setup

- Need to choose features (more on this later)
- Need to choose a learning architecture
- Need to generate training

Eligibility Traces

- An eligibility trace is a sample of the plays that were made in a game from the beginning to the end
- Training can occur on eligibility traces
  - Have an associated payoff
  - For our purposes, payoff is only at the end

Monte-Carlo

- Play a game with the current value function
  - Often use a soft-max (small probability of a random move) instead of a pure maximization
  - Gives some chance of exploration and reaching every state in the game
- At the end of the game, take note of the score
  - Train all the states in the history of moves to predict the final score of the game
- This won't work if it is hard/impossible to reach the end of the game

UNIVERSITY OF DENVER

Dynamic Programming

- Update the predicted score based on the value of the neighboring states
  - Requires a model of the world
  - World can be probabilistic
    - Expected score used instead of deterministic score

UNIVERSITY OF DENVER

Monte-Carlo (revisited)

- Given a eligibility trace s_1, a_1, s_2, a_2, … a_{n-1}, s_n
- Followed by a reward r.
- Train function approximator with:
  - output($f(s_i)$) ← r
  - f is the features associated with state i

UNIVERSITY OF DENVER

DP (revisited)

- Given a eligibility trace s_1, a_1, s_2, a_2, … a_{n-1}, s_n
- Followed by a reward r.
- Train function approximator with:
  - output($f(s_i)$) ← output($f(s_{i+1})$)
  - where first training is output($f(s_n)$) ← r
  - (from i = n to i = 1)

UNIVERSITY OF DENVER

TD(λ)

- Combination of the two approaches
  - Given a eligibility trace s_1, a_1, s_2, a_2, … a_{n-1}, s_n
  - Followed by a reward r.
  - Train function approximator with:
    - output($f(s_n)$) ← r
    - output($f(s_{n-1})$) ← (1-λ) output($f(s_n)$) + λr
  - In general over *i* steps:
    - $[(1-\lambda)\sum_{j=i+1\ldots n} \lambda^{j-i}\text{output}(f(s_j))] + \lambda^{n-i}r$

UNIVERSITY OF DENVER

Alternate Approach: TD-Leaf

- Instead of using eligibility traces, consider the alpha-beta search process to be a learning procedure which produces training values
  - If we could search the whole game, the minimax value of the game would be the proper evaluation of the start state
- At each step, search & compute the minimax value of the game; use this for supervised learning

UNIVERSITY OF DENVER

TD-Leaf caveats

- Used in Chess, checkers
- Plays best when training depth is the same as the search depth in practice
  - Tournament settings use lots of time, meaning a long time is required to train