Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

Reminders

- Model-based learning learns $p(s' \mid s, a)$ (T) and $r(s, a)$ (R)
- UCT = Upper Confidence bound applied to Trees
  – Estimate score for each possible action

Overview

- Planning based approaches exploit information not available to human players $\implies$ better performance
- Combination of:
  – **Deep Learning** - progress in perception
  – **Reinforcement Learning** - policy selection
- Contributions:
  – Imitate slow MCTS planner to learn a policy
- Two components of the **perception problem**:
  – Partial observability - observations $\neq$ states
  – High dimensionality
- Arcade Learning Environment (ALE): 60 fps, all games finite (episodic) with immediate rewards
- Learns the POMDP, as the MDP would be intractable
- State of the art:
  – DQN - no hand-engineered features, 4 previous frames used as states
  – Planning based on UCT - "number of simulation steps needed to ensure any bound on the loss of following the UCT-based policy is independent of the state space size" - good for perception problem, but still slow computation

Key ingredients

- Play 800 games with UCT agent
- UCT agent uses internal game state to perform roll-outs
- **Imitate the agent to learn the policy**
- Combine 4 previous frames
- Frame skipping
  – select action on every 3rd of 4th frame and and repeats it on the skipped frames
- Adds the last layer to the CNN (DQN) network

UCTtoRegression

- Last layer regression
- Worst performing

UCTtoClassification

- Last layer softmax
- Distribution missmatch problem**!**

UCTtoClassification-Interleaved

- Solve the distribution missmatch similar to DAgger:

– Play 200 games with UCT → learn policy → play 200 games with learned policy, but store UCT actions → learn policy …
– continue data aggregation until 800 games played

Comments

- Overall well written, interesting and uncomplicated read
- Superior performance to previous state of the art
- Does not use hand-crafted features, but uses internal state of the game for UCT
- Imitation learning with data aggregation
- “We identified a gap between the UCT-based planning agent’s performance and the best realtime player DQN’s performance and developed new agents to partially fill this gap”

Not related TODOs

- Brush up on value function approximation in Sutton’s book and Silver’s course
- Go through the Policy Search tutorial