Differentiable Tree Planning for Deep RL

Greg Farquhar
In Collaboration With

Tim Rocktaschel, Maximilian Igl, & Shimon Whiteson
Overview

- Reinforcement learning
- Model-based RL and online planning
- TreeQN and ATreeC (ICLR 2018)
- Results
- Future work
Planning and Learning for Control
Reinforcement Learning

Actions

State & Reward

Reinforcement Learning Agent

Environment
Reinforcement Learning

- Specify the reward, learn the solution
- Very general framework
- Problem is hard:
  - Rewards are sparse
  - Credit assignment
  - Exploration and exploitation
  - Large state/action spaces
  - Approximation and generalisation
RL Key Concepts

- State (Observation)
- Action
- Transition
- Reward
- Policy: states $\rightarrow$ actions
Model-free RL: Value Functions

- Learn without a model of the environment
- Value function

\[ Q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \bigg| s_0 = s, a_0 = a \right] \]

- Optimal value function

\[ Q^* (s, a) = \max_{\pi} Q^\pi (s, a) \]

- Policy evaluation + improvement
The Bellman Equation

- Temporal (Markov) structure
- Bellman optimality equation
- Q-learning
- Backups
Deep RL

- Q → deep neural network
- Q-learning as regression
- Stability is hard
  - Target networks
  - Replay memory
  - Parallel environment threads
Deep RL - Encode and Evaluate?

$S_t \xrightarrow{\text{encode}} Z_t \xrightarrow{\text{evaluate}} Q$
Model-based RL
Online Planning with Tree Search
Environment Models

- State transition + reward
- Can be hard to learn
  - Complex
  - Generalise poorly to new parts of the state space
- Need very good fidelity for planning
- Standard approach: predictive error on observations
Model fidelity in complex visual spaces is too low for effective planning

Action-conditional video prediction using deep networks in atari games (Oh et. al 2015)
Model fidelity in complex visual spaces is too low for effective planning

Action-conditional video prediction using deep networks in atari games (Oh et. al 2015)
Another Way to Learn Models

- Optimise the true objective downstream of the model
  - Value prediction
  - Performance on the real task

- Our approach: integrate differentiable model into differentiable planner, learn end to end.
TreeQN: Encode

\[ S_t \rightarrow \text{encoder} \rightarrow Z_t \]
TreeQN: Tree Expansion
TreeQN: Evaluation
TreeQN: Tree Backup
TreeQN
Architecture Details

- Two-step transition function
- Residual connections
- State normalisation

- Soft backups
Training

- Optimise end-to-end with primary RL objective
- Parameter sharing
- N-step Q-learning with parallel environment threads
- Batch thread data together for GPU
- Increase virtual batch size during tree expansion for efficient computation
Grounding the Transition Model

- Observations
- Latent states
- Rewards
  - Inside true targets
ATreeC

- Use tree architecture for policy
- Linear critic
- Train with policy gradient
Results: Grounding

- Grounding weakly (just reward function) works best
- Maybe joint training of auxiliary objectives is wrong
Results: Box Pushing

- TreeQN helps!
- Extra depth can help in some situations
Results: Atari

- Good performance
- Makes use of depth (vs DQN-Deep)
- Main benefit from depth-1
  - Reward + value
  - Auxiliary loss
  - Parameter sharing
Results: ATreeC

- Works -- easy as a drop-in replacement
- Smaller benefits than TreeQN
- Limited by quality of critic?
Just for fun
Interpretability

- Sometimes (?)
- Firmly on model-free end of spectrum
- Grounding is an open question
  - Better auxiliary tasks?
  - Pre-training?
  - Different environments?
Future Work

● Lessons learnt for model-free RL:
  ○ Depth
  ○ Structure
  ○ Auxiliary Tasks

● Online planning:
  ○ Need more grounded models to use more refined planning algorithms
Summary

- Combining online planning with deep RL is a key challenge.
- We can use a differentiable model inside a differentiable planner and train end-to-end.
- Tree-structured models can encode a valuable inductive bias.
- More work is needed to effectively learn and use grounded models.
Thank you!