COMS E6998 Bandits and Reinforcement Learning

Instructors:
Agarwal and Slivkins
Columbia University

October 30, 2017

Abstract

This document is a compilation of lecture notes from Hierarchical Reinforcement Learning Papers. The PDFs for each of these papers is available in the /ref directory.

Contents

1 Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction [Jason]

Horde formulates the problem of Hierarchical Reinforcement learning as asking questions in the form of a value function, reward function, termination function, and terminal-reward function unrelated to those of the base problem to sub agents called "demons". These seem to take the form of reinforcmenet learning problems in and of themselves.

These demons learn "knowledge" related to these subproblems in the form of approximate value functions learned through gradient based methods. The authors assert that it is possible for the value function approach to extend to a theory of all knowledge. (Unsure why they focus on this)

This work specifically approximates the action-value function using the TD lambda algorithm. They also use the weight doubling trick, which I believe learns an additional parameter vector W that is intended to weight features and enable off-policy learning. (Not 100 percent clear on this)

Experiments are run on a "Critterbot robot", which predicts quantities such as time-to-obstacle and time-to-stop. I believe the idea here is that these prediction algorithms could be used in the future as sub-routines for a control system.

2 Hybrid Reward Architecture for Reinforcement Learning [Jason]

Proposes the Hybrid Reward Architecture, which takes as input a decomposed reward function and learns a separate value function for each component reward function.

If the optimal value function is very complex, then learning an accurate low-dimensional representation can be challenging or even impossible. They suggest replacing the true reward function with another reward function that has a smoother, easier to learn optimal value function. They decompose the reward function into n different reward functions, which are each assigned to a sub-agent. These agents learn in parallel and give values for actions back to an aggregator, which combines them into a single action-value for each actions. The aggregator can then use these action-values to select the current action.

- Performance objective: specifices what type of behavior is desired
- Training objective: provides feedback signal that modifies an agents behavior

The decompose the reward function into

$$R_{env}(s,a) = \sum_{1}^{n} w_k R_k(s,a)$$

The different agents can share lower-levels of a deep Q-network so they can also be seen as a single agent with multiple "heads". Where each head produces the action-value under a different reward function. They use the target function objective from the DQN paper. Q(0) return with target params that stay fixed for a short time.

This architecture can exploit doman knolwedge by

1. Removing irrelevant features
2. Identifying terminal states
3. Using pseudo-reward functions

They test their model on a task of picking up fruit on a grid environment. In this task HRA outperfomed DQN by a considerable amount.

They also run an experiment on Pac Man. Each head in this experiment learning the Q-values for getting to a particular location in the map. The perfomance boost on pac man is unbelievably good 10x.

3 A Deep Hierarchical Approach To Lifelong Reinforcement Learning In Minecraft

The authors seek to solve the problem of "Lifelong Learning" in which learners

1. retain knowledge they have learned
2. selectively transfer knowledge to learn new tasks
3. ensure 1 & 2 happen effectively and efficiently

They claim that any framework which does this must

1. learn skills (options in some literature)
2. learn a controller which determines when a skill should be used and reused
3. efficiently accumulate reusable skills

Knowledge is retained by incorporating reusable skills via a Deep Skill Module. There are two types of Deep Skill Modules, pre-trained Deep Skill Network arrays, or a multi-skill distillation network. The multi-skill distillation network is the author's novel variation of policy distillation applied to learning skills.

The authors call this a Hierarchical Deep Reinforcement Learning Network (H-DRLN).

DSN Array - Each DQN is independent

Multi-Skill network - A single Deep Neural Network represents multiple DSNs (sharing hidden layers)

The multi-skill network allows them to incorporate multiple skills in a single network

Since independent skills take control away from the H-DRLN, they needed to optimize the Skill Bellman Equation rather than the standard Bellman Equation, and use Experience Relays (queues) to track actions taken by skills.

Actions

1. Turn left/right 30 degrees
2. move forward
3. break a block
4. pick up an item
5. put down an item

DSNs that were trained

1. two navigation domains
2. a pickup domain

3. placement domain (break is the same as place)

The Authors set up an experiment in which the agent needed to navigate through rooms, pick up an object, place an object, and break a door, at which point it would finally recieve a small non-negative reward. In a domain with two rooms, the authors trained both the H-DRLN and vanilla DQN. The H-DRLN was able to solve the task after just one epoch, and generated significantly higher rewards.

4 Stochastic Neural Networks for Hierarchal Reinforcement Learning Michael Chess

This paper applies a similar style of analysis to the hybrid reward paper but uses a stochastic neural network as the primary learning agent. The really interesting feature of this approach is that stochastic neural networks naturally 'jump' out of local minima. This means that in the context of an extremely high dimensional function such as a Q function a minimal representation (in the parameter space) neural network can potentially avoid getting stuck in local minima.

References