Topics in Machine Learning: Course Project

Team Pikachu

Neural Logic Reinforcement Learning

Team Members: Aishwarya S. (20171046), Kritika P. (20161039), Pooja S. (20171403)
Link to Paper: https://arxiv.org/pdf/1904.10729.pdf
Author's Source code: https://github.com/ZhengyaoJiang/NLRL
Link to Github Repository: https://github.com/Kritikalcoder/NLRL

Paper Summary

Main Goals

Neural Logic Reinforcement Learning (NLRL) is used to represent the policies in reinforcement learning by first-order logic. NLRL is used to significantly improve interpretability and generalisability in supervised tasks.

Problem Definition

Most Deep Reinforcement Learning algorithms suffer a problem of generalising the learned policy, which makes the policy performance largely affected even by minor modifications of the training environment. Except that, the use of deep neural networks makes the learned policies hard to be interpretable.

Solution Approach

The proposed algorithm, NLRL, is based on policy gradient methods and differentiable inductive logic programming (DILP).

Experiments

Environments vary with different initial states and problem sizes.
Block Manipulation:

- Performance examined on 3 subtasks: STACK, UNSTACK and ON
- 5 entities: 4 blocks and a floor
- Predicates:

    - on(X,Y): block X is on entity Y
    - top(X): X is on top of the column
- Background knowledge:
    - Common to all tasks: isFloor(floor)
    - For ON task: goalOn(X,Y)
- States are represented using tuples of tuples.
- Action atoms(25): move(X,Y), move X on top of Y.
- move(X,Y) is valid only if both X and Y are on top of the stack or when Y is floor and X is on top of the stack.

Cliff-walking:

- 5x5, 6x6 and 7x7 versions of cliff walking environment is used.
- Rewards:
    - Cliff position, -1
    - Goal position, +1
    - For each step: -0.02
- Termination:
    - reaching cliff or goal state
    - Taking 50 steps
- State representation: current(X,Y)
- Action atoms:up(), down(), left() and right()

Conclusion

The average and standard deviation of returns are presented to show that the NLRL not only achieves a near-optimal performance in all the training environments but also successfully adapts to all the new environments designed in experiments.

Project Plan

We aim to enhance the existing solution in the following manner:

1. Adding feed-forward neural network based state encoding (to replace the hand-crafted encoding)
2. Showing results similar to those in the paper on a new domain
3. Implementing more policy gradient methods (besides vanilla) such as:
    a. Asynchronous Advantage Actor-Critic (A3C)
    b. Proximal policy optimization (PPO)
    c. Trust region policy optimization (TRPO).

    Showing results on all domains across these policy gradient methods.

Along with the above improvements in the solution, we will create a detailed blog (report) explaining the paper's techniques, our contributions and our findings.

Connection to Syllabus

Neural Logic Reinforcement Learning (NLRL) is based on policy gradient methods and differentiable inductive logic programming that have demonstrated significant advantages in terms of interpretability and generalisability in supervised tasks. NLRL represents the policies in reinforcement learning by first-order logic.

Similar to policy gradient methods, NLRL agent learns the policy for the environment but it overcomes the problem of generalising the policy for the given environment and task faced by policy gradient methods.

The author's implementation (and the paper) uses vanilla policy gradient for the NLRL. We will implement the NLRL agent using other policy gradient methods as an improvement over the existing model.

Requirements

Improvements

1. Achieve better state encoding using feed forward neural network.
2. Improve the agent by using different policy gradient methods.

Feasibility of experiments & Availability of resources

GPUs: 12 (4 per person) available
GPU Hours: 20 minutes to train + test for 120000 epochs per model (upper bound)

10 GB memory needed to save models for the experiments
Given the recources available, experiments are feasible.

Availability of Implementation

The original implementation of the paper has been made available by the authors. It is implemented using Tensorflow 1.14 and NumPy 1.14.5

Deliverables

S.No	Stage	Task	Member
1.	Base Model	Understanding the code	All
		Running the algorithm	All
2.	New Domain	Adding & testing on a new domain	Pooja
3.	Automated State Encoding	Implement NN state encoding	Aishwarya
4.	Alternate Policy Gradient Methods	Implementing A3C	Aishwarya
		Implementing PPO	Pooja
		Implementing TRPO	Kritika
5.	Experiments	Evaluating the models empirically	Kritika
6.	Final Report + Blog	Preparing Report + Blog	All
7.	Presentation	Preparing Presentation	All

Milestones

	Project Milestones	Deadline
1.	Understanding Problem Statement, Reading Background	October 26th, 2019
2.	Project Proposal Submission	October 28th, 2019
3.	Proposal Review	October 30th, 2019
4.	Phase 1: Base Model, New Domain, State Encoding	November 6th, 2019
5.	Phase 2: Alternate Policy Gradient Methods, Experiments	November 13th, 2019
6.	Phase 3: Making Presentation & Report + Blog	November 23rd, 2019
7.	Final Project Submission	November 24th, 2019
8.	Final Presentation	November 25-26th, 2019