Introducing Self Learning Into a Robotic Arm Using Deep Reinforcement Learning

Ogheneuriri Oderhohwo, Electrical and Computer Engineering

Introduction

Objective

Having a robot that can perform diverse actions in any given environment is a trending research area.

Significance

- Robots find diverse applications in medical, defense, automation and various industries.
- They perform in a dynamically changing environment.
- Explicitly creating programs for controlling a robot's action is not enough.

Need

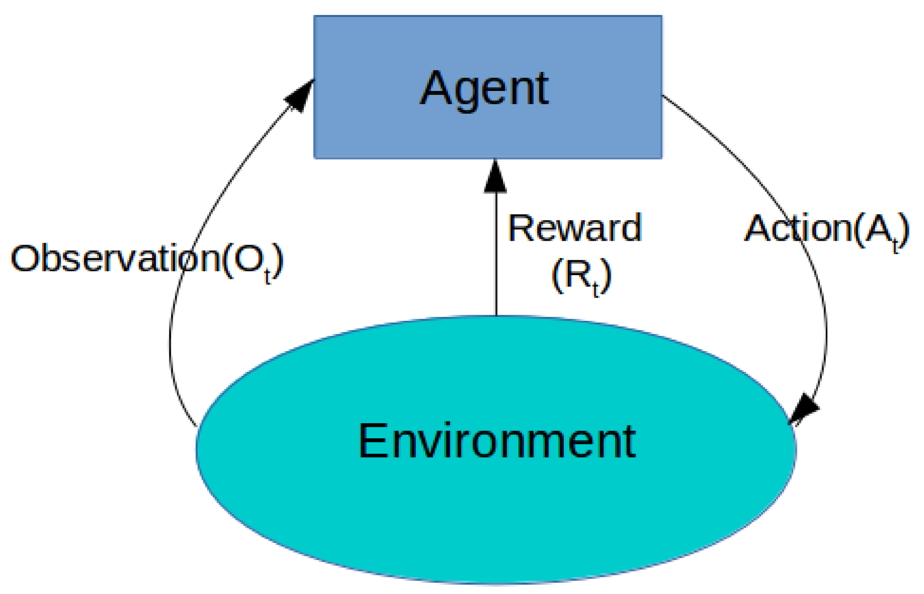
- A robot that can perform actions real-time in a resource constrained environment, without explicitly programmed to do so. is desired.
- Reinforcement Learning (RL); a goal-based machine learning approach can teach a robot to learn optimal actions in an environment through reward system [1].
- An RL method that can overcome the challenge of dimensionality for a robotic environment.

Problem Formulation

- What RL method can be deployed on a resource constrained device while achieving self learning for robot control?
- Here, we propose a Deep Reinforcement Learning (DRL) method using Deep Q network (DQN) to achieve self learning for a robot arm.

Methods

In Rl, an *agent* in a given *environment* is given *rewards* for the *actions* it performs based on the *observations/states* it finds itself. Model free environment is suitable for robotic implementation



Algorithm 1: deep Q-learning with experience replay. [2]

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

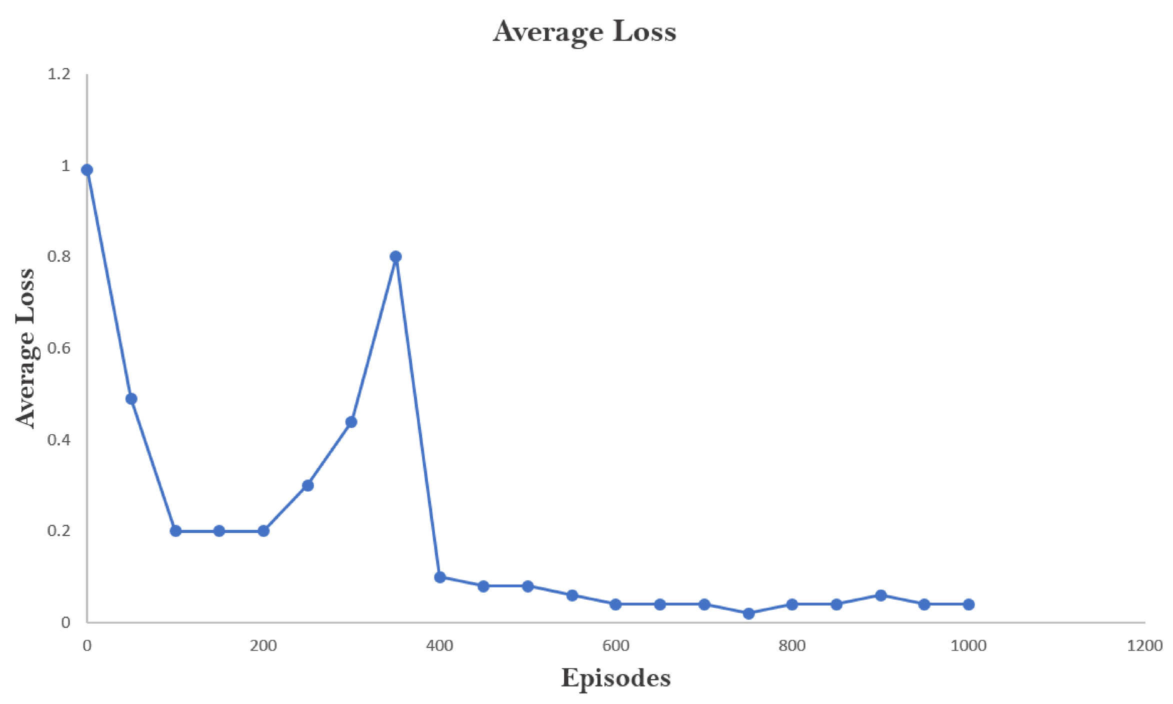
Q-Learning; An off-policy temporal difference method to predict value functions known as Q values.. Q values are probability values that shows reward expectations for a given state-action pair [3].

DQN; Neural networks to effectively handle prediction of Q values for a system of continuous states and actions and experience memory to store experiences for replay.

Results

The DQN is implemented on the Acrobot environment of the OpenAI Gym [4] classic control environments. The goal is for the robot to swing its lower link up to a given height..

Graph of average number of steps before robot achieves Swing up action for each episode.



Graph of average loss of predicted Q values after 1000 episodes.

- After several trials, a three layered neural network gave the best result.
- The input layer takes the observation space of the environment
- The second and third layer has nodes of 512 and 256 respectively and extracts useful information from the input
- The third layer predicts values for all possible actions in the action space of the environment.
- 1000 episodes consisting of 500 time steps(attempts) took 3 hours to train
- The robot failed the task at the initially episodes.
- After the 400th episode, the robot stabilizes to an average error rate of 0.04
- The robot achieves the swing up at the 120th attempt on the average after stabilizing.

Discussion

- The environment utilized has two links and joints with one of the joints being an actuator type. This conveniently models the effector ends of a robot.
- Modifying the environment to have an increased links and joints and further scaling to 3D would be sufficient for a real-world prototype.
- Using the DQN on the modified environment to train and then deploy on a resource-constrained environment such as the raspberry pi.

Conclusion

To achieve self-learning in a robotic arm, the off-policy DQN has proven to be a good choice of algorithm for the stochastic nature of robotic environments. Other policy methods such as Deep Deterministic Policy Gradient could be explored for comparison.

References

[1] R.S. Sutton and A.G.Barto. Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA, 1st Edition, 1998. ISBN 0262193981.
[2] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Pannershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu and D. Silver. Massively Parallel Methods for Deep Reinforcement Learning. https://arxiv.org/pdf/1507.04296.pdf
[3] Smruti Amarjyoti. Deep reinforcement learning for robotic manipulation-the state of the art. https://arxiv.org/pdf/1701.08878.pdf
[4] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schulman, J. Tang, W. Zaremba. OpenAI Gym. https://arxiv.org/pdf/1606.01540.pdf

Acknowledgements

Dr Syed Hasan, Department of Electrical and Computer Engineering, and Center for Manufacturing Research