Reinforcement Learning

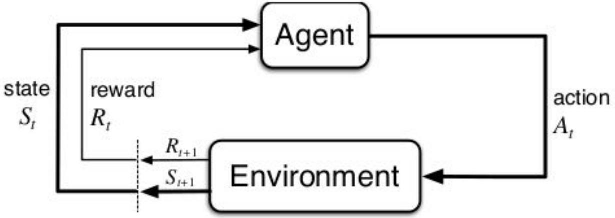
UPC DLAI Group8: Adrian, Bruno, Gianmarco and Jordi.

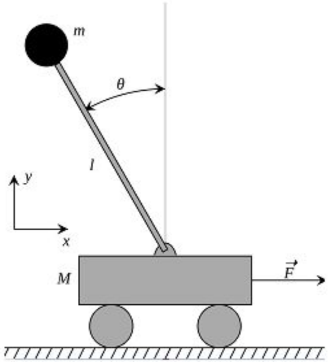
Agenda

- Introduction
- Deep Stochastic Policy Gradient Agent
- Deep Stochastic Policy Gradient Agent Experiments
- Deep Q Network Agent
- Conclusions

Introduction

Reinforcement Learning



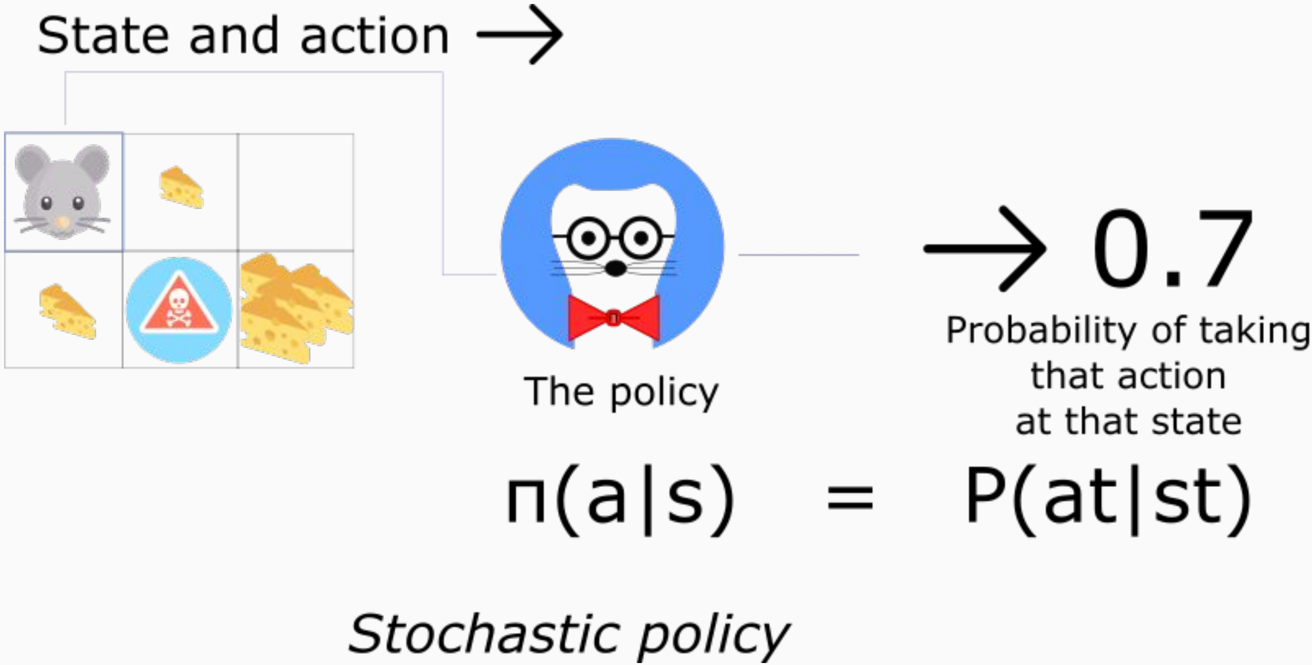


"... Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them..." Sutton, R. S.

Deep Stochastic Policy Gradient Agent I

Basics:

- **Stochastic**: Our policy outputs a probability distribution.
- **Deep**: It is modelled by a deep neural network
- **Policy:** Maps an state s to a probability disitribution among actions
- **Agent:** Takes actions by sampling from the policy distribution



Deep Stochastic Policy Gradient Agent II

Learning the policy:

- Measure how good is our policy using an **Objective Function J**

$$J_1(\theta) = V_{\pi\theta}(s_1) = E_{\pi\theta}[v_1] = \underbrace{\sum_{s \in S} d(s)}_{\text{State distribution}} \underbrace{\sum_{a \in A} \pi_\theta(s,a) R_s^a}_{\text{Action distribution}}$$

- Compute gradients:

$$\nabla_\theta J(\theta) = E_\pi[\nabla_\theta(\underbrace{log\pi(\tau|\theta)}_{\text{Policy function}})\underbrace{R(\tau)}_{\text{Score function}}]$$

- Update policy parameters Monte Carlo update

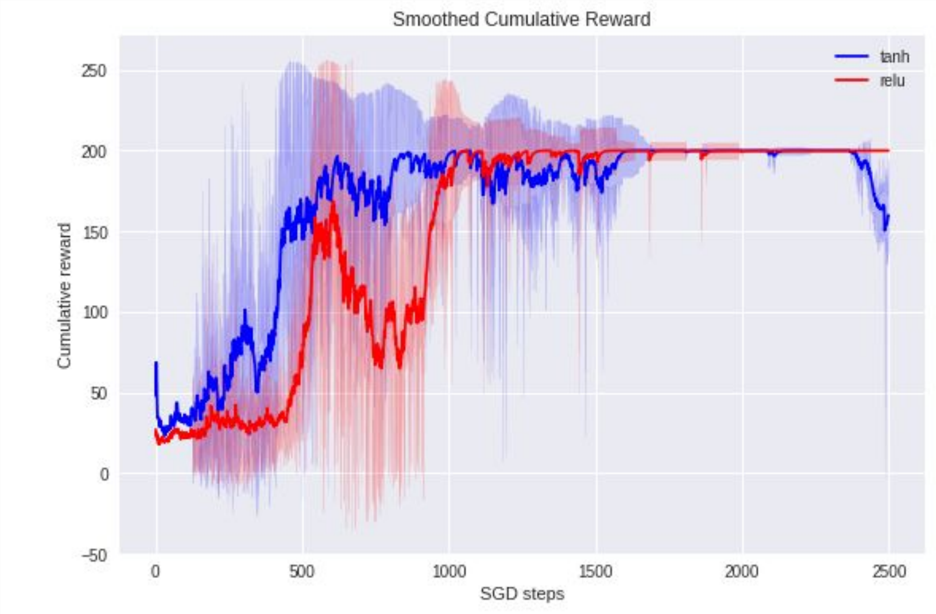
function REINFORCE
 Initialise θ arbitrarily
 for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ to $T - 1$ **do**
   $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
  end for
 end for
 return θ
end function

Deep Stochastic Policy Gradient Agent III

Tanh vs ReLU activations

Param:
learning_rate → 0.001
optimizer → adam
batch_size = 32
n_experiments= 10

Blue: FCN using tanh activations
Red: FCN using ReLU activations



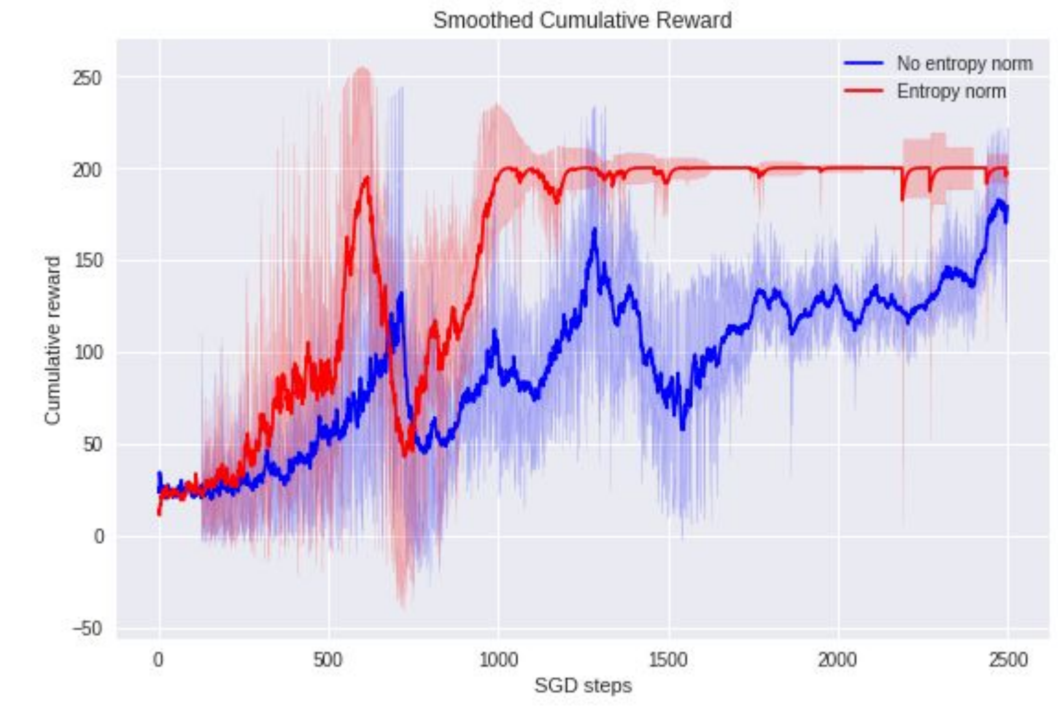
Deep Stochastic Policy Gradient Agent III

Entropy Normalization:

Encouraging our agent to explore the environment.

Param:
learning_rate → 0.001
optimizer → adam
batch_size = 32
n_experiments= 10

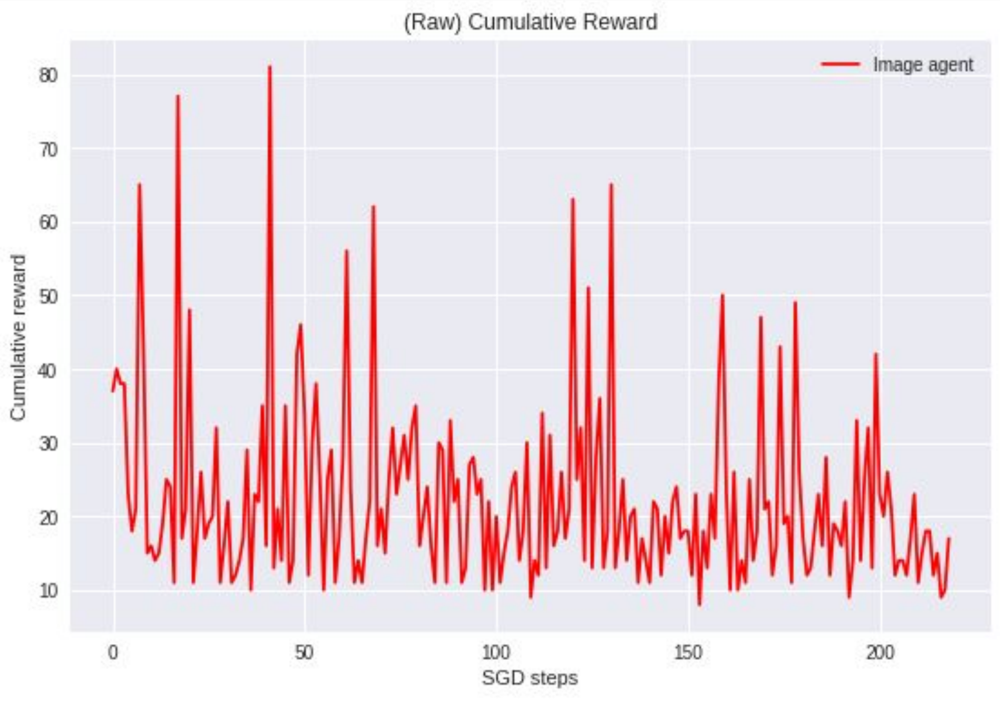
Based on the publication Understanding the impact of entropy on policy optimization, Nov 2018



Deep Stochastic Policy Gradient Agent III

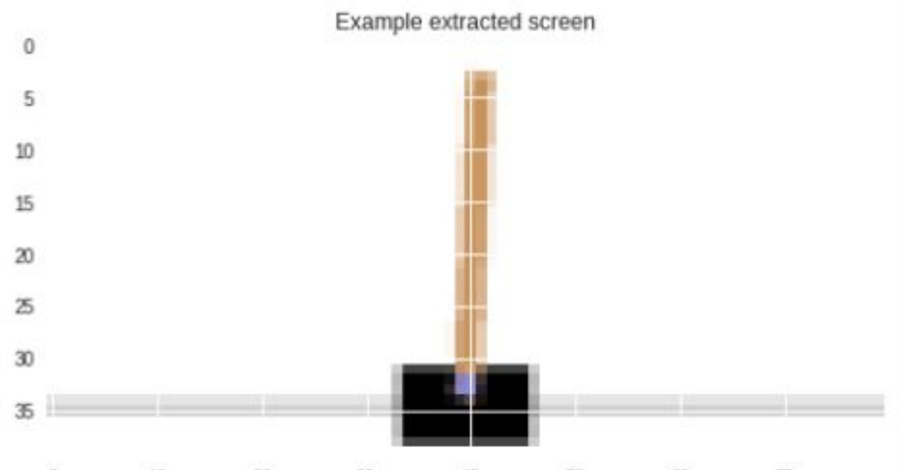
State as RGB frame

- Learning the policy from images is much more challenging.
- New architecture: 3 convolution layers + 2 FC + softmax activation.
- Poor results



Reinforcement learning with DQN

- State
  - Current frame from the environment
  - Cropped
  - Previous frame subtracted
- Replay memory
  - Memorize the previous experience
  - Randomly samples



The Network

- 3 convolutional layers with batch normalization
  - Size 16-32-32
- One linear layer with 448 units
- ReLU activation functions
- Huber loss function
- RMSprop optimizer

Similar to mean squared error, but less sensitive to outliers

Training results

- Our network did not converge
- Changing some Hyper-parameter did not help
- Perhaps it is possible to tune them better

Conclusions

- We implemented two different approaches with differents activation functions
- Real environments like an RGB image state are much more complex.
- Encouraging exploration becomes essential in complex environments.
- Reinforcement learning is not easy to converge

Thank you for your attention!

Reinforcement Learning

UPC DLAI Group8: Adrian, Bruno, Gianmarco and Jordi.