Optimus is here!

Self taught game playing AI from pixels

BY

- ***Raja Ayyanar***
- ***Jhansi***
- ***Divya***

RAMAIAH UNIVERSITY OF APPLIED SCIENCES | INNOVATION CENTRE

Introduction

- Artificial intelligence agent playing games.
- How to make a machine Intelligent?
  - Supervised, Unsupervised and Reinforcement
- How smart AI is, as of today in gaming field?
  - Smart enough to beat any human in playing 2D games
  - Not to forget Google's AlphaGO Zero which demolished stockfish and komado in Chess.
  - Learns just like Human from Experience

RAMAIAH UNIVERSITY OF APPLIED SCIENCES | INNOVATION CENTRE

Can we use Supervised Learning?

- Need Large datasets of games played.
- Search space is very huge.
- Will work for only onegame at a time.
- Need different architecture for different games.
- Impossible to make because of overfitting in DNN.

Hardness of problem

- State action pi policy can not be optimized by supervised learning.

Deep Reinforcement Learning

- Definition:

  -branch of Machine learning which learns from trial and error. Based on experience and rewards.

- Markov Decision Process

  -Mathematical formulation of reinforcement learning scenario

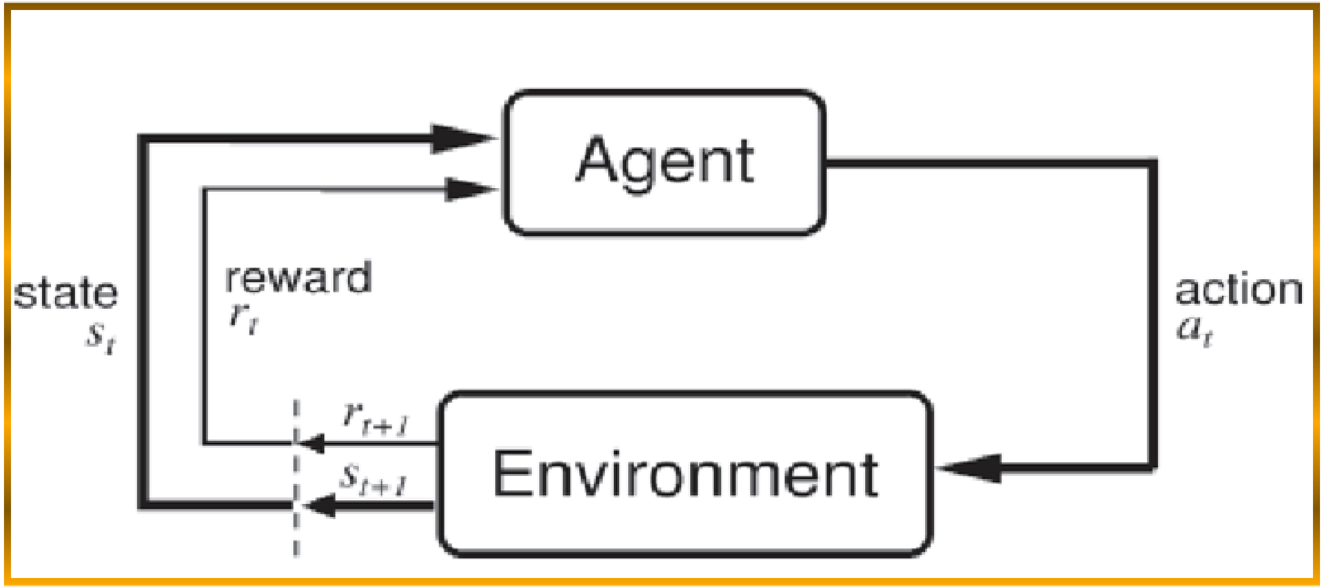
Sets of state,

Sets of Actions

Reward function

Policy pi

Value function, V



Design: Deep Q Learning

- Q function is a function which predicts action for a particular state

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    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 $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

RAMAIAH UNIVERSITY OF APPLIED SCIENCES | INNOVATION CENTRE

S.No	Content	Values	description
1	Architecture	6 layer DNN	
2	Input	State values from game simulator	Game Pixels or pre-processed info from pixel data
3	Output	Probability percentage of each actions	Actions are the possible variables that can be controlled for a particular game
4	Hidden Layer 1	160 neurons, Relu Activation	Hyper parameter
5	Hidden Layer 2	250 neurons, LRelu activation	Hyper parameter
6	Hidden Layer 3	500 neurons, Relu Activation	Hyper parameter
7	Hidden Layer 4	250 neurons, LRelu Activation	Hyper parameter
8	Hidden Layer 5	100 neurons, Relu Activation	Hyper parameter
9	Output Layer	No of actions as no. of neurons, Softmax layer	Number of possible actions is same as neuron output
10	Optimizer	ADAM	Generalized Adagrad
11	Learning Rate	0.001	Hyper parameter
12	Loss function	Cross Entropy loss	Log loss will be differentiable
13	Optimization technique	Back prop	Deterministic approach
14	Regularization	Dropout	To avoid over fitting

Simulation: Game Environment

- Using OpenAI’s Gym Library
- Pixels as a input
- Convolution Neural Network can be used if we use without preprocessing
- Gym creates an game environment which takes an action for particular state and gives reward, new state and info.

https://github.com/openai/gym/wiki/Table-of-environments

Total of more than **80 different Game** environment is available. Our code will run for all games with small tweaks.

Experimentation

- Results Obtained: we chose easiest game for demo(To save training time)

Training: For cartpole game

No of trial games	Mean score	% of prediction correctly	% of action 1 is chosen	% of action 0 is chosen
10	9	4.3%	70	30
100	14	4.1%	36	64
500	17	7.00%	69	31
1000	30	10.56%	60	40
5000	190	69.47%	55	45

Performance

- Testing Result

  -After Learning for 5000 trials.

s.No	No.of test games	Mean score	Max score	% of action 1 chosen	% of action 0 chosen
Validation 1	10	254	390	48.46	51.53
Validation 2	10	188	368	52.13	47.87
Validation 3	10	289	480	50.88	49.11

Demonstraion

- Tools used:

S.No	Content	Value
1	Programming Language	PYTHON
2	Game Environment	GYM library
3	Deep Neural Network	Tf-layer, Keras with Tensorflow Backend
4	Operating System	UBUNTU (because gym works better with LINUX)
5	Environment	Docker, Anaconda
6	IDE	Spyder, Jupyter

Application

- Generalised AI for all 2D games
- Self learning Helicopter flying
- Self taught Car drifting
- Self taught crawling robots
- Finance Market- to learn optimal trading strategy
- Manufacturing sector- Robot to pick tools and arrange
- Robot learning

Potential for commercialization

- Deep Reinforcement Learning:

Combination of Deep Learning + Reinforcement learning has led to:

1. AlphaGo beating world champion at GO
2. Self-driving Cars (Tesla)
3. Video-game playing machines that plays at superhuman level

RL has been around since the 70s But proof of its greatness has only been recently demonstrated

RAMAIAH UNIVERSITY OF APPLIED SCIENCES | INNOVATION CENTRE

Budget

- To build game playing AI – Zero
- Since most software and libraries are open source.

- But a GPU availability will be useful. Since most of the games took around 10 to 36 hours for training in a CPUs.

Findings and Conclusion

1.The Potential of DQL and DDQL for Reinforcement learning to build an more intelligent machines are yet to come.

2.RL is on bleeding edge of what we can do with AI.

3.As Professor Andrew NG said, ”Artificial Intelligence is new electricity”. We are near to see AI everywhere soon.


Q
&
A

Credits

- Paper from Google DeepMind's Deep-Q-learning algorithm
- Andej Karpathy's blog
- CS231n classroom lecture from stanform university
- Keon.io for demo/Explanation on RL
- Sentdex youtube channel for lecture videos on coding RL and python libraries.