Human-Level Control Through Deep Reinforcement Learning

Suraj Kumar

Department of Computer Engineering
Aligarh Muslim University

surajkumar@zhcet.ac.in

October 10, 2018

Overview

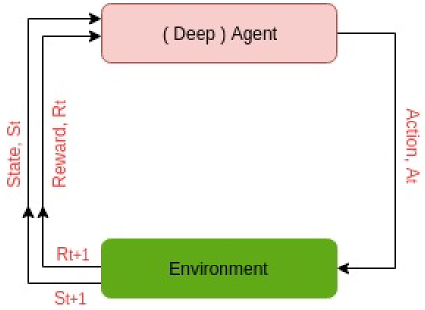
1. AI Learning
2. Deep Reinforcement Learning(Deep RL)
3. Markov Decision Process(MDP)
4. Bellman Equation
5. Deep Q-Network(DQN)

AI Learning

Supervised Learning	Unsupervised Learning	Reinforcement Learning
• Data and Labels	• Non labelled data	• Time delayed label
• Learn a function to map data to its label	• Learn some hidden structure of the data	• Learn how to take actions in order to maximize reward
• Classification, Regression, Object detection	• Clustering, Association, Dimensionality reduction	• Game, Control and Information theory, Robotics, Genetic algorithms

Deep Reinforcement Learning

- Deep RL = RL + Deep Learning



- Agent interacts with an environment through a sequence of observations(states), actions and rewards.
- Goal of the agent is to select actions in a fashion that maximizes cumulative future reward.

Markov Decision Process(MDP)

- MDP provide mathematical formation for the RL problem.
- Current state completely characteristics the state of the world.
- MDP is defined by five tuples :
  1. S: set of finite states.
  2. A: set of finite actions.
  3. R: Immediate reward received after transitioning from one state to another due to some action.
  4. Transition Probability($\mathbb{P}$): Probability of transitioning from one state to another.
  5. Discount Factor($\gamma \epsilon$ [0, 1)): Represents the difference in importance between future and present rewards.

MDP: Policy

- A policy is a function $\pi : \text{S} \rightarrow \text{A}$ that specifies what action the agent should take in any given state.

- Policy can be of two type :
  1. Deterministic Policy : Action depend on input, $\pi : S \rightarrow A$
  2. Stochastic Policy : Agent choose the action randomly, $\pi : SxA \rightarrow [0, 1]$
     $\pi(a|s) = P(At = a|St = s)$

- *Optimal policy* π^* : policy that maximize cumulative reward function.

$$\pi^* = argmax_{\pi}\mathbb{E}\big[\sum_{t\geq 0}\gamma^t r_{\mathbf{t}} \mid \pi\big]$$

- Cumulative future reward is the discounted sum of future rewards throughout an episode.

$$R = \sum_{t=0}^{T}\gamma^t r_{\mathbf{t}}$$

Value Functions

1. State Value Function : returns the expected reward for selecting a certain state s.
   - $V^{\pi} = \mathbb{E}_{\pi}\left[R_{\mathbf{t}} \mid s_{\mathbf{t}} = s\right]$
2. Action Value Function : returns the expected reward for using action a in a certain state s
   - $Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R_{\mathbf{t}} \mid s_{\mathbf{t}} = s, a_{\mathbf{t}} = a\right]$
   - Optimal Action value function: maximum sum of rewards discounted by γ at each time step t, achievable by a behaviour policy π, after making an observation (s) and taking an action (a).
   - $Q^{*}(s, a) = \max_{\pi}\mathbb{E}\left[\ \mathrm{r}_{\mathbf{t}} + \gamma r_{\mathbf{t+1}} + \gamma^{2} r_{\mathbf{t+2}} + \ldots \mid s_{\mathbf{t}}, \mathrm{a}_{\mathbf{t}}, \pi\right]$

Bellman Equation

- In MDP value of any state can be written as sum of immediate reward and value of state that follows.
- $V^{\pi} = \mathbb{E}_{\pi}\left[r_{\mathbf{t+1}} + \gamma V^{\pi}(s_{\mathbf{t+1}}) \mid s_{\mathbf{t}} = s\right]$

Deep Q-Network(DQN)

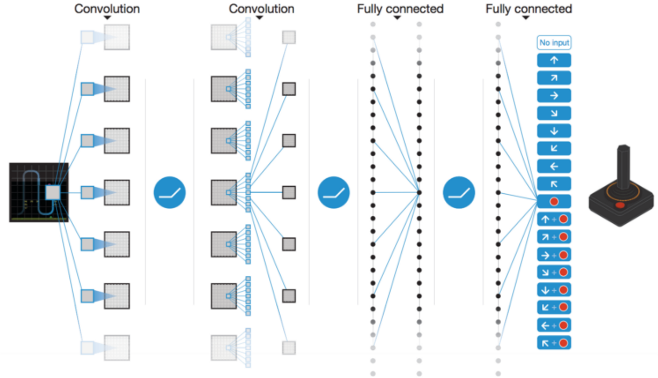
- DQN combine RL with deep neural networks.
- DQN agent receive only the pixels and the game score as inputs.
- Learn successful policies directly from high dimensional sensory inputs using end to end deep q-network agent.
- Proposed DQN is tested on 49 Atari games.

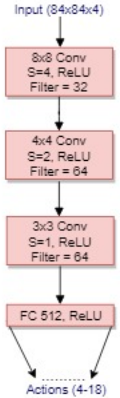
- Challenge :
    - RL is unstable when a nonlinear function approximator is used to represent action value function.
    - correlation between sequence of observations may updates in Q that can change the policy.

Nobel Variant of Q-Learning - Deep Convolutional Neural Network

- Correlation between observations can be address with two key ideas
  1. Experience Replay randomize over data to remove correlation and smooth over changes in data distribution.
  2. An iterative Update adjusts the action value toward target that updates periodically to reduce correlation.
- Stable method for training neural network in RL setting are Neural Fitted Q-iteration
  - Involves the repeated training of network on hundreds of iterations.
  - Inefficient for large neural networks.
- Approximate value function can efficiently parameterize using Deep Convolutional Neural Network to perform experience replay.

Architecture





Algorithm

- Deep Q-learning with experience replay

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 $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    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 $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Dataset

- Uses Raw Atari 2600 frames which are 210x160 pixel images.
- Data Preprocessing
  1. Encode a single frame by take the maximum value for each pixel colour value over the frame being encoded and the previous frame.
  2. Extract the Y channel, also known as luminance, from the RGB frame and re-scale it to 84x84.

Estimating the Q-Network

- Loss function is the simple squared error
  $L = E[(r + \gamma max_{\mathbf{a}'}\ Q(s', a') - Q(s,a))^2]$

- Q-Learning gradient
  $\frac{\partial L(w)}{\partial (w)} = \mathbb{E}\big[(r + \gamma \text{ max }_{\mathbf{a}'}\ Q(s', a') - Q(s,a))\frac{\partial Q(s,a,w)}{\partial (w)}]$

- Optimize objective end-to-end by SGD(stochastic gradient descent).

References

1. Volodymyr, Koray, David, Andrei, Joel, Marc, Alex, Martin, Andreas, Georg, Stig, Charles, Amir, Ioannis, Helen, Dharshan, Daan, Shane and Demis, "Human-level control through deep reinforcement learning", Nature 2015, 10.1038/nature14236
2. Daiki Kimura, "DAQN: Deep Auto-encoder and Q-Network", arXiv:1806.00630v1 [cs.CV] 2 Jun 2018.
3. http://cs231n.stanford.edu/
4. https://www.theschool.ai/courses/move-37-course/

Thank You