논문 요약

<Playing Atari with Deep Reinforcement Learning>

1. Abstract

-High-Dimensional Sensory input으로부터 reinforcement learning을 통해 control policy를 성공적으로 학습하는 deep learning model을 선보임

-이 모델은 Atari는 CNN 모델을 사용하며, 변형된 Q-learning을 사용하여 학습되었다

-여기서 이 모델의 input은 raw pixels이고, output은 미래의 보상을 예측하는 value function

-Atari games 중에서 7개의 games 환경을 통해 이 방법을 적용했고, 7개 중에서 6개의 game에서 이전의 모든 접근들을 능가했고, 6개 중에서 3개는 game에서 인간을 넘어섰다. (성공적인 결과!)

1. Introduction

vision이나 speech와 같은 high-dimensional sensory inputs로부터 agent를 학습시키는 것은 RL의 오랜 과제였음.

이러한 데이터를 RL에 성공적으로 적용시킨 사례들은 Linear한 value function이나 policy representation에 결합된 hand-crafted features에 많이 영향을 받았음. 또한, 이러한 Features들의 quality에 성능이 역시 많이 좌우되었음.

딥러닝이 발전함에 따라, Vision, Speech와 같은 고차원의 데이터들을 추출하는 것이 가능해짐 ->딥러닝에서는 이러한 고차원의 데이터들을 입력으로 사용하여 CNN, Multi-Layer Perceptrons, restricted Boltzmann machines, RNN 등을 통해 지도 및 비지도 학습에 사용함.

결과는 성공적이었고, 딥러닝을 강화학습에 도입하는 연구를 진행하게 됨!

\*\* 이 때 딥러닝과 강화학습의 차이는?\*\*

Deep Learing(딥러닝):

autonomous, slef-teaching system으로 어떤 pattern을 찾기 위한 알고리즘을 학습시키기 위해 존재하는 데이터를 사용함. 예를 들어, 이미지에서 강아지를 찾기 위해 딥러닝을 사용한다고 할 때, 강아지가 있는 이미지와 없는 수백만장의 이미지를 학습 데이터로 사용하다 보면 edges, shapes, colors 등과 같은 이미지 데이터를 사용하고 분류하는 패턴을 정립하고, 이러한 패턴들은 학습데이터를 통해 생성한 model을 기반으로 새로운 이미지를 보고 고양이의 포함여부를 밝힘. 이처럼 딥러닝을 통해 계속된 cycle을 거쳐 학습을 진행하여 패턴을 좁히고, 각 cycle마다 예측을 개선함

Reinforcement learning(강화학습):

시도와 실패(trial and error)을 통해 학습하는 autonomous, self-teaching system으로, 보상의 극대화에 초점을 두고 행동하며, 최상의 결과를 얻기 위해 학습함. 여러가지의 서로 다른 행동을 취하며 결과에 대한 피드백을 받으며 해당 행위가 최고의 보상을 주는지 학습함. 그리고 항상 최고의 결과를 내기 전까지 수많은 학습을 통해 알고리즘을 수정하고 변형하며 강화함

\*차이!

둘 다 모두 자동적으로 학습하는 시스템이지만, deep learning은 training set으로부터 학습하고, 진행한 학습을 새로운 데이터에 적용시키지만, reinforcement learning은 최고의 보상을 위해 계속해서 동적으로 학습하며 행동을 조정함. 이 둘은 상호작용이 가능하며 이를 deep reinforcement learning이라고 함.

그러나, 딥러닝을 강화하는 과정에서 몇가지 문제점 발견된다

1. 대부분의 DL Application은 손으로 label된 많은 양의 training data를 필요로하는 반면에, RL 알고리즘은 sparse, noisy, delayed한 reward signal이라는 scalar 값을 통해 학습을 해야 한다.

* 딥러닝은 input에 대한 결과가 직접 작성되어 계산의 시간이 적지만, RL에서는 어떠한 행위를 하면 trial and error을 통해 그 행위에 대한 결과를 알기까지 시간이 필요한 delay가 어려움을 자아낸다

1. 딥러닝 알고리즘에서 각 데이터들은 독립적인 반면에, 강화학습에서는 하나의 행위가 다른 것들과는 연관성이 높다

* 현재 상태의 행동이 다음 상태의 보상에 영향을 주는 등 상호연관성이 매우 높음

1. RL에서는 알고리즘이 새로운 행동들을 배울 때마다 data의 distribution가 변하게 되는데, 이는 데이터의 분포가 고정되어 있다고 가정하는 딥러닝의 assumptioin과 충돌하여 문제가 될 수 있다

이 논문에서는 CNN이 복잡한 RL환경에서 원시 비디오로부터 성공적인 Control Policy를 학습할 수 있음을 증명합니다. CNN은 변형된 Q-Learning을 통해 학습되며, weight를 update하기 위해 stochastic gradient descent를 사용합니다. 또한 correlated data와 non-stationary distributions의 문제를 약화시키기 위해 Experience Replay Memory을 사용하는데, 이것은 무작위로 이전의 transition을 추출하여 training distribution이 원활해지게 합니다.

DeepMind는 하나의 Neural Network를 만들어 가능한 많은 게임을 학습시키는 것을 목표로 하였고, 게임에 대한 특정 정보나 게임의 우위를 위한 데이터 등을 제공하지 않았습니다. 오직 비디오의 시각 데이터와 Reward 그리고 터미널로부터 오는 신호 그리고 가능한 몇개의 행등으로만 학습을 진행하였습니다. 또한 다양한 게임들에 대해 동일한 Network Architecture와 Hyperparameter를 사용하였다.

1. Background

Agent가 환경(EE, Atari Emultator) 와 상호작용하는 task를 생각해보겠습니다. 각 time-step마다 Agent는 할 수 있는 행동( aTaT) 들 중에서 한가지를 선택하게 됩니다. Action이 전달되면 Emulator는 내부 상태를 변경하고 게임 점수를 수정하게 됩니다. 여기서 Agent는 게임의 내부 상태를 알 수 없고, 단지 현재 화면을 나타내는 raw pixel 의 vector로 이루어진 이미지와 game score의 변화를 나타내는 reward(rtrt) 만을 전달받습니다.   
=> 즉, 매 time-step마다 Agent가 action(atat) 를 선택하면 Emulator를 통해 state를 수정하고 reward(rtrt)가 return 됩니다.    
  
하지만 게임의 점수는 현재의 행동뿐만 아니라 이전에 거쳤던 일련의 행동에 의존하여 결정되고, 행동에 대한 피드백은 수천 회의 time-step이 진행된 후에 받게됩니다.

하지만 Agent는 오직 현재의 장면 만을 관찰하기 때문에, 전체적인 상황을 이해하기 힘듭니다. 그래서 이를 해결하기 위해 action의 sequence를 관찰하고 이를 통해 학습을 진행합니다. 이러한 Formalism은 크지만 유한한 Markov Decision Process(MDP)을 야기하는데, 여기서 각 시퀀스는 별개의 상태에 해당한다. 결과적으로 우리는 MDP에 standard한 reinforcement learning method를 적용할 수 있고, 이것은 시간 t에서의 상태를 표현하기 위해 전체 시퀀스를 사용함을 의미합니다. 그래서 Agent의 목표는 Future Reward을 극대화시키는 방식으로 action을 선택하고 이를 Emulator에 전달는 것입니다. 시간이 오래 지날수록 그 reward의 가치는 점점 내려가는데, 이를 적용시키기 위해 discount factor rr이 정의된다.  

시간 t에서 discounted factor가 정의된 reward를 RtRt라고 할 때, 해당 값은 아래와 같이 정의합니다. 

Rt=∑t‘=tTrt‘−trt‘Rt=∑t‘=tTrt‘−trt‘

이 식에서 T는 game이 종료되는 time이고, 현재의 시점에서 종료 시점까지 reward값에 reward factor의 t'-t 지수 값을 한게 됩니다. 그리고 어떤 일련의 행동(s)를 파악한 후에, 취한 행동(a)을 통해 얻을 수 있는 Expected 최대값을 반환하는 최적의 action-value function인 Q∗(s,a)Q∗(s,a)를 정의하였는데 해당 함수는 아래와 같습니다. 여기서 ππ 는 stst 에서 atat 를 Mapping하는 Policy Function이다. 

Q∗(s,a)=maxπE[Rt|st=s,at=α,π]Q∗(s,a)=maxπE[Rt|st=s,at=α,π]

최적의 Q-Function은 Bellman Equation이라는 중요한 특성을 따른다.    
  
*=> Bellman Equation에 대한 개념이 부족하면*[***여기***](https://mangkyu.tistory.com/64)*를 참고해주세요!*  
  
  
  
이 방정식은 sequence s′s′의 다음 time-step에서 최적의 Q∗(s,a)Q∗(s,a) 값이 모든 모든 행동 a′a′ 에 알려져 있다면, 최적의 전략은 r+γQ∗(s′,a′)r+γQ∗(s′,a′) 의 Expected Value를 최대화하는 것이다. 이를 수식으로 표현하면 다음과 같다. 

Q∗(s,a)=Es′∼E[r+γmaxa′Q∗(s′,a′) ∣∣ s,a].Q∗(s,a)=Es′∼E[r+γmaxa′Q∗(s′,a′) | s,a].

많은 강화학습 알고리즘에서는 Q-Function을 Estimate하기 위하여 Bellman Equation을 Iterative Update한다. Value Iteration 알고리즘은 매 i번째 Iteration마다 아래와 같은 Procedure를 수행하게 된다. 

Qi+1(s,a)=E[r+γQi(s′,a′) ∣∣ s,a]Qi+1(s,a)=E[r+γQi(s′,a′) | s,a]

이런 Value Iteration Algorithm은 MDP에서 Qi→Q∗ as i→∞Qi→Q∗ as i→∞ 라는 것이 알려져있다.