Playing Atari with Deep Reinforcement Learning

0. Abstract

강화학습할 때 고차원 인풋으로부터 policy를 조절하는 것을 배우는 딥러닝 모델

Q-러닝으로 학습된 CNN

인풋에는 raw 픽셀, 아웃풋에는 미래 보상을 추정하는 함수가 나옴.

1. Introduction

고차원 sensory 인풋으로부터 직접적으로 조절하는 것을 학습하는 것이 강화 학습의 가장 큰 문제

최근에는 다양한 신경망 구조와 지도 학습, 비지도 학습으로 높은 수준의 피처 추출 가능

딥러닝에서의 문제점

- 딥러닝 모델은 주로 라벨링 된 데이터가 많이 필요함. 강화학습은 sparse, nosiy, delayed된 스칼라 reward signal 한 데이터 학습에 어려움이 있음
- 딥러닝 모델이 데이터 샘플이 독립적이라고 가정함

이 논문에서는 CNN으로 문제점 해결.

다양한 Q-learning 알고리즘과 SGD로 가중치 학습.

상관 관계 있는 데이터 문제 완화를 위해 랜덤한 샘플 사용

Atari 2600 게임을 실행해 많은 게임을 학습할 수 있는 단일 신경망 생성 목표

2. Background

매 시간마다 에이전트는 게임 세트에서 액션을 선택. 액션은 emulator로 보내져서 내부 상태와 게임 점수 수정. 게임 점수의 변화에 따라 보상 받음.

에이전트는 현재 화면 이미지만 관찰하므로 현재 시점을 현재 화면만으로 이해하기 불가능하므로 이전 시점들의 관찰된 것들을 고려하고 게임 전략을 익힘.

모든 시퀀스가 유한해 markov decision process 적용 가능

에이전트의 목표는 미래 보상을 최대화하는 액션 선택하는 것. 매 시간마다 보상이 변하는 비율 감마 정의하고 반환율 정의함.

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'},$$

action value Q*(s,a) : 최적의 액션 밸류 함수

Bellman equation: 최적의 action value 함수 구하는 방식. 다음 시간에서 가능한 행동에 대한 최적의 action value 알고 있으면 최적의 전략은 r + γQ*(s,a)의 평균 최대화.

반복적 방법으로 action-value 함수 최적화. 강화 학습에서 주로 선형 함수이지만 가끔 비선형 approximator가 사용됨.

Q-network: 가중치 세타를 가진 신경망 함수 approximator.

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right],$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right].$$

타깃은 반복마다 loss function 최소화하는 방향으로 훈련

p(s,a): behavior distribution

Q-learning 알고리즘: model-free, off-policy

3. Related Work

TD-gammon: 가장 알려진 강화학습. model-free 강화 학습 알고리즘 사용. 다층 구조 퍼셉트론으로 근사화

stochasticity로 인해 value 함수 완화.

Q-learning은 선형 함수를 사용해 convergence 보장

NFQ(neural-fitted Q-learning): RPROP 알고리즘으로 파라미터 업데이트. 반복할 때 비용이 큼. 단순한 실제 세상 문제에 성공적으로 적용됨. deep autoencoder 를 사용

4. Deep Reinforcement Learning

강화 학습 알고리즘을 deep neural network와 연결해 RGB 이미지에 적용하고 학습 데이터를 SGD로 효율적으로 업데이트하는 것을 목표로 함.

Experience replay 매커니즘을 사용해 데이터를 저장하고 ϵ-greedy를 사용해 state space 탐색.

ϵ-greedy: 액션을 1-ϵ 확률로 실행하고 나머지에는 랜덤한 액션 실행하면서 다양한 state 탐색

기존의 Online Q-learning보다의 장점

- 한 데이터가 여러번 업데이터에 사용되어 효율적
- 랜덤 샘플로 학습해 gradient 편차 줄임.
- off-policy인 experience replay를 사용해 파라미터 감마의 발산 방지

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

4.1 Preprocessing and Model Architecture

128 색깔의 210*160 픽셀 이미지 데이터.

RGB를 gray-scale로 변환하고 110*84 픽셀로 다운 샘플링 진행. 최종 단계에 84*84로 잘라서 데이터 준비.

Q를 신경망으로 구하는 방법

- 이미지와 액션을 신경망에 넣어 Q값 구함. 각 액션마다 forward pass 를 실행해야해서 반복이 많이 필요.
- 신경망에 마지막 출력 노드를 하나당 하나의 액션으로 정의. 노드에 나온 출력값이 액션에 따른 Q값에 대응. 시간 복잡도 효율적, action space 넓어져 효과적.

84*84*4 ⇒ 8*8*16 ⇒ ReLU ⇒ 4*4*32 ⇒ ReLU ⇒ 256개 은닉 노드층 ⇒ 출력 노드는 유효한 액션 노드 개수와 대응됨.

5. Experiments

데이터: Atarti 2600 game

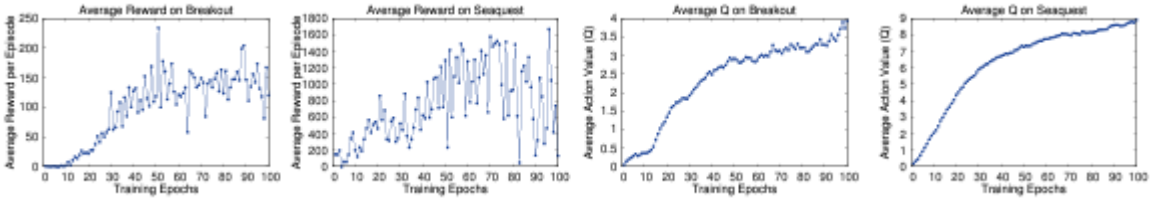
7개의 유명한 게임에 대해 성능 테스트. 모든 게임에 같은 조건으로 실험. 리워드는 -1,0,1 중 하나로 지정해 여러 게임에서 같은 학습률 사용 도움.

RMSProp 알고리즘 사용. 미내배치 크기 32.

학습에 천만 프레임 사용하고 최근의 백만 프레임을 replay 메모리에 사용. 처음 학습률 값을 1에서 0.1로 선형적으로 조정하고 그 이후 프레임에서는 0.1로 고정된 학습률 사용.

frame skipping 기술: k번째 프레임만 보고 액션 선택. 그 외의 프레임에서는 가장 최근의 액션 선택. 연산량 감소 효과가 있음. space invader 게임에서는 k=4일 때 레이저가 보이지 않아 k=3으로 레이저가 보이게 설정함.

5.1 Training and Stability



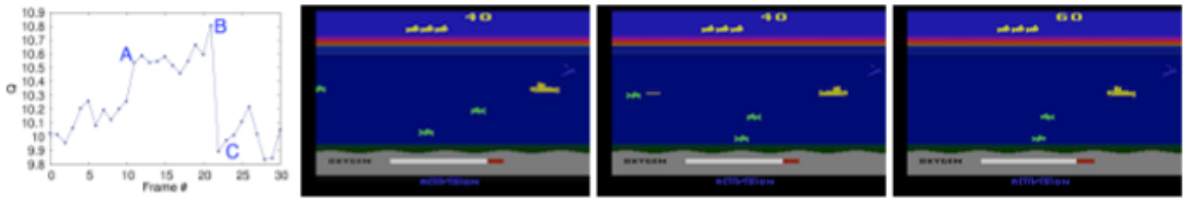
10000 단계동안 학습률 0.05로 적용.

1 에포크는 50000 미니배치의 가중치 업데이트 혹은 30분 정도의 학습 시간

학습이 진행될수록 보상과 액션 값이 증가함.

보상은 노이즈가 많아서 학습 알고리즘을 파악하기 어려운데 Q는 부드럽게 증가하는 것을 확인할 수 있다.

5.2 Visualizing the Value Function



Sequest 게임에서 학습된 값 함수 시각화 .적이 나타났을 때 예측해서 점프하는 것. 적이 없을 때 땅으로 내려옴.

5.3 Main Evaluation

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Sara로 라벨된 사라 알고리즘에서는 선형 모델 적용

Contingency는 독립된 채널의 128개의 색깔에서 학습

HNeat Best 점수는 손수 엔지니어된 객체 감지 알고리즘 사용

DQN이 인간보다 breakout 게임에서 더 좋은 결과를 보임.

6. Conclusion

이 논문에서 강화학습을 위한 새로운 딥러닝 모델 소개함. 인풋으로 raw 픽셀을 사용하며 Atari 2600 게임에서 테스트함. Experiece replay memori와 확률적 미니배치 방식을 사용한 병형된 online Q-learning 방식 보여줌. 별도의 하이퍼 파라미터 조정 없이도 좋은 성능을 보임.