2018년도

학사학위논문

김택윤

이상수

Artificial intelligence related technology

and Game implementation with

reinforcement learning

인공지능 관련 기술과

강화학습을 적용한 게임 구현

순천향대학교 공과대학

컴퓨터공학과

2018년 11월 13일

인공지능 관련 기술과

강화학습을 적용한 게임 구현

Artificial intelligence related technology

and Game implementation with

reinforcement learning

순천향대학교 공과대학

컴퓨터공학과

이 논문을 공학사학위 논문으로 제출함

지도교수 이 해 각

2018년 11월 13일

김택윤

이상수

김 택 윤

이 상 수 의 공학사학위논문을 인준함

2018년 11월 13일

순천향대학교 공과대학

컴퓨터공학과

심 사 위 원 남 윤 영 인

**초 록**

최근 IT 기술을 활용해 언제 어디서나 사람이 할 수 있는 일 또는 할 수 없는 일을 대체하여 컴퓨터가 대신 해주는 서비스가 트렌드로 떠오르고 있다. 이것을 인공지능 산업이라고 부르고 지금도 개발이 진행되고 있다. 대표적인 예로 딥 마인드에서는 사람이 아닌 컴퓨터가 바둑을 둘 수 있는 AI를 개발했다.

필자가 처음 이 분야에 관심을 가졌던 계기는 “알파고”이다. 알파고는 스스로 생각하여 사람과 바둑 경기를 할 수 있게 만들어진 인공지능이다.

본 논문에서는 이것을 게임에 적용하여 사람만큼 혹은 사람보다 게임을 잘 할 수 있는 AI개발을 제안한다. 제안한 AI는 딥러닝을 통하여 컴퓨터가 스스로 훈련할 수 있도록 구성되었다. 딥러닝의 여러 가지 기법 중 몇 가지의 기법이 소개되고 그 알고리즘을 게임에 적용하게 되는 방법으로 한다. 기존의 게임과 직접 개발한 게임들을 통해서 적용함으로써 훈련되어가는 과정을 보여준다.

**ABSTRACT**

By using a recent IT technology, services that a computer does instead works that person can or can't do at any time are emerging as a trend. We called this 'artificial intelligence industry' and development is still under way. For example, Deep Mind developed AI that can play 'Badduk' to a computer instead people. "Alphago" is a reason that writer is interested in this field at first. Alphago is an artificial intelligence that was made for playing 'Badduk' with a person thinking itself.

In this paper, it suggests Al development that can apply this to games and play games as well as humans. Suggested AI was consist of training itself through using deep learning. Some techniques of various techniques about deep learning was introduced and we used a method that a algorithm was applyed in games. By applying through games we made and traditional games, we show a process that it was training.

**차 례**

**제 1 장 서 론**1

**제 2 장 관련 기술 분석**3

2.1 선행연구3

2.1.1 인공지능3

2.1.2 머신러닝4

2.1.3 인공신경망5

2.1.4 딥러닝6

2.2 관련 개발 도구6

2.2.1 텐서플로우 (TensorFlow)6

2.2.2 파이썬 (Python)7

2.2.3 케라스 (KERAS )7

2.2.4 open-AI gym 7

2.3 필요기술8

2.3.1 강화학습8

2.3.1.1 Q-Learing9

2.3.2 CNN10

2.3.2.1 CNN의 작동 방식13

**제 3 장 AI 게임 구현13**

3.1 소개 및 설명 13

3.1.1 AI 게임 제작 환경13

3.1.2 장애물 피하기 게임15

3.1.2.1 장애물 피하기 게임 소개15

3.1.2.2 장애물 피하기 게임 설명16

3.1.3 과일 받기 게임19

3.1.3.1 과일 받기 게임 소개19

3.1.3.2 과일 받기 게임 설명19

**제 4 장 실행 및 결과 분석26**

4.1 실험 도구26

4.2 실험 방식27

4.3 실험 결과28

4.3.1 장애물 피하기 게임 훈련과정 몇 결과28

4.3.1.1 장애물 피하기 게임 훈련 과정28

4.3.1.2 장애물 피하기 게임 훈련 수치 결과29

4.3.1.3 장애물 피하기 게임 결과30

4.3.2 과일 받기 게임 훈련과정 몇 결과31

4.3.2.1 과일 받기 게임 훈련 과정 및 수치31

4.3.2.2 과일 받기 게임 훈련 결과32

**제 5 결론 및 향후 과제33**

**참고문헌35**

**감사의 글36**

**그 림 차 례**

[그림 1] 인공지능 상관 관계 3

[그림 2] 머신러닝 종류 4

[그림 3] 비지도 학습 예시 5

[그림 4] open-AI gym에서 제공하는 게임 환경 8

[그림 5] 에이전트와 환경의 상호작용 9

[그림 6] Q-Learning 수식 10

[그림 7] CNN의 작동 방식 11

[그림 8] CNN의 각 계층 12

[그림 9] Python 3.5 version 13

[그림 10] 환경변수 설정 14

[그림 11] Jupyter(Jupyter notebook) 14

[그림 12] Q 함수의 수학적 표기 15

[그림 13] DQN의 학습 방법 16

[그림 14] DQN의 train1 17

[그림 15] DQN의 train2 18

[그림 16] 입력 레이어와 히든레이어, 출력 레이어 코드 20

[그림 17] cost를 최소화 하기 위한 경사하강법 이용 20

[그림 18] 경사하강법의 원리 21

[그림 19] 클래스 catchEnviroment 내용 코드 - 1 22

[그림 20] 클래스 CatchEnvironment 내용 코드 -2 23

[그림 21] 클래스 CatchEnvironment 내용 코드 -3 24

[그림 22] 클래스 ReplayMeory 내용 코드 24

[그림 23] 훈련의 초기 방법 코드 25

[그림 24] Jupyter notebook 실행 cmd창 26

[그림 25] 장애물 피하기 게임 수치 28

[그림 26] 텐서보드로 나태낸 평균 reward 29

[그림 27] 훈련의 데이터를 게임에 적용 30

[그림 28] 과일 받기 게임 훈련 수치 - 1 31

[그림 29] 과일 받기 게임 훈련 수치 – 1 그래프 32

[그림 30] 훈련된 모델 과일 받기 게임에 적용 32

**제 1 장 서 론**

최근 IT 기술을 활용해 언제 어디서나 사람이 할 수 있는 일 또는 할 수 없는 일을 대체하여 컴퓨터가 대신 해주는 서비스가 트렌드로 떠오르고 있다. 이것을 인공지능 산업이라고 부르고 지금도 개발이 진행되고 있다. 딥 러닝의 등장으로 인해 머신 러닝의 실용성을 강화됐고, 인공지능의 영역은 확장됐다. 여러 분야의 딥 러닝 기반의 기술들은 우리 일상에서 이미 사용되고 있거나, 실용화를 앞두고 있다. 딥 러닝은 AI를 실현할 수 있는 인공 지능의 현재이자 미래로 평가 받고 있는 기술로 본 논문에서는 AI를 게임환경에서 다루어 보고자 한다.

따라서 본 논문에서는 이와 같은 인공지능 시스템을 적용하여 사람만큼 혹은 사람보다 게임을 잘 할 수 있는 AI를 제안한다. 제안한 AI는 인공지능의 기법중 하나인 딥 러닝을 적용하여 훈련된 모델을 생성할 수 있도록 한다. 훈련과정에서 딥 러닝의 인공 신경망 즉, 뉴런들의 데이터 전달 과정을 거치면서 완벽에 가까운 AI 개발을 결과물로 한다. 또한 몇 개의 게임을 Q-Learning과 RNN기법을 사용하여 적용하고 훈련이 끝난 AI는 사람과 같은 사고를 갖춤으로써 스스로 생각하고 해당 게임의 스테이지를 한 단계씩 풀어가는 과정을 보여준다..

본 논문의 구성은 다음과 같다. 2장에서는 관련 기술에 대한 선행 지식과 개발 도구에 대해서 설명하고 필요한 기술에 대해서 기술한다. 3장에서는 AI 모델의 설계 및 제작에 대해서 설명한다. 4장에서는 AI모델의 실행 및 결과분석 에 대해 설명하면서 테스트를 진행한다. 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

**제 2 장 관련 기술 분석**

2.1 선행연구

2.1.1 인공지능

인공 지능(人工知能)이라는 개념은 인간의 지능을 기계로 구현하는 것으로 1956년 미국 다트머스 대학에 있던 존 매카시 교수가 개최한 다트머스 회의에서 처음 등장했다고 알려져 있지만 인공지능이라는 개념 자체는 앨런 튜링이 ‘생각하는 기계’의 구현 가능성과 튜링 테스트를 제안한 것은 1950년의 일이며, 최초의 신경망 모델은 1943년에 제안되었다. 또한, 2015년 이후 신속하고 강력한 병렬 처리 성능을 제공하는 GPU의 도입으로 더욱 가속화되고 있다.

인공지능 분야에는 몇 가지 기법이 있는데 기계학습, 인공신경망, 딥 러닝이 몇

가지 예이다.[1]

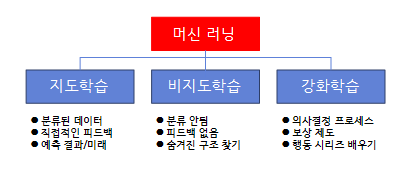
[그림 1] 인공지능 상관 관계



2.1.2 머신러닝

인공지능에서 기계학습(機械學習) 또는 머신 러닝은 컴퓨터가 훈련을 가능하게 하는 알고리즘과 기법을 개발하는 분야로써 의사 결정 기준에 대한 구체적인 지침을 소프트웨어에 직접 적용하는 것이 아닌, 대량의 데이터와 알고리즘을 통해서 도출되는 결론 그 자체를 ‘학습’ 시켜 작업 수행 방법을 익히는 것을 목표로 하는 것이다. 다양한 기계 학습의 응용이 존재한다. 기계학습 알고리즘은 지도 학습(supervised)과 비지도 학습(unsupervised learning)의 두 가지 주요 범주 중 하나로 분류 될 수 있다. 지도 학습이란 문제에 대한 답을 알고 있을 때 반대로 로직을 파악할 수 있는 것을 의미하며 비지도 학습이란 지도학습과 다르게 정확한 데이터가 없을 경우에 사용됩니다.

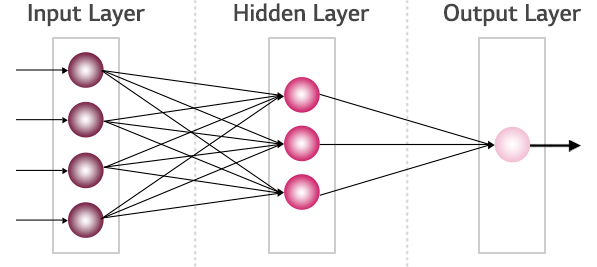
[그림 2] 머신러닝 종류



2.1.3 인공신경망

인공신경망((人工神經網,)은 기계학습과 인지과학에서 생물학의 신경망의 영감을 얻은 통계학적 학습 알고리즘이다. 시냅스의 결합으로 네트워크를 형성한 인공 뉴런이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제해결능력을 가지는 비선형 모델이다. 일반적으로 사용되는 기본적인 인공신경망 알고리즘인 다층인공신경망의 경우 입력층, 은닉층, 출력층 이렇게 세가지 층으로 구분됩니다. 또한 각 층들은 노드들로 구성되어 있다. 인공신경망의 학습 방법은 회귀 분석/지도 학습의 방법론에 입각해서 ‘정답’들과의 오차를 최소화하는 가중치와 역치를 찾는 최적화 문제를 푸는 것이다. 이러한 인공신경망은 커다란 문제점인 과적합 문제가 발생하는데 최근 알파고가 도입하고 있는 딥러닝 이라는 알고리즘이 인공신경망의 과적합 문제를 해결할 수 있는 강력한 대한으로 떠오르고 있습니다.[2]

[그림 3] 비지도 학습 예시



2.1.4 딥러닝

딥러닝(Deep Learning)이란 아직까지 불완전한 기계학습을 보다 더 완전하게 실현하는 기술이다. 딥러닝이라는 단어는 본래 아는 사람만 아는 단어였는데, 국내에 바둑 두는 기계인 ‘알파고’가 대한민국의 이세돌 9단을 바둑으로 압승하면서 널리 알려지게 되었습니다. 딥러닝은 머신러닝의 세부 방법론들을 통칭하는 개념입니다. 즉, 딥러닝은 머신러닝의 ‘부분집합’이라고 정의할 수 있습니다.

2.2 관련 개발 도구

본 장에서는 인공지능 프로그램에 관련 개발 도구인 텐서플로우, 파이썬, 케라스, open-AI gym을 소개 한다.

2.2.1 텐서플로우(TensorFlow)

본 논문에서는 구글이 개발 하여 오픈 소스로 공개한 기계학습 라이브러리로 딥러닝과 기계학습 분야를 일반인들도 사용하기 쉽도록 다양한 기능을 제공하고 있다. 기계학습의 한 분야인 강화 학습을 게임에 적용 하기 위하여 본 논문 에서는 텐서플로우(TensorFlow) 라이브러리를 사용하였다.

2.2.2 파이썬 (Python)

파이썬은 귀도 반 로섬(Guido van Rossum)이 창시한 인터프리터 방식의 프로그래밍 언어이다. 본 논문에서는 기계학습 라이브러리인 텐서플로우 (TensorFlow)의 관련 자료가 가장 많은 파이썬을 사용하였다.

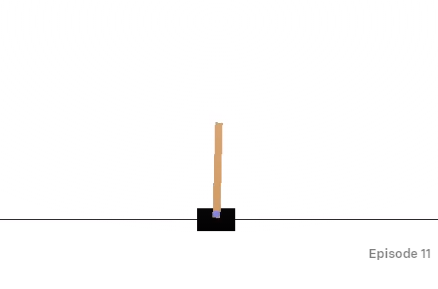
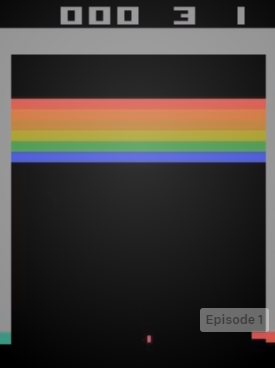
2.2.3 케라스 (Keras)

케라스(Keras)는 파이썬으로 작성된 오픈 소스 신경망 라이브러리이다. MXNet, Deeplearning4j, 텐서플로, Microsoft Cognitive Toolkit 또는 Theano 위에서 수행할 수 있다. [3]

2.2.3 open-AI gym

open-AI에서 만든 gym은 강화 학습 알고리즘을 개발하고 비교 하기 위한 툴킷(tooklit)이다. 에이전트의 구조에 대한 가정을 하지 않으며 텐서플로우(TensorFlow) 또는 테아노(Theano)와 같은 모든 수치 계산 라이브러리와 호환 된다.[4]

[그림 4] open-AI gym에서 제공하는 게임 환경 (Breakout, CartPole)



2.3 필요 기술

2.3.1 강화학습

강화학습(Reinforcement Learning)은 기계학습이 다루는 문제들 중 하나로 어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 혹은 순서를 선택하는 방법이다.

[그림 5]에이전트와 환경의 상호작용

에이 전트

보상

다음 상태

다음상태

행동

환경

2.3.1.1 Q-Learning

Q-Learning은 모델 없이 학습하는 강화 학습 기법 가운데 하나이다. Q-Learning은 주어진 MDP(유한 마르코프 결정 과정)의 최적의 정책을 찾기 위해 사용할 수 있다. Q-Learning은 주어진 상태에서 주어진 행동을 수행하는 것이

가져다 줄 효용의 기대값을 예측하는 함수인 Q 함수를 학습함으로써 최적의 정책을 학습한다.

수식에서 사용되는 변수에 대하여 설명한다.

[그림6]Q-Learning 수식 

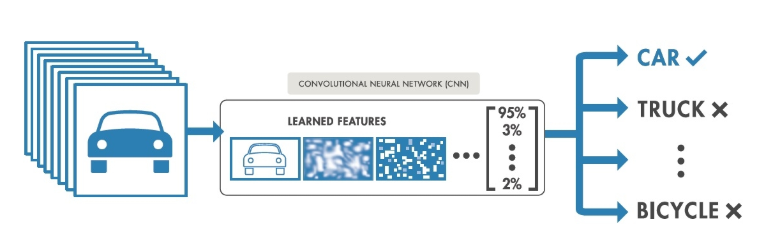
|  |  |
| --- | --- |
| s 현재 state | 다음 state |
| 다음 state에 대한 reward | a 현재 action |
| Learning rate | Discount factor |
| Q 어떤 state에 대하여 최상의 reward 값을 도출하는 action을 맵핑해 놓은 테이블 | |

2.3.3 CNN

컨벌루션 뉴럴 네트워크(CNN 또는 ConvNet)는 모델이 직접 이미지, 비디오, 텍스트 또는 사운드를 분류하는 머신 러닝의 한 유형인 딥러닝에 가장 많이 사용되는 알고리즘이다. [5]

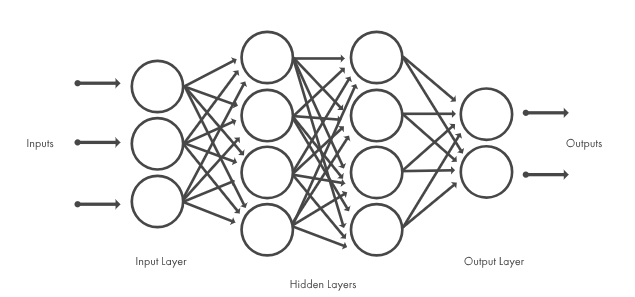
CNN은 이미지에서 객체, 얼굴, 장면을 인식하기 위해 패턴을 찾는데 유용하다. CNN은 데이터에서 직접 학습하며, 패턴을 사용하여 이미지를 분류하고 특징을 수동으로 추출할 필요가 없다.

[그림 7]CNN의 작동 방식



2.3.3.1 CNN의 작동 방식

CNN를 구성하는 수십, 수백개의 계층이 각각 이미지의 서로 다른 특징을 감지하도록 학습할 수 있다. 필터는 각 학습 이미지에 서로 다른 해상도로 적용되고, 필터의 출력은 다음 계층의 입력으로 활용된다. 필터는 밝기 및 가장자리 등과 같이 매우 단순한 특징에서 시작하여 객체만의 고유한 특징으로 더 복잡하게 발전할 수 있다.

[그림 8] CNN의 각 계층

|  |  |
| --- | --- |
| 컨벌루션 | 각 이미지에서 특정 특징을 활성화하는 컨벌루션 필터 집합에 입력 이미지를 통과 시킨다. |
| ReLU | 음수 값을 0에 매핑하고 양수 값을 유지하여 더 빠르고 효과적인 학습을 가능하게 한다. 이때 활성화된 특징만 다음 계층으로 전달 되기 때문에 이 과정을 활성화라 부른다 |
| 풀링 | 비선형 다운샘플링을 수행하고 네트워크에서 학습해야 하는 매개 변수 수를 줄여서 출력을 간소화합니다. |

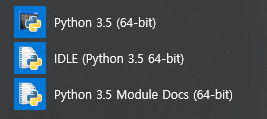
**제 3 장 AI 게임 구현**

3.1 소개 및 설명

3.1.1 AI 게임 제작 환경

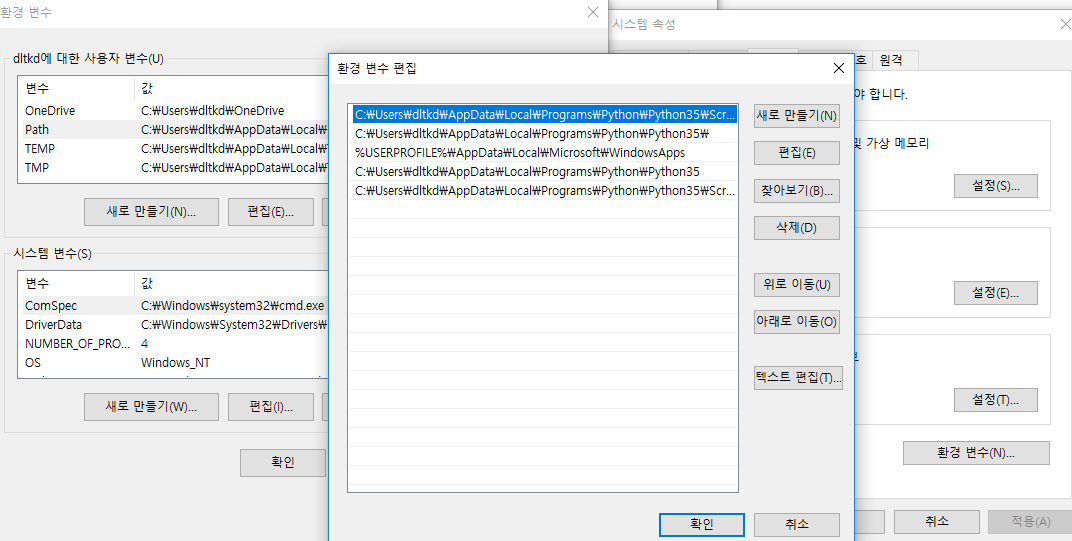
일단 가장 먼저 해야 할 일은 파이썬(Python)을 설치하는 것이다.. 윈도우 환경에서는 TensorFlow가 Python 3.5 이상부터 동작하기 때문에 반드시 Python 3.5 이상의 버전을 설치 해야한다.

[그림 9] Python 3.5 version



TensorFlow를 설치하기 위해서는 윈도우 cmd창에서 pip명령어로 install 해야한다. 그러려면 환경변수 즉, PATH를 지정해 줘야한다.

[그림 10] 환경변수 설정



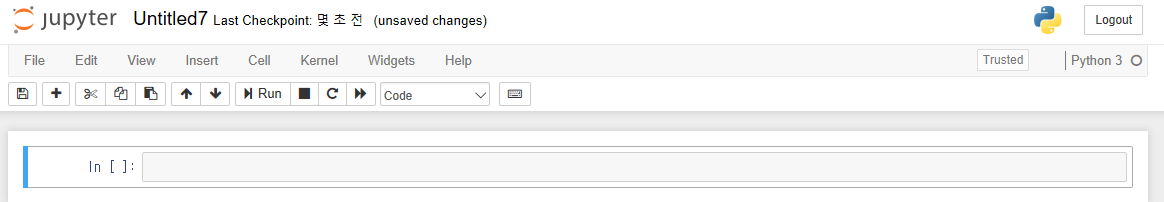
**3**

**2**

**1**

여기에서 PATH는 자신의 PC의 Python이 설치되어 있는 경로를 설정하고 또 하나는 Scripts폴더를 하나 더 추가하면 PATH설정은 완료가 된다. Pip 명령어를 사용하여 tensorflow, numpy, matplotlib, Jupyter를 설치한다. 여기서 tensorflowms ‘1.5.0’ version, numpy는 ‘1.15.4’ version, matplotlib는 ‘3.0.2’ version을 사용하였다. Python 자체의 환경에서 구현한 게임 하나와 Jupyter에서 구현한 게임을 구성으로 한다.

[그림 11] Jupyter(Jupyter notebook)



3.1.2 장애물 피하기 게임

3.1.2.1 장애물 피하기 게임 소개

기존에 있는 장애물 피하기 게임은 키보드로 조작하여 랜덤으로 떨어지는 장애물을 피하는 게임이다. 이 게임과의 차이는 사용자가 직접 키보드로 조작하지 않고 훈련된 AI가 스스로 과일을 받아 내기 위해서 강화 학습 기법을 활용하였다. 여기서 Q-learning 알고리즘을 적용하였고 본 논문에서는 Q-learning에서 현재 상태에서 발생하는 최대의 보상(MAX reward)을 획득하는 최적의 행동 정책을 학습한다. 에이전트는 게임 환경에서 랜덤 하게 행동을 수행하게 되고 이 행동들은 결과로 리워드를 얻는 방법이다. 이런 행동들을 선택하는 방법을 지침이라고 하고 다음 상태가 무작위적으로 발생하게 된다. 결국 주어진 행동을 수행하는 것이 가져다 줄 효용의 기대값을 예측하는 함수인 Q함수를 학습함으로써 최적의 정책을 유도한다.

학습이 끝난 AI는 사용자의 키 입력 없이 학습 데이터 기반으로 움직이며 랜덤하게 떨어지는 장애물을 피하는 결과가 나타나게 된다.

[그림 12] Q 함수의 수학적 표기

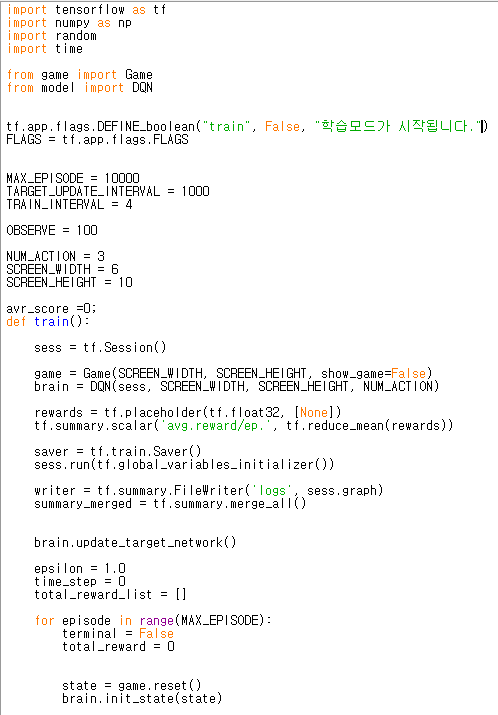


3.1.2.2 장애물 피하기 게임 설명

게임 구현을 위한 핵심 내용은 Q\_value 를 이용해 얻어온 액션을 수행하고, 해당 액션에 의한 게임의 상태화 리워드를 획득한 뒤 이것을 메모리에 순차적으로 쌓아둔다. 일정 수준 이상의 메모리가 쌓이면 메모리에 저장된 것들 중 샘플링함수를 이용해 최적화를 수행한다. Q\_value로 예측한 액션으로 상태를 만들어내고, target\_Q\_value와 비교하는 형식을 취한다.

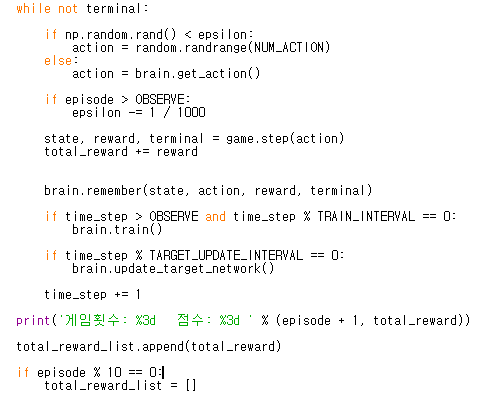
[그림 13] DQN의 학습 방법

|  |  |
| --- | --- |
| 1. | 상태 를 입력 시, Q()가 최대가 되는 행동 를 선택, 단 확률 으로 랜덤하게 선택 |
| 2. | 행동 에 의해 상태 과 보수 를 얻는다. |
| 3. | 상태 를 입력하고 Q()를 구한다. |
| 4. | 다음을 정답으로 해서 변수 를 갱신한다.  +YQ()  -  L()= |

[그림 14] DQN의 train1

장해물 피하기 게임의 에이전트는 최대한 많은 훈련을 해야 한다. 그렇게 됨으로써 좀더 완벽한 훈련 데이터가 쌓이기 때문이다. UI부분이 실행되기 전에 먼저 DQN의 train 함수 코드를 실행시켜서 먼저 학습을 시킨다.

DQN의 최대 학습 횟수를 10000으로 전역으로 설정합니다. 자동차의 action : 0:좌, 1 : 유지, 2 : 우; 이렇게 설정한다. Def train():은 학습 함수로 세션을 생성하고 DQN 모델링 한 클래스를 brain에 선언합니다. 리워드 placeholder로 빈 노드 생성후 다음에 취할 액션을 DQN 을 이용해 결정할 시기를 결정한다. for문에서 최대 10000번까지 실행한다.

[그림 15] DQN의 train2

처음에는 학습된 데이터가 없기 때문에 0~1까지의 랜덤 한 값을 주어 1보다 작은 값이 나오게 되면 액션을 취하도록 하였다. 점점 랜덤 하게 플레이를 하는 횟수를 줄이고 어느 순간 학습을 시작 하도록 하게 한다. 결정한 액션을 이용해 게임을 진행하고, 보상과 게임의 종료 여부를 받아온다. 현재 상태를 Brain에 기억시키고 기억한 상태를 이용해 학습한다. 다음 상태에서 취할 행동을 결정하고 DQN으로 학습을 진행한다.

즉 agent.py에서 학습을 진해하고 game.py에서 훈련 데이터를 기반으로 해당 게임이 진행 된다.

3.1.2 과일 받기 게임

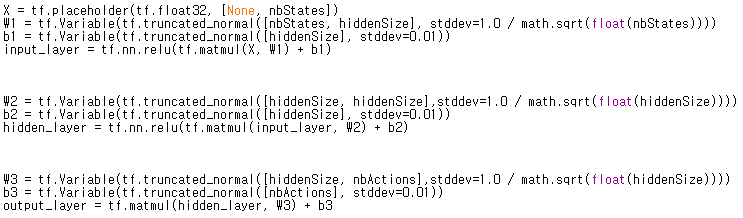
3.1.2.1 과일 받기 게임 소개

위의 게임인 장애물 피하기 게임과는 다르게 위에서 떨어지는 과일을 받아면 점수가 오르는 게임 이다. 이 게임은 간단한 인공 신경망 모델을 이용하고 최적의 Q함수 값을 이용하여 성능을 높여가게 된다.

3.1.2.2 과일 받기 게임 설명

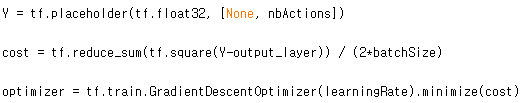
게임 구현을 하기 위해서 모델을 구성 한다. 모델은 인공 뉴런을 연결 하여 만들어진 인공 신경망을 이용을 한다. 인공 뉴런은 입력 값 X에 가중치(W)를 곱하고 편향(b)을 더한 뒤 활성화 함수(Sigmoid, ReLU 등)를 거쳐 결괏값 y를 만들어 낸다. 그리고 원하는 y 값을 만들어 내가 위해서는 가중치(W)와 편향(b)의 값을 변경해 가면서 적절 한 값을 찾아 내게 된다.

[그림 16]입력 레이어와 히든레이어, 출력 레이어 코드



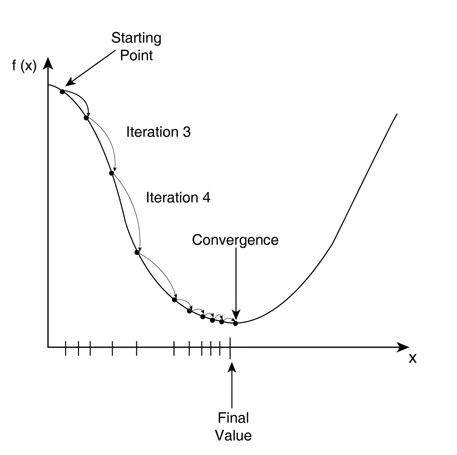
입력, 히든, 출력 레이어를 만든 후 cost값을 최소화 하기 위하여 경사하강법인 GradientDescentOptimizer를 이용한다.

[그림 17] cost를 최소화 하기 위한 경사하강법 이용



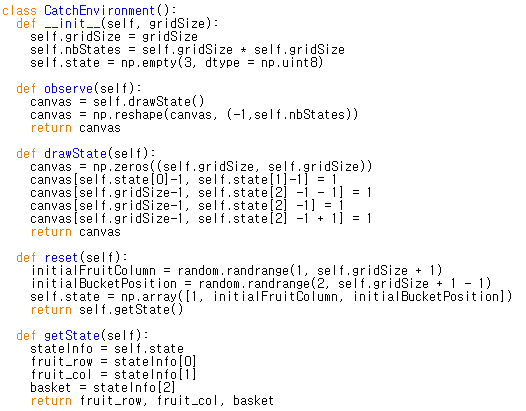
경사하강법을 이용하여 비용함수의 최소값 위치를 찾기 위해 비용함수(Cost Function)의 그레디언트 반대 방향으로 정의한 step size를 가지고 조금씩 움직여 가면서 최적의 파라미터를 찾는 방법이다. 이때 그레디언트는 파라미터에 대해 편미분한 벡터를 의미한다.

[그림 18] 경사하강법의 원리



게임의 직접 만들어 사용 하기 때문에 게임환경을 클래스 CatchEnviroment로 생성 하였다. \_\_init\_\_은 게임의 크기 상태를 초기화 한다. observe는 화면의 정보를 반환 한다. drawState는 위에서 떨어지는 과일과 과일을 받는 바구니를 표시하여 화면정보를 반환한다. reset는 과일과 바구니의 상태의 위치를 초기화 한다. getState는 과일과 바구니의 상태를 리턴한다.

\_\_init\_\_은 게임의 기본 틀 크기, 상태를 초기화 하는 함수

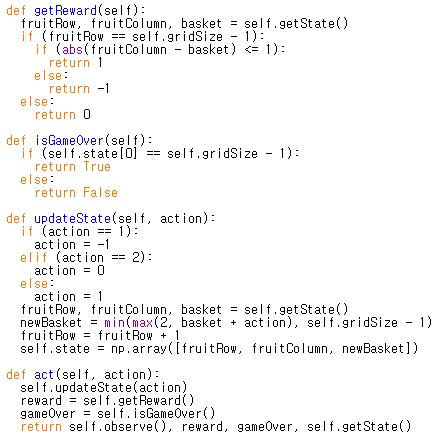
[그림 19] 클래스 catchEnviroment 내용 코드 - 1

getReward는 떨어지는 과일을 바구니가 받았을때 얻는 보상값을 반환 한다. 이때 과일이 바닥-1에 닿았을때 과일의 x값과 바구니의 x값이 같을때 보상값이 얻어 지게 된다. isGameOver는 바구니가 과일을 받는지 못받는지 검사하여 값을 반환 한다. updateState는 과일과 바구니의 상태를 업데이트 한다 action이 -1이면 왼쪽으로 0이면 대기 1이면 오른쪽으로 바구니가 움직이게 된다.

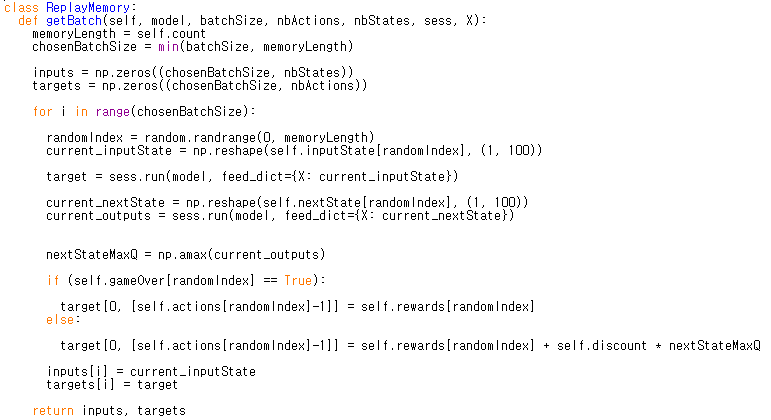
[그림 20]클래스 CatchEnvironment 내용 코드 -2



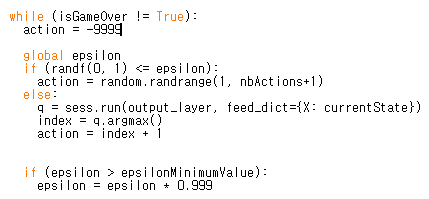
마지막으로 상황에 맞춰 움직인 후 과일을 받았다면 보상값을 얻고 과일을 받지 못했다면 게임오버에서 반환된 값을 가지고 마지막에 각각의 값을 반환 한다.

[그림 21]클래스 CatchEnvironment 내용 코드 - 3

클래스 ReplayMemory는 게임의 내용을 배치로 묶어서 리턴하게 된다 이때 메모리에 배치단위로 저장된 게임중 한개를 랜덤하게 선택 하여 Q함수 값을 구하여 최적의 값을 반환 하게된다.

[그림 22]클래스 ReplayMeory 내용 코드

위의 두개의 클래스를 이용하여 메인 함수에서 훈련 모델을 만들게 된다. 훈련 초기에는 훈련된 데이터가 없기 때문에 무작위로 바구니를 움직여 데이터를 수집 하게 된다. 무작위로 훈련이 되면서 무작위로 되는 확률을 줄이게 되어 결국에는 훈련된 데이터를 사용하게 된다.

[그림 23] 훈련의 초기 방법 코드

위의 코드로 훈련이 된 데이터는 model.ckpt로 저장되어 게임에서 훈련된 데이터를 이용하여 작동하게 된다.

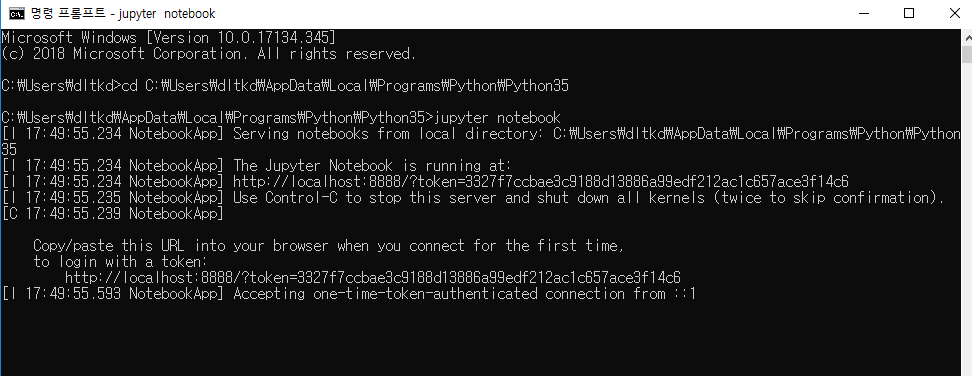
**제 4 장 실행 및 결과 분석**

4.1 실험 도구

장애물 피하기 게임과 과일 받기 게임은 인터프리터 언어인 파이썬 3.5 version으로 개발하였다. 먼저 장애물 피하기 게임은 cmd command 창에서 해당 모델을 훈련 시키고 .py 파일을 실행 시켜서 게임 UI가 나타나도록 하였다.

두 번째로 과일 받기 게임은 실행 환경을 Jupyter notebook을 사용한다. Jupyter로 실행하기 위해서 기존에 .py파일을 .ipynb 확장자로 변경 해야 한다. Jupyter를 사용한 이유는 python과 다르게 한 줄씩 컴파일 해주기 때문에 좀 더 편리한 개발 환경을 제공하기 때문이다.

[그림 24] Jupyter notebook 실행 cmd창



4.2 실험 방식

장애물 피하기 게임이 그저 운이 좋게 장애물을 피하는 것인지 실제로 딥러닝의 Q-learning을 적용한 모델 토대로 움직이는 것인지 실험을 통해서 알아볼 예정이다. 장애물 피하기 게임은 훈련이 되는 과정을 cmd창으로 띄워서 보여주고 훈련된 AI가 해당 게임을 사람의 손을 거치지 않고 얼마나 많은 스테이지를 클리어 하는지 Python의 UI 기능을 통해서 결과를 확인한다.

또 하나의 과일 받기 게임도 장애물 피하기 게임과 동일한 방식으로 실험을 진행 하지만 Jupyter notebook을 활용한다. 마찬가지로 Jupyter를 통해서 과일 받기 게임의 학습되는 과정을 확인하고 Python 라이브러리 즉,Jupyter의 matplotlib를 토해서 게임UI를 확인한다.

4.3 실험 결과

4.3.1 장애물 피하기 게임 훈련과정 및 결과

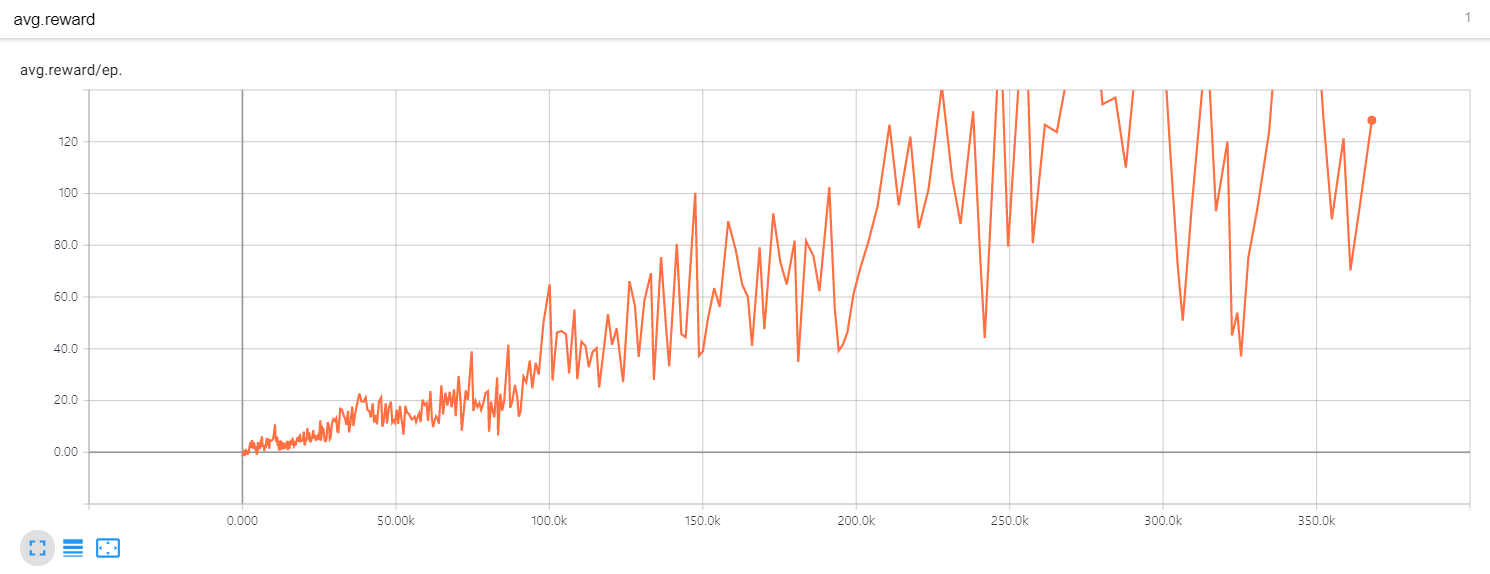
4.3.1.1 장애물 피하기 게임 훈련 과정

장애물 피하기 게임의 훈련 과정을 cmd창을 통해서 확인하고 게임 횟수가 늘어남에 따라 점수가 점점 올라가는 것을 확인할 수 있다. 모델의 학습 횟수는 최대 10000번으로 제한시켰고 제한된 학습 횟수 안에 agent가 획득한 최대 점수를 기반된 모델로 게임이 진행된다. Agent를 훈련 시키는 방법은 command창에 pyhon agent.py –train 명령어를 입력하면 아래와 같은 방법으로 훈련을 하게 된다.

[그림 25] 장애물 피하기 게임 수치

4.3.1.2 장애물 피하기 게임 훈련 수치 결과

장애물피하기 게임의 agent가 딥러닝 기반으로 실행되는지 확인하기 위해서 tensorflow에서 제공하는 tensorboard를 활용해서 그래프로 나타내었다. Agent가 게임의 reward값을 최대로 하여 최적의 길을 찾아가고 해당 로그가 C:\Users\dltkd\AppData\Local\Programs\Python\Python35\logs 폴더에 적재되도록 하였다. 그 결과 처음에는 랜덤 하게 게임이 실행되지만 시간이 지남에 따라서 성공률이 높아지고 그래프에서 확인할 수 있듯이 평균 reward가 점점 높아지는 것을 확인할 수 있었다.

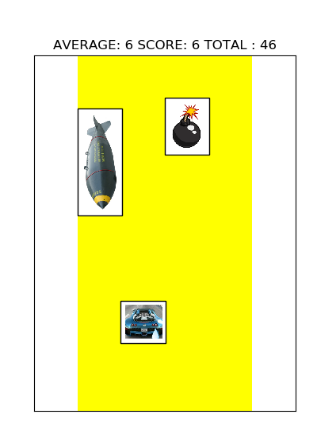
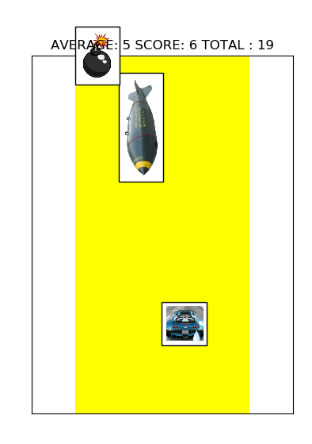
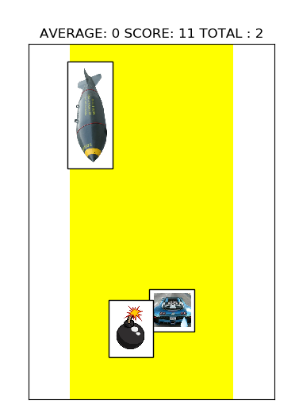
[그림 26] 텐서보드로 나태낸 평균 reward

4.3.1.3 장애물 피하기 게임 결과

Agent.py는 게임을 진행하거나 학습시키는 에이전트이고 game.py는 게임을 구현해 놓은 파일로 게임의 상태를 화면의 픽셀로 가져오지 않고, 좌표값을 이용하여 계산량을 최대한 줄이도록 하였다. Model.py는 DQN을 구현해 놓은 파일입니다.

장애물 피하기 게임의 agent안에 DQN을 적용한 결과 화면이다. 실행 실간이 지날수록 AVERAGE와 SCORE가 높아지는 것을 확인할 수 있으므로 DQN이 agent에 적용이 잘 되었음을 알 수 있다.

[그림 27] 훈련된 데이터를 게임에 적용

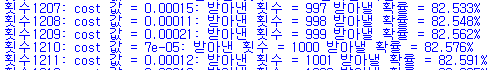
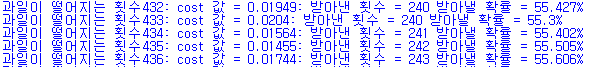
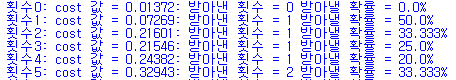


4.3.2 과일 받기 게임 훈련과정 및 결과

4.3.2.1 과일 받기 게임 훈련과정 및 수치

위에서 설계한 모델을 실행 하여 훈련 하였다. 훈련 횟수는 총 1800번으로 제한 시켰고 cost의 값이 소수점을 나타내고 훈련을 거듭 할 수록 cost의 값이 0에 수렴 하게 된다.

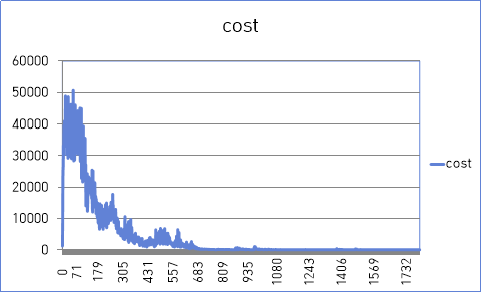
[그림 28] 과일 받기 게임 훈련 수치 - 1



위에서 나온 cost의 값이 소수점 이기 때문에 그래프로 나타내기 위해서

cost \*1000을 하여 결과 확인을 좀더 수월하게 할 수 있도록 하였습니다.

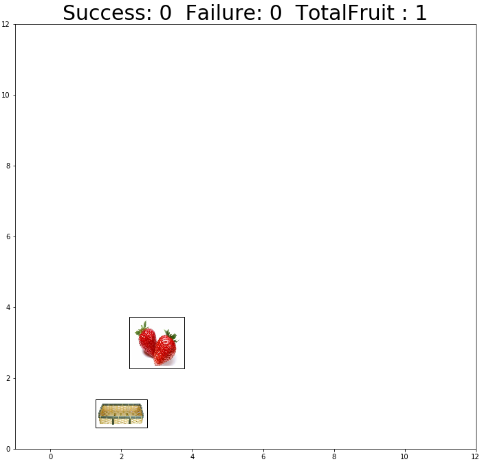
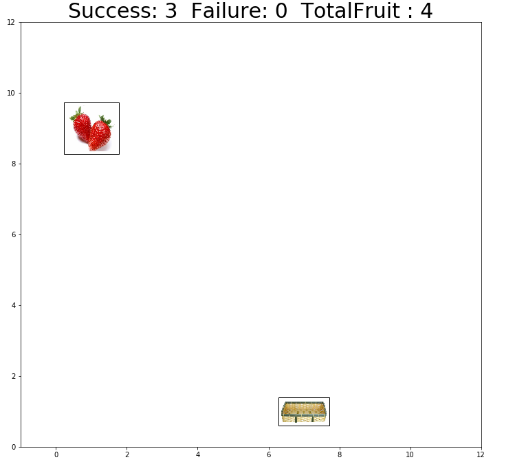
[그림 29] 과일 받기 게임 훈련 수치 - 2 그래프



4.3.2.2 과일 받기 게임 게임 결과

두개의 파일 중 TrainCatch.py는 모델을 훈련 한 뒤 학습 데이터를 저장한다. 여기서 저장된 학습 데이터를 가지고 게임을 하는 PlayCatch.py를 실행한 결과 학습된 모델의 적용이 잘 되었음을 알 수 있습니다.

[그림 30] 훈련된 모델 과일 받기 게임에 적용



**제 5 장 결론 및 향후 과제**

딥러닝 기술은 컴퓨터에게 데이터를 학습시켜 마치 사람처럼 스스로 의사결정을 할 수 있게 한다. 최근에 뛰어난 성능을 보이며 많은 관심을 받고 있는 딥러닝은 알파고 등장이 우리에게 큰 충격 이였다. 인공 신경망은 생물학의 신경망에서 영감을 얻은 수학적 모델로 시냅스의 결합으로 네트워크를 형성한 인공 뉴런이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제 해결 능력을 가진다. 이러한 이론을 몇 가지의 게임에 적용해 보았는데 학습이 진행 되면서 Q\_value를 이용해 얻어온 Action을 수행하고, 해당 Action에 의한 게임의 상태와 Reward를 획득하고 이것을 Memory에 순차적으로 쌓는 것을 확인하였다. 어느 정도의 이상의 Memory가 쌓이게 되면, 적재 된 것들 중 Sample을 하여 위에서 언급한 수식을 이용해 최적화를 진행하게 된다. Q\_value로 예측한 Action 상태를 만들어내고, 목표 value값과 비교를 하면서 학습을 진행하게 된다. 학습 시간이 오래되면 오래될수록 배운 것이 많아지게 되면서 학습 시간이 길어지게 된다. 여기서 CPU를 사용하는가 GPU를 사용하는가는 엄청난 시간 차이를 보였다. 병렬처리를 사용하는 GPU의 실행속도가 눈에 보일 정도로 차이를 보였고 게임 결과는 CPU와 GPU중 어느 것을 사용해도 동일하였다.

본 논문에서는 Q\_learning만 사용하여 인공지능 게임을 개발하였다. 하지만 현재 이미지 처리에 뛰어난 효과를 발휘하는 합성곱 신경망인 CNN을 게임에 적용 하게 된다면 Q-learning만을 사용한 게임보다 성능이 좋아 질것이다. 추후 CNN을 이용하여 좀더 정확하고 빠른 인공지능 게임을 만드는 것이 향후 과제 이다.

**참 고 문 헌**

[1] 인공지능의 이해, <http://blogs.nvidia.co.kr/2016/08/03/difference_ai_learning_machinelearning/>

[2] 인공신경망

<https://blog.lgcns.com/1359>

[3]케라스

<https://ko.wikipedia.org/wiki/%EC%BC%80%EB%9D%BC%EC%8A%A4>

[4] open-AI gym <https://gym.openai.com/docs/>

[5] CNN의 이해

<http://hamait.tistory.com/535>

**감 사 의 글**

2013년 3월에 입학하고 다사다난한 일이 많았던 대학 생활이 벌써 졸업을 앞두고 있다는 것이 실감이 나기 시작했습니다. 입학 후 처음 1년을 원 없이 놀다 보니 군대라는 사회에 내 자신이 있었고 다시 학교로 돌아오는 시간은 그렇게 길지 않았습니다. 복학 후 열심히 해야겠다는 마음에 임베디드 연구실에 들어갔습니다. 그곳은 홍인식 교수님의 지도아래 공부할 수 있는 정말 좋은 환경 이였고 저를 몇 단계 성장할 수 있게 해준 곳 이였기에 정말 감사하다고 전하고 싶습니다. 또한 강의 시간에 정말 많은 것을 학생들에게 가르쳐 주시려고 노력하신 컴퓨터공학과 교수님들 너무 감사합니다. 뒤돌아보면 정말 짧은 시간 이였고 더 많은 것을 배울 수 있었기에 많이 아쉽기도 하고 지금까지 배운 것을 활용하여 사회에 나가서 잘할 수 있을까 라는 걱정도 들지만 지금까지 함께해온 동기들이 있기에 위안이 됩니다. 먼저 초,중,고,대학교까지 같이 오고 자취까지 같이 했던 동현이, 현재 룸메이트인 지훈이와 택윤이형, 소중한 정윤이에게 많이 기대기도하고 서로의 힘이 되어줘서 고맙다는 말을 하고 싶습니다. 그리고 함께 과제도하고 지식도 많이 공유했던 경준이, 영재, 자민이, 경무, 대영이형, 동수, 정완이형, 종화, 태훈이, 민형이 에게 고맙다는 말을 전하고 싶습니다.

마지막으로 지금까지 키워주시고 철없던 아들을 뒷바라지 하시고 보살펴 주시느라 고생 제일 많으셨던 저희 부모님에게 감사하고 표현을 많이 못하는 아들이지만 정말 사랑한다고 전하고 싶습니다.

**감 사 의 글**