

Tic-Tac-Toe (AI)



| | |
|------|----------------|
| 과목명 | 인공지능 |
| 담당교수 | 김동근 교수님 |
| 학과 | 컴퓨터공학부 컴퓨터공학전공 |
| 학번 | 201401932 |
| 이름 | 노 명 환 |

목 차

I . 서론

II . 본론

2.1 사용 기술

2.1.1 Tensorflow

2.1.2 인공 신경망

2.1.3 picle

2.1.4 tkinter

2.2 Tic-Tac-Toe 학습

2.2.1 UCI Tic-Tac-Toe Dataset

2.2.2 MLP 설계 및 학습

2.3 Tic-Tac-Toe Game with AI

III . 실험결과

3.1 Dataset 학습 결과

3.2 Tic-Tac-Toe Game 구현 결과

IV . 결론

V . 참고문헌

VI . 부록 (소스코드)

I. 서론

인공지능이란 인간이 지닌 지적 능력의 일부 또는 전체를 인공적으로 구현한 것으로 인간이 머리를 써서 해야 할 일을 기계가 하도록 만드는 것에 목적을 두고 있다.

인공지능이란 용어가 처음 등장한 것은 1956년에 존 매카시가 학회에서 처음 사용하였을 때이다. 하지만 실제로 인공지능이란 개념 자체는 앨런 튜링이 튜링 모방 게임이라는 것을 1950년에 제안하였으며 최초의 신경망 모델은 맥 클록과 월터 피츠가 1943년에 제안하였다. 1960년대에 들어서 로젠 블랫은 학습 알고리즘이 퍼셉트론의 연결 강도를 조정할 수 있다는 것을 통하여 퍼셉트론 수렴 이론을 증명하였다. 이후 로트피 자데가 퍼지 집합에 대한 논문을 발표하며 퍼지 집합 이론에 대한 토대를 마련하였으며 이를 바탕으로 똑똑한 기계와 지능형 시스템이 만들어졌다. 그러나 이후 AI 기술은 여러 가지 한계점에 부딪히게 된다. 일정 영역에 대한 문제의 해결에 대해서는 일정 수준 이상의 능력을 보여주었으나, 작업의 범위를 넓히거나 더 어려운 실세계 문제를 해결하지 못하는 현상을 보인 것이다. 이러한 문제점으로 인하여 AI의 관심은 쇠퇴하게 된다. 이후 AI는 암흑기를 맞이하다가 전문가 시스템 및 신경망 이론의 복귀 등으로 인하여 다시 부흥기를 맞는다. 루멜 하트는 역전파 알고리즘을 발견하면서 신경망에 큰 공헌을 하였다. 이후 딥러닝, 강화학습 등 여러 가지 용어가 등장하며 현대에 이르기까지 AI의 부흥기는 다시 지속되고 있다.[1][2][3]



그림 1. 튜링테스트

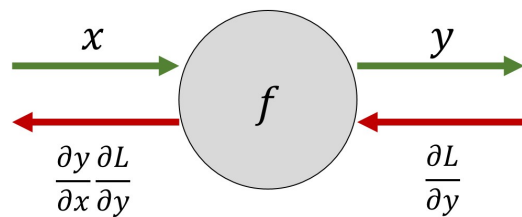


그림 2. 역전파

II. 본론

2.1 사용 기술

2.1.1 Tensorflow

Tensorflow란 구글이 2011년에 개발을 시작하여 2015년에 오픈 소스로 공개한 기계학습 라이브러리이다. 주로 Python을 사용하여 동작하며 다른 언어도 대부분 지원한다. 계산 구조와 목표 함수만 정의하면 자동으로 미분 계산 등 계산에 필요한 과정을 처리하기 때문에 편리한 사용성을 가지고 있다.

Tensorflow의 버전으로는 CPU / GPU 두 가지의 버전이 존재하며 GPU 버전에서는 GPGPU를 사용해 대량 연산을 빠르게 수행하여 훨씬 빠른 동작이 가능하다.[4]

2.1.2 인공 신경망

인공신경망(Artificial neural network, ANN)은 노드들의 결합으로 네트워크를 형성한 인공 뉴런이 학습을 통하여 노드가 가지고 있는 값들을 변경하여 문제 해결 능력을 가지는 모델이다. 인공 신경망 학습에는 크게 교사 학습 / 비교사 학습이 있으며 명확한 해답이 존재하는지 여부를 통하여 구분이 된다. 일반적으로 해답이 존재할 경우 교사 학습이 이루어지며 데이터 클러스터링과 같은 분야에서는 비교사 학습이 이용된다. 인공신경망은 많은 입력들이 주어질 경우 근사치를 추측하거나 근사치를 추측할 경우 사용한다.[5]

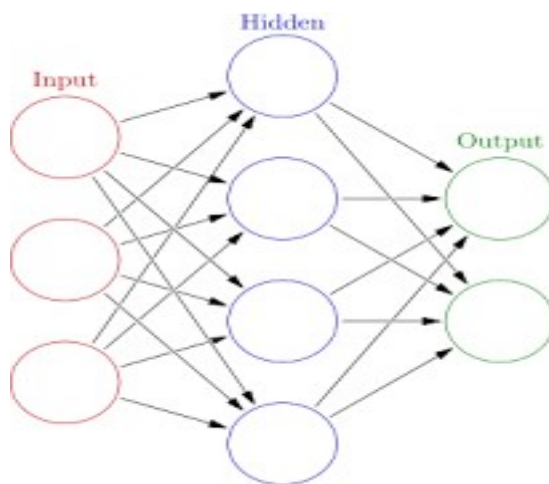


그림 3. 인공신경망

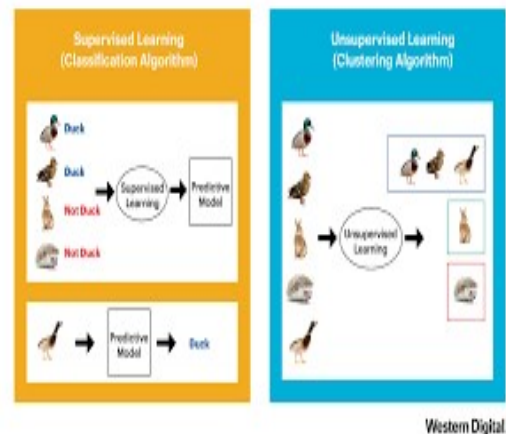


그림 4. 교사 / 비교사 학습

2.1.3 pickle

일반 텍스트를 파일로 저장할 때는 파일 입출력을 이용한다. 하지만 리스트나 클래스와 같은 텍스트가 아닌 자료형은 파일 입출력을 이용하여 데이터를 저장하거나 불러오는 것이 불가능하다. 이러한 데이터를 저장하고 불러오기 위하여 파이썬에서

는 pickle이라는 모듈을 사용한다.

2.1.4 tkinter

tkinter는 Tcl/Tk에 대한 파이썬 Wrapper로써 Tcl/Tk를 파이썬에 적용하여 사용할 수 있도록 한 GUI 모듈이다. 파이썬 설치 시 기본적으로 내장되어 있는 파이썬 표준 라이브러리로서 쉽고 간단한 GUI 프로그램을 만들 때 활용할 수 있다.[6]

2.2 Tic-Tac-Toe 학습

2.2.1 UCI Tic-Tac-Toe Dataset

UCI에서 제공하는 틱택토 게임의 데이터 셋은 게임이 끝난 상태에서 보드의 모양과 승자가 누구인지에 대한 정보를 제공한다. 이 데이터 셋은 보드의 모양을 x, o, b 이렇게 3가지로 표현하였는데 각각 x가 놓여있는 상태, o가 놓여있는 상태, 아무것도 놓여있지 않은 상태를 의미한다. 데이터는 x가 먼저 시작하는 것을 기본 전제로 생각하며 게임의 결과는 x의 승리 시 true, o의 승리 시 false로 표현하였다. 데이터에서 처음 9개는 각각 왼쪽 위부터 시작하여 오른쪽 아래의 끝까지 각 보드의 위치에 놓여있는 돌의 상태를 표현하였다.

| | | |
|------------------------------|--------------------------|-------------------------|
| Top Left {x,o,b} | Top Middle {x,o,b} | Top Right {x,o,b} |
| Middle Left {x,o,b} | Middle Middle {x,o,b} | Middle Right {x,o,b} |
| Bottom Left {x,o,b} | Bottom Middle {x,o,b} | Bottom Right {x,o,b} |
| true : X win / false : O win | | |

표 1. Tic-Tac-Toe Dataset

컴퓨터에서 학습을 하기 위하여 데이터를 처리하기 위해서는 각각의 데이터를 알파벳으로 처리할 수 없으므로 수치화를 해주어야 정확한 처리가 가능하였다. 본 연구에서는 이를 위하여 아래의 표와 같이 각각의 알파벳을 임의의 수치로 표현하였다.

| | | | | | |
|---------|----|---|---|------|-------|
| 원래 Data | x | o | b | true | false |
| 치환 Data | -1 | 1 | 0 | -10 | 10 |

표 2. 데이터 치환

이후 위와 같이 치환된 데이터를 저장 후 프로그램에서 데이터를 불러올 수 있도록 구현하였다. 입력받은 데이터는 기존 데이터 셋에서 true와 false에 대하여 정렬

이 되어있는 상태이기 때문에 그대로 학습을 시킨다면 먼저 한쪽에 편향되어 학습이 되기 때문에 이를 방지하고 numpy의 shuffle함수를 이용하여 index를 모두 랜덤하게 섞어주는 과정을 추가하였다. 이후 7:3의 비율로 training data와 test data 두 개의 세트로 나누어주는 과정을 거친 후 처음 각 행에서 처음 9개의 데이터는 input으로 사용하기 위한 x_data로 마지막 데이터는 정답을 알려주기 위한 y_data로 저장하는 과정을 수행하였다.

2.2.2 MLP 설계 및 학습

입력받은 데이터 셋을 통하여 학습을 구현하기 위하여 2층의 구조를 가지고 있는 Multi-Layer Perceptron을 설계하였다. 설계한 MLP의 구조는 다음과 같다.

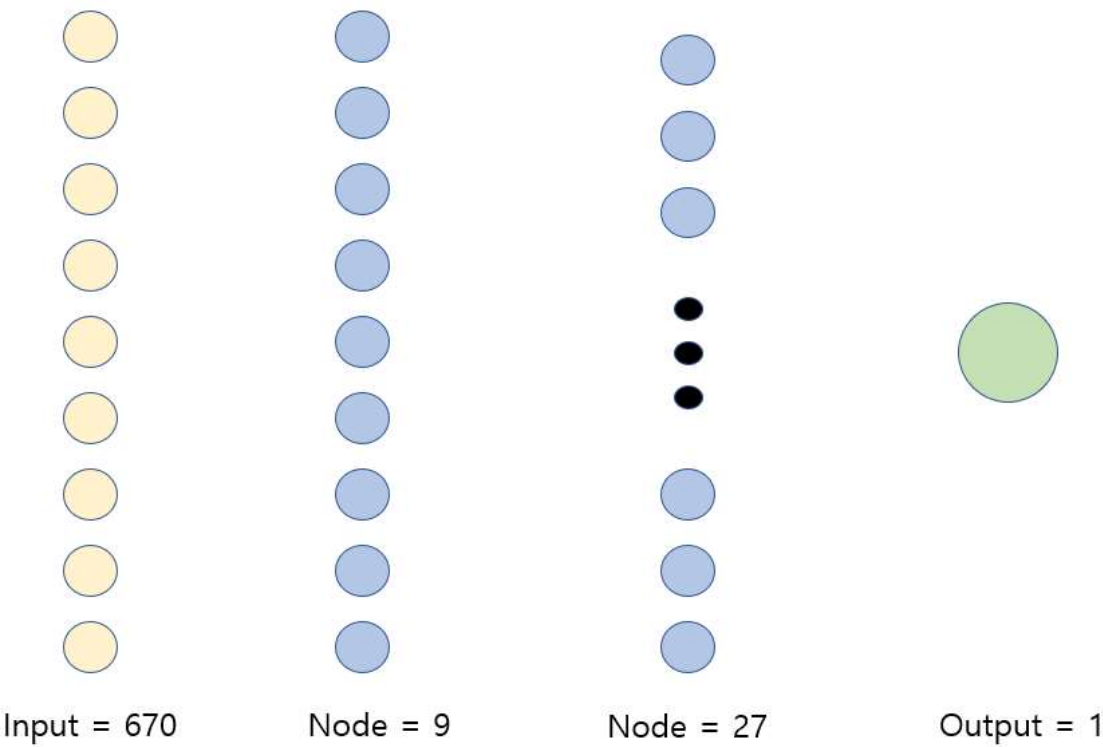


그림 5. MLP 설계도 (화살표 생략)

그림과 같이 설계된 MLP는 최초 9개의 보드 모양에 대한 input을 받는다. 이후 9개의 node와 연산 후 bias 값을 더한 후 27개의 node와 연산하는 과정을 거친 후 bias 값을 더하여 최종 output이 계산된다. 본 프로그램에서는 별도의 활성화 함수 모델을 사용하지 않고 출력된 결과 값을 다음과 같이 변경하였다.

| | | |
|------|-----------------|--------------|
| 결과 값 | output ≥ 0 | output < 0 |
| 변화 값 | 10 | -10 |

표 3. MLP 설계도 (화살표 생략)

표와 같이 변환 한 이유는 최초 true 값을 -10, false 값을 10으로 판별하였기 때문이다. 출력 값을 보다 가까운 값으로 인식하게 하여 테스트를 진행하여 정답 여부를 판별하여 정확도를 계산하였다.

이후 학습 된 파라메타들은 나중에 실제 게임에서 인공지능이 어디에 둘 것인지 판별하는 과정을 거칠 때 사용하기 위하여 pickle을 사용하여 각 파라메타들을 저장하였다.

2.3 Tic-Tac-Toe Game with AI

인공지능을 이용한 틱택토 게임을 구현하기 위하여 이미 학습 된 파라메타들을 피클을 통하여 불러와 저장하였다. 이후 input으로 보드 모양을 받으면 연산을 바로 수행할 수 있도록 forward함수를 만들어 준 후 게임을 시작하도록 구성하였다. 게임에서 컴퓨터는 자신의 차례가 되면 보드의 모양을 input으로 받는다. 그리고 이후 보드의 모양에 대하여 빈칸을 차례대로 확인하며 빈칸이 존재할 경우 컴퓨터가 놓았을 경우에 대한 output 값을 계산한다. 이 때, 컴퓨터가 이기는 경우는 최초 데이터 셋에서 10으로 정의하였기 때문에 컴퓨터는 자신이 둘 수 있는 모든 칸에 대하여 output 정보를 비교하여 최대 값이 나타나는 칸이 가장 승률이 높다고 판단하며 그 위치에 돌을 놓도록 구성하였다.

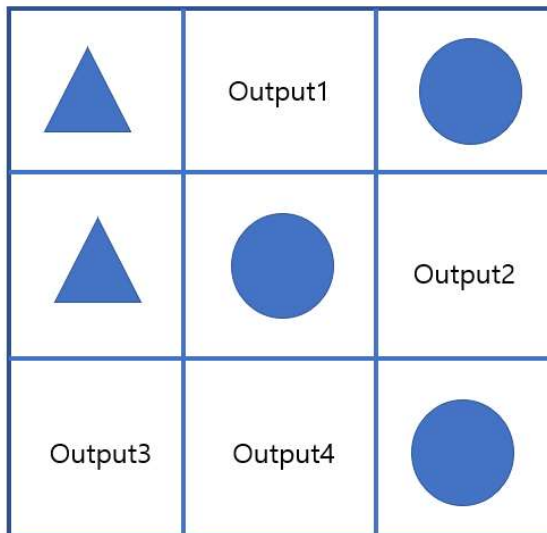


그림 6. AI 게임 설계도

위의 그림을 예로 들어보면 현재는 세모의 차례인 것을 알 수 있다. 이 때, 세모가 돌을 착수할 수 있는 위치는 Output1, Output2, Output3, Output4 총 4가지이다. 컴퓨터는 4가지의 위치에 모두 착수를 한 이후 보드의 모양을 기준으로 결과 값을 계산하게 된다. 이후 이기는 값이 가장 크게 나타는 위치를 선택하여 착수를 하게 된다.

III. 실험결과

3.1 Dataset 학습결과

| 항목 | 단층 퍼셉트론 | MLP (2층) | MLP (2층) |
|---------------|----------------|------------------|-------------------------------|
| 학습 횟수 | 1000번 | 1000번 | 1000번 |
| Optimizer | Adam Optimizer | Adam Optimizer | Adam Optimizer |
| Learning Rate | 1e-1 (0.1) | 1e-3 (0.001) | 3e-1 (0.3) |
| Hidden Layer | 9 | 1층 : 9 / 2층 : 27 | 1층 : 9 (sigmoid) / 2층 : 27 |
| 정답 수 | 283 | 285 | 286 |
| 오답 수 | 5 | 3 | 2 |
| 정확도 | 98.26% | 98.96% | 99.03% |

표 4. 학습 결과 (단층 및 다층 퍼셉트론)

학습 결과 MLP가 보다 높은 정확도를 보여주는 것을 확인할 수 있다. 하지만 단층에서도 이미 충분히 학습이 잘 되어있는 상태였기 때문에 큰 차이를 확인할 수는 없었다. MLP는 두 가지 경우를 나누어서 학습을 하였는데 첫 번째 히든 층에서 sigmoid 활성화 함수를 적용하는지 여부에 따라 나누어 진행하였다. 그 결과, 서로 비슷한 결과를 보였지만 sigmoid를 적용한 경우가 조금 더 나은 결과를 보였다.

3.2 Tic-Tac-Toe 게임 구현 결과

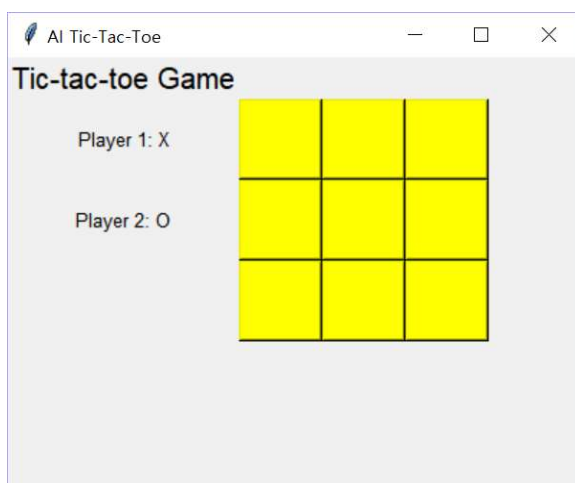


그림 7. AI Tic-Tac-Toe 최초 실행화면

게임에서 테스트는 컴퓨터가 선수를 잡은 상황이라고 가정한 후 진행하였다.

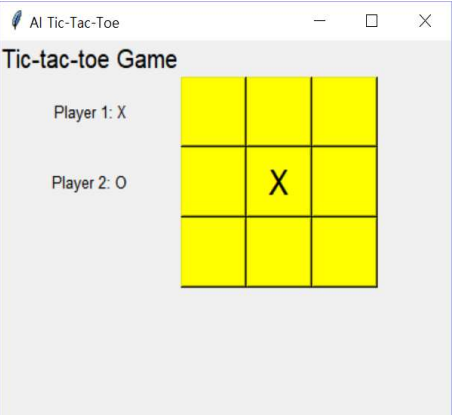
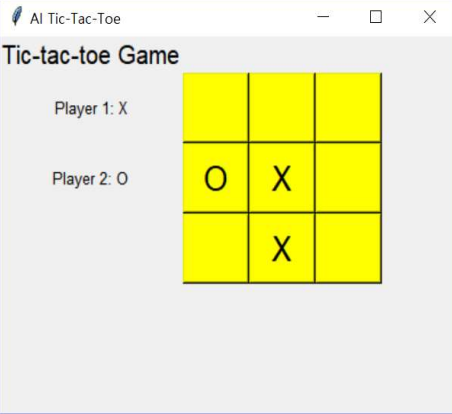
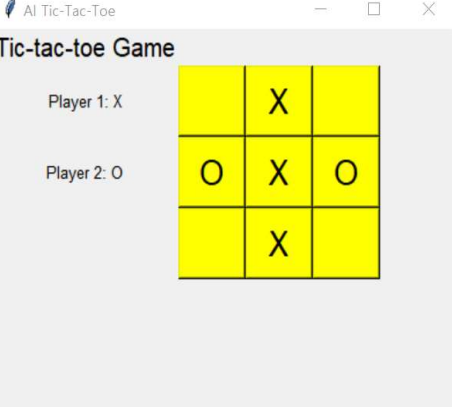
| 컴퓨터 예측 | 결과 |
|---|--|
|  | <p>최초 실행 시 학습 결과 컴퓨터는 가운데에 두는 것이 가장 유리하다고 판단하기 때문에 가운데에 선수를 두었다. 이후 사용자가 두는 위치에 따라서 컴퓨터는 빈칸에서 자신의 수를 둔 이후 가장 승률이 높은 위치의 인덱스에 수를 두도록 설계하였다.</p> |
|  | <p>Board : [0, 0, 0, 1, -1, 0, 0, 0, 0] index : 7, [27.40023661]</p> <p>사용자가 왼쪽 가운데에 수를 두자 컴퓨터는 연산 결과 7번 인덱스 위치가 가장 최선의 수라고 판단하여 착수하였다.</p> <p>현재 상황에서 컴퓨터는 가운데 위를 두면 게임을 이길 수 있다.</p> |
|  | <p>[0, 0, 0, 1, -1, 1, 0, -1, 0] index : 1, [27.43490252]</p> <p>일부러 엉뚱한 위치에 돌을 놓은 결과 컴퓨터는 자신이 이기는 위치를 올바르게 찾아 둬으로써 게임은 컴퓨터의 승리로 끝났다.</p> |

표 5. 컴퓨터 vs 사용자

위의 실험에서는 컴퓨터가 효율적으로 수를 잘 뒤 사용자를 손쉽게 이기는 모습을 보여주었다. 하지만 이후 수백 번의 자체 테스트 결과 컴퓨터는 자신이 이길 수 있는 위치가 있음에도 불구하고 계속하여 엉뚱한 위치에 두거나 상대방이 이기는 상황에서도 방해를 하지 않는 모습을 보였다. 똑같은 패턴으로 진행하면 같은 패턴이 나온다는 점만 제외한다면 랜덤 함수가 아닐까하는 생각이 들만큼 실제 게임에서는 높은 성능을 보이지는 못하였다.

IV. 결론

본 실험에서 학습 시 사용한 데이터 개수는 1000개 미만의 데이터로 상당히 제한적인 양의 데이터이다. 그 결과 학습 결과가 단층 퍼셉트론과 다층 퍼셉트론에서 큰 차이가 나타나지 않는 모습을 보였다. 또한 실제 게임을 구현하였을 때 좋은 결과를 보이지 못하였는데 이에 대한 원인을 크게 2가지로 분류하였다.

1. 트레이닝 데이터의 수가 너무 적다.

실제 트레이닝 데이터의 개수는 958개로 실제 틱택토 게임에서 나타나는 경우의 수(약 26만개)에 비교한다면 너무나도 적은 숫자이다. 또한 데이터의 개수가 너무 적어 과적합(overfitting)이 일어날 확률이 크다. 제한된 데이터 내에서 학습이 이루어진 결과 실제 게임에서는 올바른 판단을 하지 못하였다고 사료되는 부분이다.

2. 트레이닝 데이터가 실제 게임에 적합하지 않다.

트레이닝 데이터는 모두 끝난 게임에서 누가 이겼는지를 판단하는 데이터이다. 이는 틱택토 게임을 진행하는 과정에서 이루어지는 모든 판단의 수가 게임을 이기는 방향으로 이끌어 간다고 판단하기에는 다소 무리가 있다.

이러한 문제점을 보완하기 위해서는 데이터의 개수가 많아야하며 또한 각 상황에서 어느 수를 두어야 하는지에 대한 데이터가 필요하다고 생각된다.

본 실험을 진행하면서 가장 크게 느꼈던 점은 데이터의 중요성이었다. 학습이 아무리 잘 이루어졌다고 하더라도 실제 게임에서는 그보다 훨씬 많은 경우의 수가 존재하기 때문에 정확한 판단을 기대하는 것은 힘든 일이었다. 만약 데이터가 훨씬 많이 있고 학습이 잘 이루어진다면 거의 대부분의 경우에서 현재보다 성능을 높이는 방향으로 충분히 개선이 잘 이루어질 수 있을 것이라고 판단된다.

현대 시대에 들어서 인공지능이라는 분야는 4차 산업에서 가장 큰 이슈로 떠오르고 있다. 이러한 상황에서 직접 데이터를 핸들해보며 학습을 시키는 일은 귀중한 경험이라고 생각된다. 현대 시대에는 전산 및 통계학이 발전하면서 인터넷에는 엄청난 수의 데이터들이 떠돌고 있다. 이러한 데이터들 사이에서 일련의 규칙을 발견하여 학습 후 콘텐츠를 창출하는 것이 미래시장의 핵심이라고 생각한다. 단순히 학습만이 아닌 데이터 마이닝이라는 분야 또한 수업에서 함께 접할 수 있다면 좋을 것 같다는 생각이 들었다.

V. 참고문헌

[1] 위키피디아 : 인공지능

https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5#%EC%A0%84%EB%AC%B8%EA%B0%80_%EC%8B%9C%EC%8A%A4%ED%85%9C%EC%9D%98_%EC%83%81%EC%8A%B9

[2] 나무위키 : 인공지능

<https://namu.wiki/w/%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5>

[3] 인공지능 개론 (마이클 네그네비츠키, 한빛아카데미) 1장

[4] 나무위키 : 텐서플로

<https://namu.wiki/w/%ED%85%90%EC%84%9C%ED%94%8C%EB%A1%9C%EC%9A%B0?from=TensorFlow>

[5] 위키피디아 : 인공 신경망

https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B3%B5_%EC%8B%A0%EA%B2%BD%EB%A7%9D

[6] Python pickle

<http://pythonstudy.xyz/python/article/120-Tkinter-%EC%86%8C%EA%B0%9C>

VI. 부록

개발 환경

[OS] Window 10 Home

[Language] Python 3.6.6 64-bit

[IDE] PyCharm Community 2018.1.4

[Source Code]

1. learning.py (학습 및 학습 파라메타 저장)

```
import numpy as np
import tensorflow as tf
import pickle

####Data Handling###

#read data from csv file
data_all = np.genfromtxt('data.csv', delimiter=',', dtype=np.float32)
np.random.shuffle(data_all)

#Separate data by 7:3 rate
train_num = int(len(data_all)*0.7)
test_num = len(data_all)-train_num
data_train = data_all[:train_num]
data_test = data_all[train_num:]
x_train = data_train[:, 0:-1]
y_train = data_train[:, [-1]]
```

```

x_test = data_test[:, 0:-1]
y_test = data_test[:, [-1]]

###Train data###
NUM_ITER = 1000
NUM_HIDDEN = 27

X = tf.placeholder(tf.float32, [None, 9])
Y = tf.placeholder(tf.float32, [None, 1])
"""단층 네트워크
W = tf.Variable(tf.random_normal([9, 1]))
B = tf.Variable(tf.random_normal([1]))

H = tf.matmul(X, W) + B
"""

#MLP
W = tf.Variable(tf.random_uniform([9, NUM_HIDDEN], -0.5, 0.5))
B = tf.Variable(tf.random_uniform([1, NUM_HIDDEN], -0.5, 0.5))

#hiddenLayer = tf.sigmoid(tf.matmul(X, W)+B)
hiddenLayer = tf.matmul(X, W)+B

W2 = tf.Variable(tf.random_uniform([NUM_HIDDEN, 1], -0.5, 0.5))
B2 = tf.Variable(tf.random_uniform([1], -0.5, 0.5))
H = tf.matmul(hiddenLayer, W2)+B2
#outLayer = tf.matmul(hiddenLayer, W2)+B2
#H = tf.sigmoid(outLayer)

cost = tf.reduce_mean(tf.square(H-Y))
optimizer = tf.train.AdamOptimizer(1e-3)
train = optimizer.minimize(cost)

cost_list = []
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for step in range(NUM_ITER):
        sess.run(train, feed_dict={X: x_train, Y: y_train})
        w_pred = sess.run(W).tolist()
        b_pred = sess.run(B).tolist()[0]
        w_pred2 = sess.run(W2).tolist()
        b_pred2 = sess.run(B2).tolist()[0]
        # 1회학습마다 업데이트 된 파라메타 출력
        print(step, w_pred, b_pred, w_pred2, b_pred2)

    print("Test the data")
    true_num = 0
    false_num = 0
    for i in range(0, test_num):
        result = sess.run(H, feed_dict={X: [x_test[i]]})

```

```

        #print(y_test[i], result)

    if(result>=0):
        result = 10
    else:
        result = -10
    if result == y_test[i]:
        true_num = true_num + 1
    else:
        false_num = false_num + 1

#Save the parameters
W_list = w_pred
W2_list = w_pred2
B_list = b_pred
B2_list = b_pred2

with open("W_param.pkl", 'wb') as f:
    pickle.dump(W_list, f)

with open("W2_param.pkl", 'wb') as f:
    pickle.dump(W2_list, f)

with open("B_param.pkl", 'wb') as f:
    pickle.dump(B_list, f)

with open("B2_param.pkl", 'wb') as f:
    pickle.dump(B2_list, f)

print(true_num, false_num)

```

2. check.py (틱택토 게임)

```

import pickle
from tkinter import *
from tkinter import messagebox
import numpy as np

with open("W_param.pkl", 'rb') as f:
    W = pickle.load(f)

with open("W2_param.pkl", 'rb') as f:
    W2 = pickle.load(f)

with open("B_param.pkl", 'rb') as f:
    B = pickle.load(f)

with open("B2_param.pkl", 'rb') as f:
    B2 = pickle.load(f)

def forward(X):

```

```

hidden = np.matmul(X, W) + B
hidden = 1 / (1 + np.exp(-hidden))
result = np.matmul(hidden, W2) + B2
return result

def findBlank(myBoard):
    print(myBoard)
    predict = -100
    index = 0
    board = myBoard
    for i in range(len(board)):
        if(board[i]==0):
            board[i] = 1
            result = forward(board)
            print(i, result)
            #컴퓨터가 먼저 시 최솟값
            #사람이 먼저 시 최댓값
            if(predict < result):
                predict = result
                index = i
            board[i]=0
    print("computer : " + str(index) + ", " + str(predict))
    return index

#Game

window=Tk()

window.title("AI Tic-Tac-Toe")
window.geometry("400x300")

lbl=Label(window,text="Tic-tac-toe Game",font=('Helvetica','15'))
lbl.grid(row=0,column=0)
lbl=Label(window,text="Player 1: X",font=('Helvetica','10'))
lbl.grid(row=1,column=0)
lbl=Label(window,text="Player 2: O",font=('Helvetica','10'))
lbl.grid(row=2,column=0)

turn=1; #For first person turn.

global board
global next
global first
next = 0
board = [0, 0, 0, 0, 0, 0, 0, 0, 0]
first = 1

def put(next):
    if next==0:
        clicked1()
    elif next==1:

```

```

        clicked2()
    elif next==2:
        clicked3()
    elif next==3:
        clicked4()
    elif next==4:
        clicked5()
    elif next==5:
        clicked6()
    elif next==6:
        clicked7()
    elif next==7:
        clicked8()
    elif next==8:
        clicked9()

def clicked1():
    global turn
    global next
    global first
    if btn1["text"]==" ":    #For getting the text of a button
        if turn==1:
            turn =2;
            board[0] = -1
            btn1["text"]="X"
        elif turn==2:
            turn=1;
            board[0] = 1
            btn1["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked2():
    global turn
    global next
    global first
    if btn2["text"]==" ":
        if turn==1:
            turn =2;
            board[1] = -1
            btn2["text"]="X"
        elif turn==2:
            turn=1;
            board[1] = 1
            btn2["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked3():

```

```

global turn
global next
global first
if btn3["text"]==" ":
    if turn==1:
        turn =2;
        board[2] = -1
        btn3["text"]="X"
    elif turn==2:
        turn=1;
        board[2] = 1
        btn3["text"]="O"
    check();
    if turn==first:
        put(findBlank(board))

def clicked4():
    global turn
    global next
    global first
    if btn4["text"]==" ":
        if turn==1:
            turn =2;
            board[3] = -1
            btn4["text"]="X"
        elif turn==2:
            turn=1;
            board[3] = 1
            btn4["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked5():
    global turn
    global next
    global first
    if btn5["text"]==" ":
        if turn==1:
            turn =2;
            board[4] = -1
            btn5["text"]="X"
        elif turn==2:
            turn=1;
            board[4] = 1
            btn5["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked6():

```



```

global turn
global next
global first
if btn6["text"]==" ":
    if turn==1:
        turn =2;
        board[5] = -1
        btn6["text"]="X"
    elif turn==2:
        turn=1;
        board[5] = 1
        btn6["text"]="O"
    check();
    if turn==first:
        put(findBlank(board))

def clicked7():
    global turn
    global next
    global first
    if btn7["text"]==" ":
        if turn==1:
            turn =2;
            board[6] = -1
            btn7["text"]="X"
        elif turn==2:
            turn=1;
            board[6] = 1
            btn7["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked8():
    global turn
    global next
    global first
    if btn8["text"]==" ":
        if turn==1:
            turn =2;
            board[7] = -1
            btn8["text"]="X"
        elif turn==2:
            turn=1;
            board[7] = 1
            btn8["text"]="O"
        check();
        if turn==first:
            put(findBlank(board))

def clicked9():

```

```

global turn
global next
global first
if btn9["text"]==" ":
    if turn==1:
        turn =2;
        board[8] = -1
        btn9["text"]="X"
    elif turn==2:
        turn=1;
        board[8] = 1
        btn9["text"]="O"
    check();
    if turn==first:
        put(findBlank(board))

flag=1;
def check():
    global flag;
    b1 = btn1["text"];
    b2 = btn2["text"];
    b3 = btn3["text"];
    b4 = btn4["text"];
    b5 = btn5["text"];
    b6 = btn6["text"];
    b7 = btn7["text"];
    b8 = btn8["text"];
    b9 = btn9["text"];
    flag=flag+1;
    if b1==b2 and b1==b3 and b1=="O" or b1==b2 and b1==b3 and b1=="X":
        win(btn1["text"])
    if b4==b5 and b4==b6 and b4=="O" or b4==b5 and b4==b6 and b4=="X":
        win(btn4["text"]);
    if b7==b8 and b7==b9 and b7=="O" or b7==b8 and b7==b9 and b7=="X":
        win(btn7["text"]);
    if b1==b4 and b1==b7 and b1=="O" or b1==b4 and b1==b7 and b1=="X":
        win(btn1["text"]);
    if b2==b5 and b2==b8 and b2=="O" or b2==b5 and b2==b8 and b2=="X":
        win(btn2["text"]);
    if b3==b6 and b3==b9 and b3=="O" or b3==b6 and b3==b9 and b3=="X":
        win(btn3["text"]);
    if b1==b5 and b1==b9 and b1=="O" or b1==b5 and b1==b9 and b1=="X":
        win(btn1["text"]);
    if b7==b5 and b7==b3 and b7=="O" or b7==b5 and b7==b3 and b7=="X":
        win(btn7["text"]);
    if flag ==10:
        messagebox.showinfo("Tie", "Match Tied!!! Try again :)")
        window.destroy()

def win(player):
    ans = "Game complete " +player + " wins ";

```

```
messagebox.showinfo("Congratulations", ans)
window.destroy() # is used to close the program
```

```
btn1 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked1)
btn1.grid(column=1, row=1)
btn2 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked2)
btn2.grid(column=2, row=1)
btn3 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked3)
btn3.grid(column=3, row=1)
btn4 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked4)
btn4.grid(column=1, row=2)
btn5 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked5)
btn5.grid(column=2, row=2)
btn6 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked6)
btn6.grid(column=3, row=2)
btn7 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked7)
btn7.grid(column=1, row=3)
btn8 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked8)
btn8.grid(column=2, row=3)
btn9 = Button(window, text=" ",bg="yellow",
fg="Black",width=3,height=1,font=('Helvetica','20'),command=clicked9)
btn9.grid(column=3, row=3)

window.mainloop()
```