인공신경망을 이용한 숫자 인식 시스템 개발

정채은*, 김병욱 *‡
*동국대학교 경주캠퍼스 컴퓨터공학과
e-mail : jackpot1112@naver.com, bwkim@dongguk.ac.kr

Development of neumerical recognition system using artificial neural network

Chae-Eun Jeong*, Byung-Wook Kim *‡
*Dept. of Computer Science, Dong-Guk University-Gyeongju

요 약

인공신경망은 인간의 신경세포인 뉴런을 모델로서 사용했다. 인간은 외부에서 오는 정보를 뇌에서 받아들이고 판단한다. 받아들인 정보를 통해 어떻게 산출할 것인지에 대한 일들을 기능하게 된다. 그러한 일련의 과정을 필기체 숫자 데이터를 통하여 사람이 유도하는 예측 값을 인식해내고, 학습된 예측 값을 실제 값과 비교해 분석하였다. 그리고 더 나아가 인공신경망에 대해 어떻게 응용할 것인지 논의하였다.

1. 서론

1.1 연구의 필요성

최근 인공지능이 유행하고, 그에 따라 자연히 '기계의 모델을 인간으로 만들면 어떻게 해야 할까?' 라는 의문점에서부터 이 연구에 대해 초점을 맞추어 가기 시작하였다. 이러한 호기심은 추상적이었고, 객관적인 표현을 생각하는데 있어 생긴 두 키워드가 있다. 그것은 '데이터'와 '학습'이다. 꽤나 간단한 두 키워드를 어떤 방식으로 개발을 하면 될까 고민하였다. 그 단계에서 많은 데이터를 통해 학습과정을 거쳐 예측값에 도달 할 수 있는 인공 신경망 알고리즘(Artificial Neural Network Algorithm)을 이용하기로 하였다.

신경망의 기술에 대한 응용분야는 사실상 광범위하다. 예를 들어 Robot, Data Mining, Financial Services, Language recognition, Information Engineering 과 같은 수많은 분야에 직간접적 영향을 끼치고 있다. 그 중에서 언어 인식을 채택하여 실험하고자 하는데, 이유는 비교적 얻기 쉬운 데이터이기 때문이다. 사실 언어 인식도 깊이 들어가기 보다는 수준을 고려해 숫자 데이터로서 값을 얻을 예정이다. 학습을 하려면 대량의 데이터가 필요한데, 수집이 어려운 데이터는 시간이나 비용적인 측면에서 부득이한 애로사항을 겪게 만들 것이다.

1.2 연구의 목적

뉴런을 기반한 인공 신경망은 학습을 통해 데이터를 update 한다. 이 때 계속해서 주어진 데이터를 통해서 최신의 정보를 얻을 것이고, 최신의 정보는 보다 더 정확하고 의미 있는 성격을 가지게 된다. 이 연구에 있어 데이터는 어떻게 표현할 것인지, 어떻게 학습할 것인지에 치중을 해 분석할 것이다. 또한 정확한 것에 대한 척도 즉, 성능에 대해서도 알아볼 것이다. 이 때 사람이 임의로 쓴 숫자로 데이터가 주어질 것이다. 그리하여 본 개발의 목적은 사람이 쓴 손글씨 데이터를 컴퓨터가 인식하고, 무슨 숫자인지 맞추는 것과 그것이 얼마나 정확한 지 도출하는 것이다.

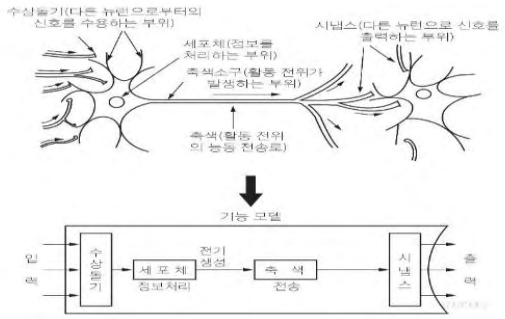
1.3 연구문제

서론에 나온 내용에 토대로 했을 때 연구의 문제는 이러하다.

1) 어떤 데이터를 쓸 것인가?
2) 데이터를 어떻게 표현할 것인가?
3) 어떻게 학습시킬 것인가?
4) 출력 값의 범위와 개수는 어떻게 되는가?
5) 출력 값이 실제 결과값에 비교해 어느 정도의 성능 혹은 정확도를 갖고 있는가?

2. 이론적 배경

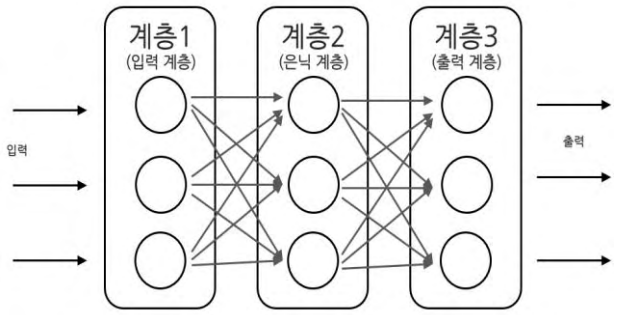
2.1 인공신경망의 기본 개념



출처-NAVER 지식백과 컴퓨터인터넷 IT 용어대사전 뉴런

(그림 1) 뉴런과 기능 모델

인공신경망은 앞서 말한 것과 같이 생물학적으로 뉴런에 기초했다. 뉴런에 있는 수상돌기는 신호를 받아 들이고, 축삭돌기는 신호를 보내며, 시냅스가 다시 다른 뉴런으로부터 받은 신호를 출력한다. 그러한 원리와 과정을 컴퓨터적 사고에 빗대어 표현한 것이 (그림 1)에서의 기능 모델이다. 이러한 뉴런의 시놉시스로 신호를 데이터로 생각하고 이 뉴런들을 거쳐서 학습하는 과정을 연상할 수 있다. 게다가 단순화된 뉴런을 보면 입력-처리-출력의 간단한 순서를 떠올릴 수 있다.



(그림 2) 3 계층 신경망

그렇다면 여러 개의 입력-처리-출력 순서를 가진 뉴런이 상호연결이 되어있는 (그림 2)와 같다. 각각의 데이터 연결 지점을 노드(node)라고 부르는데, 이 입력 노드의 개수는 얼마나 많은 데이터를 입력하느냐에 따라 달라질 것이다. 그리고 화살표는 데이터의 이동을 나타낸다. 각각 계층 1 은 입력 계층(Input Layer), 계층 2 는 은닉 계층(Hidden Layer) 계층 3 은 출력 계층(Output Layer)이다. 입력 계층은 말 그대로 입력을 받아들이는 계층이다. 이 입력 받은 계층이 가중치와 활성화 함수 등 연산을 거쳐 은닉 계층으로 간다. 그 후 은닉계층에서도 연산을 거쳐 출력 계층에서 최종 출력의 형태로 나오게 된다. 신경망의 모습이 반드시 이러한 것은 아니며 계층의 수나 노드의 수는 데이터에 따라 항시 바뀐다. 이러한 신경망의 모델 혹은 구조를 퍼셉트론(Perceptrone)이라고 하는데, 출력 값이 하나인 단층 퍼셉트론과 출력 값이 여러 개인 다층 퍼셉트론으로 나뉜다.

2.2 활성화 함수와 가중치

뉴런은 입력을 받고 바로 출력이 나오지 않는다. 예를 들어 우리에게 상대방이 “너의 혈액형은 무엇이니?” 라고 질문 했을 때, 우리는 작은 시간 안에 혈액형을 대답할 것이다. 그리고 그 대답은 태어나서 지금까지 거쳐온 학습 데이터로 형성되고 반복된 정보일 것이다. 하지만 우리에게 혈액형을 알려줄 만한 정보가 이제껏 없었다면 대답을 하지 못 했을 것이다. 이렇듯이 우리는 정보의 정확도에 따라 어떤 상한선을 넘어야 출력이 된다는 것을 알 수 있다. 마찬가지로 어떤 분계점(Threshold)을 넘어야 출력해주는 함수가 바로 활성화 함수(Activation Function)이다.

활성화 함수의 종류로는 tanh, Step Function, ReLU, Maxout, ELU 등 많은 종류가 있지만 다층 퍼셉트론에서 보통 쓰이는 시그모이드 함수(Sigmoid Function)을 사용할 예정이다.

노드에서의 데이터 이동에는 개별적으로 가중치(Weight)가 따라 붙는다. 가중치가 붙는 이유는 한 데이터에 대한 가치를 수치로 매기는 것과 같다고 생각한다. 그러한 가치는 신뢰성 혹은 정확성에 근거하고, 그 데이터가 중요할수록 높은 신호를 띄고 그렇지 않을 수록 낮은 신호를 띌 것이다. 이러한 현상은 우리가 한 주제에 대해서 생각을 할 때 다른 여러 요인들을 함께 떠올리게 되고 그 정보들의 상호 관련성에 입각해 더 정확한 정보로 만들곤 한다. 그렇기 때문에 데이터에 대한 가중치가 필요하다.

2.3 행렬곱과 전파법

행렬곱(Matrix Multiplication)은 행렬(Matrix)의 곱셈 연산이다. 이 연구에서 어떠한 데이터들과 가중치 값들을 표현하는 데 있어서 행렬이 유용할 것이다. 우리의 신경망은 신호를 각각의 레이어에 전달(Feed)을 하는데, 그 때 전달해줄 연산이 행렬곱의 역할이다.

다시금 (그림 2)를 상기해보면 왼쪽에서부터 입력을 받아 계층 1,2,3 을 거쳐 전달한다. 이렇게 앞으로 신호를 넘겨주는 방식을 순전파(Forward Propagation)라고 한다. 그럼 그러한 순전파의 방식대로 입력 값을 행렬곱 연산을 해서 신호를 전달하고 활성화 함수를 통해 출력 값을 도출하면 실제 값과의 오차를 구할 수 있다. 그러면 반대로 그 결과 값을 통해서 출력에서 반대 방향으로 가중치에 비례한 오차로 가중치 업데이트(Update)를 해주면 된다. 이러한 방식을 역전파(Backpropagation)이라고 한다.

3. 연구방법

본 연구에서의 핵심은 많은 데이터들을 학습시키고자 하는 것인데, 데이터에 대한 직관적인 실험 결과를 필요로 하고 이에 상응하는 데이터 표현 역시 요구된다. 그렇기 때문에 많은 라이브러리를 참조할 수 있는 Python3 를 기본 언어로 설정하고, 처리의 결과를 인터프리터로 바로 알 수 있는 iPython 을 사용할 것이다. (필자는 python 3.5.4 version 이다.) 참고로 사용하고자 하는 라이브러리가 내장되어 있기 때문에 더욱이 편리할 것이다. 이 iPython 은 Anaconda Prompt 를 통해 jupyter notebook 명령어를 치면 인터렉티브 파이썬을 실행할 수 있는 웹 브라우저가 열린다. 해당 실행 환경에서 연구를 진행 해보도록 하겠다.

3.1 신경망 초기화

이론으로 살펴봤던 신경망을 클래스로 정의해주고 생성자인 init 함수에 매개 변수로 입력 계층, 은닉 계층, 출력 계층의 노드 개수를 설정을 해준다. 해당 연구에서는 28x28 픽셀의 행렬이니 784 개의 입력 노드와 100 개의 은닉 노드 10 개의 출력 노드로 초기화한다. 은닉 노드가 100 개인 이유는 따로 존재 하지 않는다. 은닉 노드의 수는 결과에 따라 계속 조정하면서 더 나은 쪽으로 변화될 것이다. 출력 노드는 나타낼 숫자가 10 개이므로 10 개로 나타내고자 했다.

3.2 데이터의 수집과 표현

숫자가 0 부터 9 까지 있다고 했을 때 우리는 여백의 한 칸 또는 한 단면에 이 숫자를 표현할 수 있다. 그렇다면 어떻게 연산 처리를 하도록 할 수 있을까? 이미지를 보았을 때 그 이미지에 대한 최소 단위는 아마 한 픽셀(Pixel)일 것이다. 그리고 그 한 픽셀에는 0 부터 255 까지의 값을 가지는 RGB 색상을 알 수 있을 것이다. 이미지는 각각의 가로와 세로를 가지는 단면이며 그것은 행렬로써 나타낼 수 있겠다. 또한 그 행렬은 2 차원 배열로 다시금 고칠 수 있다. 그러한 데이터를 가지고 있는 MNIST 손 글씨 데이터 셋(28x28 pixel csv file)을 이용할 것이다. 추가적으로 데이터의 표현을 위해 matplotlib 라이브러리를 사용할 것이다.

입력 데이터뿐만 아니라 가중치 값도 행렬을 적용한다. 대신 가중치는 실수로 0.0 에서부터 1.0 까지의 랜덤 값을 가진다. 처음에는 랜덤 값을 가지지만 나중에는 오차 계산법을 통하여 가중치의 업데이트가 일어난다.

3.3 행렬의 변환과 계산

연구에서 사용할 28x28 이미지의 정보는 index 0 에는 해당 숫자의 값이, 1 부터 284 index 에는 한 pixel 에 있는 RGB 값이 나열되어 있다. MNIST 에서 제공되는 숫자 데이터를 보기 위해 file open 으로 그래픽 처리된 데이터를 볼 수 있다. 이러한 28x28 RGB 정보를 2 차원 행렬로 바꾸어 주기 위해 numpy 라이브러리를 사용해 array 라는 함수를 호출하고 T 로 변환시켜줄 것이다. 이 행렬로 바뀐 입력 리스트로 Hidden Layer 에 들어가는 신호를 dot 함수를 통해 계산한다. dot 함수는 신호를 연산 하기 위해 행렬곱을 해주는 연산함수이다. 그리하여 계산된 Hidden Input 을 activation_function 함수에 시그모이드 연산을 대입 해준다. 시그모이드 연산을 하기 위해서는 scipy.special 라이브러리가 사용되고 expit()함수를 호출해 대입하면 된다.

$$y = \frac{1}{1 + e^{-x}}$$

(그림 3) 시그모이드 수식

같은 방법으로 은닉 계층에서 연산된 값을 출력 계층에 들어갈 때 다시 행렬곱과 활성화 함수를 통한 연산을 거치게 된다. 저장되어 있던 label(index0)의 실제 값에 출력 값을 빼서 오차를 계산한다. 다음 단계로 오차들을 가중치에 의해 나누어 다시 조합을 한다. Transpose 함수로 개별 계층 간 가중치를 update 시켜줄 수 있다.

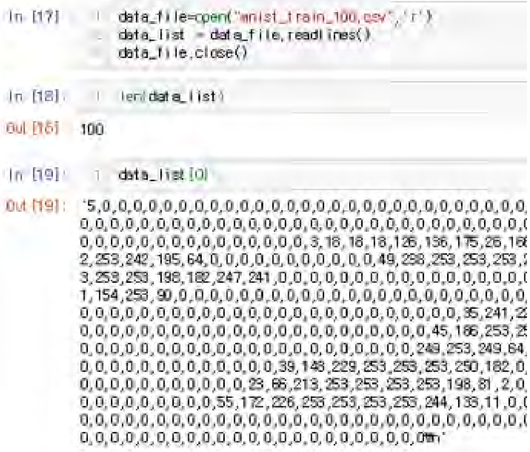
3.4 학습과 성능

신경망에 대한 초기화가 이루어지고, 학습 함수와 질의 함수가 만들어지게 되면 신경망의 객체(Instance)를 생성해 데이터를 List 로 가져온다. 그 다음에 RGB 로 저장된 값들을 쉼표(,)로 분리를 해서 학습을 시켜 결과 값을 가져온다. 이러한 시도를 준비된 데이터셋으로 실험한다.

해당 실험에 대한 정확도를 알아보기 위해 실제 값과 적합한지 아닌지에 대한 판단 리스트를 만들어서, 일치할 경우 1 을 더해주고 틀릴 경우 0 을 더해준다. 구한 결과를 리스트의 합을 사이즈로 나누어서 성능(Performance)을 본다.

4. 연구결과

먼저 데이터가 어떤 값으로 이루어져 있는지 확인하기 위해 파일 open 함수를 통해 데이터 리스트를 확인 했다.



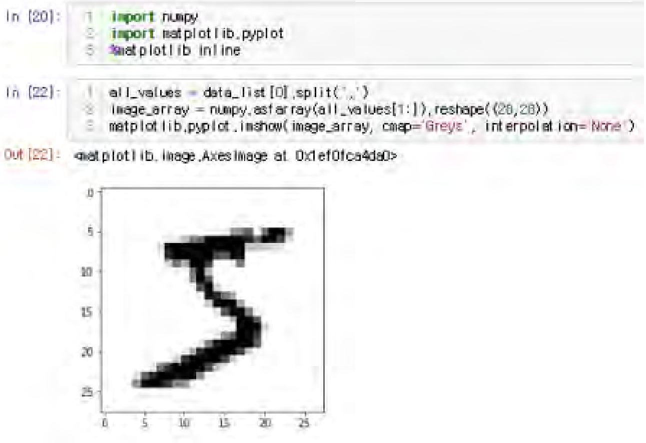
```
In [17]:  data_file=open("mnist_train_100.csv",'r')
          data_list = data_file.readlines()
          data_file.close()

In [18]:  len(data_list)
Out[18]:  100

In [19]:  data_list[0]
Out[19]:  '5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,18,18,18,126,136,175,26,166
          2,253,242,195,64,0,0,0,0,0,0,0,0,0,0,0,49,238,253,253,253,
          3,253,253,198,182,247,241,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(
          1,154,253,90,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,35,241,2
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,45,186,253,2
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,249,253,249,64,
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,39,148,229,253,253,253,250,182,0,
          0,0,0,0,0,0,0,0,0,0,0,0,23,66,213,253,253,253,253,198,81,2,0,
          0,0,0,0,0,0,0,0,0,0,55,172,226,253,253,253,253,244,133,11,0,(
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(
          0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\n'
```

(그림 4) 데이터셋 확인

해당 데이터의 길이가 100 이라고 나오는 것은 레이블(Label) 즉, 데이터의 개수가 100 개인 것이다. 맨 첫번째 값을 보면 '5'가 있는데 이것이 실제 값이다. 그리고 나머지 부분은 모두 RGB 값으로 이루어져 있는 이미지를 표현한 것이다.



```
In [20]:  import numpy
          import matplotlib.pyplot
          %matplotlib inline

In [22]:  all_values = data_list[0].split(',')
          image_array = numpy.asfarray(all_values[1:]).reshape((28,28))
          matplotlib.pyplot.imshow(image_array, cmap='Greys', interpolation='None')

Out[22]:  <matplotlib.image.AxesImage at 0x1ef0fca4da0>
```

(그림 5) 데이터 이미지 확인

Matplotlib 를 통해 그래프처럼 나타내어 이미지를 볼 수 있다. 가로 세로 28 픽셀인 것을 확인할 수 있으며 앞서 레이블이 5 였던 것을 확인해 사람의 눈으로는 '5'라는 숫자임을 알 수 있다.

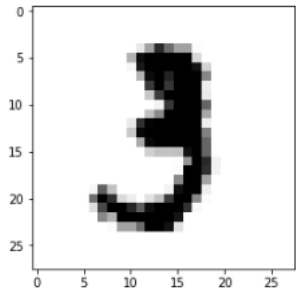
간단하게 실제 학습이 되는지 체크하기 위해 10 개의 데이터셋으로 시도를 해보았다. 10 개의 데이터셋에는 레이블에 '7'이라는 숫자가 있고 run 해본 결과

```
Out[35]: array([[0.02962178],
                [0.01421374],
                [0.03502048],
                [0.16256992],
                [0.03832804],
                [0.06554464],
                [0.01609243],
                [0.71125206],
                [0.09725413],
                [0.02311628]])
```

(그림 6) 10 개의 데이터셋 결과

10 개의 데이터셋 만으로도 유의미한 결과를 얻을 수 있었다. (그림 6)의 배열 위에서부터 0~9 까지에 대한 정확성이 얼마나 되는지 소수점으로 나타나져있다. 1 에 가까운 수치를 띌수록 해당 인덱스에 대한 숫자라고 판단하는 것이다. 0.71125206 이라는 수치가 제일 높고 8 번째(0,1,2…,9)에 위치해 있으므로 이 신경망이 '7'이라는 숫자를 판별해낸 것이다.

이번에는 100 개의 데이터셋을 사용하여 알아보겠다. 100 개의 데이터셋 중 11 번째 레이블인 '3'을 판별할 것이다.



```
Out[118]: array([[0.0688951 ],
                 [0.11142754],
                 [0.0088621 ],
                 [0.78092344],
                 [0.00185799],
                 [0.00620651],
                 [0.02370831],
                 [0.02869089],
                 [0.18533335],
                 [0.00652611]])
```

(그림 7) 100 개의 데이터셋 결과

그 결과 0.78092344 라는 수치가 제일 높은 것으로 이것도 마찬가지로 3 이라는 것을 알아맞췄다. 두번째로 높은 수치는 8 인데 이는 '3'과 '8'이 닮아 있기 때문에 나온 것이 아닌가 추측한다. 물론 10 개의 데이터셋도 된 경우를 가져온 것이지 확실히 100 개의 데이터셋보다는 정확도가 떨어지는 것이 맞다. 임의의 레이블들로 여러 번 실험해본 결과 데이터셋이 많으면 많을수록 정확도가 높아진다. 적어도 데이터를 학습시킨다는 목표는 충족시킨 것 같다. 이번 실험으로 정확도를 측정하는 성능까지 뽑아내고 싶었으나, 아직 진행단계에 있어 미치지 못하였다. 그 점에 대해서는 논의에서 이야기할 것이다.

그리고 다른 한가지에서 재미난 점은 학습률이 낮으면 오히려 판별력이 더 높아지는 양상을 띄는 점이었다. 학습률은 0 에서부터 1 까지로 두는데, 보통 0. 에서 0.3 까지가 가장 많은 성공 횟수를 보였다.

5. 결론 및 논의

이번 실험에 대해서 이론적인 연구의 단계는 많이 나아갔으나, 실제 개발 구현 단계에 있어서는 그만큼에 미치지 못하였다. 하지만 적어도 인공 신경망에 있어서 중요한 '학습'을 하는 과정은 심심치 않게 성공하는 모습을 보였다. 한 가지 궁금증이 풀리지 않는 점은 은닉 노드의 개수를 조정하는데 어떤 의미가 있는지 알기가 어렵다는 점이었다. 학습률은 조정시 값의 변화가 유추 가능한 방면 은닉층은 어려웠다. 앞서 처음에 인공신경망에 대한 원리를 깨우치자 시작했던 연구가 예상보다 그래도 잦은 성공을 보여 만족스러운 결과가 되지 않았나 느낀다. 물론 라이브러리들을 직접 구현해 모듈화 시켜 사용 했다면 더욱 더 심층적인 연구가 됐지 않을까 싶다. 앞으로의 남은 과제는 판별과 동시에 성능을 알려주는 기능과 더 뛰어난 연산, MNIST 데이터가 아닌 직접 숫자 데이터를 만드는 것이다. 그리고 인공 신경망 같은 경우는 서론에서처럼 여러 분야에서 이용되기 때문에 라즈베리파이에서 구동을 시켜 응용하는 것도 생각하고 있다. 접근하기가 편하고 구축이 쉽기 때문이다. 추후에 숫자가 정확히 학습이 된다면 문자를 벡터화(Vectorization) 시켜서 알맞은 문자의 결과를 가져오는 것도 다음의 연구에 조심스레 염두에 두고 있다.

연구의 한계는 시그모이드 함수가 실제 현업에서 활성화 함수로 그다지 쓰이지 않는다는 사실이다. 보통 ReLU 를 많이 쓰인다고 알려져 있어 ReLU 에 대한 지식도 차후 늘려나갈 계획이다. 그리고 데이터에 회전각이 어느 정도 들어가게 되면 성능이 떨어지는 부분에 대해서도 회전각을 고려한 새로운 학습 데이터를 고려할 필요가 있다.

참고문헌

고창룡(2011), 신경회로망과 퍼지 추론에 의한 필기체 숫자 인식, 한국컴퓨터정보학회 한국컴퓨터정보학회 논문지.

김래현(2014), 인공신경망을 이용한 KOSPI 200 단기예측에 관한 연구, 국민대학교 비즈니스 IT 전문대학원 석사학위 논문.

신정훈(2019), 인공신경망 알고리즘을 활용한 가뭄 취약지역 분석, 성균관대학교 일반 대학원 석사학위논문.

윤진호(2012), 인공신경망을 이용한 머플러의 피로수명 예측에 관한 연구, 부산대학교 대학원 석사학위 논문.

Tariq Rashid, 신경망 첫걸음, 출판지: 한빛미디어, 2017

MNIST 학습 데이터 모음 https://git.io/vySZ1
MNIST 테스터 데이터 모음 https://git.io/vySZP

(그림 1)
https://terms.naver.com/entry.nhn?docId=830581&cid=42344&categoryId=42344