JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 1: 49~57, Jan. 2022

데이터와 인공신경망 능력 계산

이덕균[1] · 박지은[1*]

Calculating Data and Artificial Neural Network Capability

Dokkyun Yi[1] · Jieun Park[1*]

[1*]Professor, Seongsan Liberal Arts College, Daegu University, Daegu, 38453 Korea

요 약

최근 인공지능의 다양한 활용은 기계학습의 딥 인공신경망 구조를 통해 가능해졌으며 인간과 같은 능력을 보여주고 있다. 불행하게도 딥 구조의 인공신경망은 아직 정확한 해석이 이루어지고 있지 못하고 있다. 이러한 부분은 인공지능에 대한 불안감과 거부감으로 작용하고 있다. 우리는 이러한 문제 중에서 인공신경망의 능력 부분을 해결한다. 인공신경망 구조의 크기를 계산하고, 그 인공신경망이 처리할 수 있는 데이터의 크기를 계산해 본다. 계산의 방법은 수학에서 쓰이는 군의 방법을 사용하여 데이터와 인공신경망의 크기를 군의 구조와 크기를 알 수 있는 Order를 이용하여 계산한다. 이를 통하여 인공신경망의 능력을 알 수 있으며, 인공지능에 대한 불안감을 해소할 수 있다. 수치적 실험을 통하여 데이터의 크기와 딥 인공신경망을 계산하고 이를 검증한다.

ABSTRACT

Recently, various uses of artificial intelligence have been made possible through the deep artificial neural network structure of machine learning, demonstrating human-like capabilities. Unfortunately, the deep structure of the artificial neural network has not yet been accurately interpreted. This part is acting as anxiety and rejection of artificial intelligence. Among these problems, we solve the capability part of artificial neural networks. Calculate the size of the artificial neural network structure and calculate the size of data that the artificial neural network can process. The calculation method uses the group method used in mathematics to calculate the size of data and artificial neural networks using an order that can know the structure and size of the group. Through this, it is possible to know the capabilities of artificial neural networks, and to relieve anxiety about artificial intelligence. The size of the data and the deep artificial neural network are calculated and verified through numerical experiments.

키워드 : 설명 가능한 인공지능, 인공지능 능력, 데이터 크기, 인공신경망 구조, 해석

Keywords : Explainable artificial intelligence, Artificial intelligence capabilities, Data size, Artificial neural network structure, Analysis

Received 1 November 2021, **Revised** 11 November 2021, **Accepted** 20 December 2021

*** Corresponding Author** Jieun Park(E-mail:writer2yah@daegu.ac.kr, Tel:+82-53-850-4597)
Professor, Seongsan Liberal Arts College, Daegu University, Daegu, 38453 Korea

Open Access http://doi.org/10.6109/jkiice.2022.26.1.49 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

기계학습은 다양한 분야에 널리 쓰이며 바둑, 무인 자동차, ... 등 활용 분야 또한 커지고 있다. 인간의 능력을 뛰어넘는 일이 가능해졌다. 이처럼 기계학습의 활용범위는 커지고 있으며, 그에 따른 학습에 필요한 데이터양도 많아지고, 데이터의 크기도 커지고, 또한 복잡한 문제를 해결해야 한다.

이 문제를 해결하기 위해서 기계학습은 복잡한 인공신경망을 요구하고, 인공신경망의 구조가 커지고, 컴퓨터 성능은 높은 사양을 요구한다. 지금의 해결 방법은 딥 구조의 인공신경망과[1-4], 많은 학습 데이터를 통하여 해결하고 있다. 그러나 이러한 딥 구조는 인공신경망의 구조에서 이루어지는 계산 과정을 개발자는 모르게 되며, 알지 못하는 계산 과정으로 인공지능에 대한 막연한 두려움이 커지고 있다. 인공지능으로 유명한 Stanford 앤드류 응 교수는 앞으로의 인공지능의 발전은 '설명 가능한 인공지능'이어야 한다[5]고 이야기하였다. 설명 가능한 인공지능에 대한 분석 및 해석은 여러 방면으로 연구되고 있다[6-8]. 그러나 이러한 논문들은 함수(기계학습)의 연속상황에서 함숫값의 범위를 계산하는 단계일 뿐으로, 구체적인 데이터에 대한 값을 계산한 것은 아니다. 데이터의 크기와 데이터의 양에 따른 정형화된 계산 그리고 이를 처리할 수 있는 기계의 능력을 알고자 한다. 즉 정형화(수학에서 쓰이는 군의 방법)된 방법을 통하여 데이터와 인공신경망을 계산한다. 그리고 이 둘 사이의 관계를 통하여 데이터를 처리할 수 있는 인공신경망의 능력이 설명 가능하다.

우리는 이 논문을 통해서 디지털 데이터의 크기, 용량에 따라서 데이터를 처리하기 위한 인공 신경의 개수를 알아보고자 한다. 가장 기본이 되는 군을 가지고 데이터와 인공신경망을 계산하고, 군으로 계산 가능한 부분까지 알아본다. 이를 바탕으로 최소의 인공신경망 구조의 디자인 또한 가능할 것이다. 딥 구조에 따라 성능의 한계와 요구 범위 등을 알려줌으로써 인공신경망 구조 연구에 유용하게 쓰일 것이다. 또한 인공신경망을 이용한 응용문제(자율주행 자동차, 음성인식 등)를 해결하고자 하는 사람들에게 인공신경망의 크기, 구조에 기준이 될 것이다. 짧은 시간에 많은 계산이 이루어져야 하는 상황에서 아주 유용하게 이용될 것이다.

이 논문에서는 디지털 데이터의 크기와 인공신경망 크기를 수학의 군을 이용하여 서로의 연관성을 설명한다. 본문에서 더 자세히 설명하겠지만, 우리는 군 $<1>_2 = Z_2$을 이용한다. 디지털 데이터는 0과 1로 표현 가능하다는 것을 알고 있다. 수학에서 쓰이는 군 $<1>_2 = Z_2$이며, 원소는 0과 1로 이루어진 집합이다[9]. 이 군을 이용하여 디지털 데이터를 계산하고 인공신경망도 계산한다. 구조의 변화를 통하여 인공지능의 능력이 향상된 잘 알려진 방법은 CNN(Convolutional Neural Network) 방법이 있다[10]. Convolution의 효과는 인공신경망 안의 구조에 따른 기하 구조의 해석을 요구하기 때문에 양을 계산하는 군의 방법만으로 설명하기에는 어려움이 있다. 군의 방법만을 사용하는 이 논문에서는 계산하지 않는다. 그러나 CNN의 효과는 너무나 유용함으로 실험상에 추가하여 비교 실험한다[4].

논문의 구성은 2장에서는 이 논문을 이해하기 위한 이론적 배경을 설명한다. 3장에서는 데이터의 크기에 따른 군으로의 계산과 인공신경망 구조에 따른 군의 계산 방법을 설명한다. 데이터의 크기와 이를 처리하기 위한 인공신경망의 크기에 대한 가설을 설명한다. 4장에서는 가설을 수치적 실험을 통하여 증명한다.

Ⅱ. 이론적 배경

기계학습의 기본은 입력 데이터와 출력 데이터로 구분된다. 이 논문에서는 입력 데이터는 x로 표기하고 출력 데이터는 y로 표기하기로 한다. 여기서 기계를 f라 할 때, 우리는 입력하여 출력 y를 얻는 과정을 수식으로 표현할 수 있다. 즉 $f(x) = y$의 식이 성립하게 된다. 입력 데이터 x의 값이 여러 가지 있다고 할 때 우리는 x에 첨자 i를 추가하여 표시하기로 한다. 즉 x_i로 쓰고 같은 방법으로 출력 데이터 y에도 첨자를 활용하여 y_i로 쓰기로 하자. 그러면 각 첨자에 따라 입력과 결과의 순서쌍을 다음과 같이 만들 수 있으며, (x_i, y_i)로 표시한다. 우리가 학습하고자 하는 데이터의 개수가 l이라 할 때 데이터의 수는 l이다. 따라서 l개의 학습 데이터로부터 우리는 출력 y가 나올 수 있는 기계 f를 만들어야 하는 문제인 것이다. 즉 $f(x_i) = y_i$를 1부터 l까지 만족하는 기계 f를 만들어야 한다.

기계 f는 함수(function)이다. 우리는 활성화(activation)

함수를 sigmoid σ 함수로 이용하여, 기계 f를 다음과 같이 만든다.

$$f(x)=\sigma(wx+b) \quad (1)$$

여기서 w는 wight, b는 bias라 칭한다. 따라서 w와 b를 정함으로 우리는 기계 f를 완성할 수 있다. 기계학습에 대하여 자세한 사항은 논문 및 책을 참조하기 바란다[11]. 최근에는 활성화 함수를 ReLU함수[7]를 많이 사용하고 있으나, 이 논문에서는 sigmoid 함수를 쓰는 것으로 하겠다. w와 b를 정하는 방법은 입력값에 따라 출력값이 나와야 하는 조건, $f(x_i)=y_i$로부터,

$f(x_i)-y_i=\sigma(wx_i+b)-y_i$의 식을 얻는다. 이것을 이용하여, 우리는 비용함수를 정의하는데, 비용함수 C는 다음과 같이 정의한다.

$C=\sum_{i=1}^{l}(\sigma(wx_i+b)-y_i)^2$ 이 비용함수를 최소로 만드는 w와 b를 찾는 것으로 기계학습은 완성된다. 비용함수를 최소로 만드는 순간의 w와 b를 고정하여 기계 f를 활용한다. 학습 데이터 이외의 데이터에서 적용함으로 기계학습이 마무리된다.

예를 들어, l값이 2라 하면, 모르는 변수 w와 b, 두 개, 식 2개이므로 정확한 w와 b를 구할 수 있다. 그러나 l이 1이라 하면 아니면 l이 3 이상이라 하면 정확한 변수 w와 b를 구할 수 없다. 이러한 경우를 우리는 부족방정식이라 이야기하며, 정확한 값을 위하여 다른 조건이 필요하다. 그러나 기계학습은 학습 데이터의 수와 변수 w와 b에 차이가 발생하여도, 비용함수의 최솟값을 찾는 것에는 문제가 없다. 다만 최솟값의 값에는 차이가 발생할 것이다. 즉 비용함수의 최솟값이 정확하게 0이라고 할 수 없다. 이러한 점에 관한 연구는 데이터 해석 및 학습 추론 등에 관한 것으로 더 자세한 내용은 [11,12]의 자료를 참조하기 바란다.

학습 데이터가 커짐으로(l값 커짐) 변수 w와 b의 수 또한 늘려야 한다. 다양한 방법이 있으나 이 논문에서는 우선 가장 기본적인 딥 방법의 모형을 이용하기로 하며, 방법은 다음과 같이 비용함수를 구성한다.

$$C=\sum_{i=1}^{l}\left(\sum_{j=1}^{m}\left(w_j^1\sigma(w_jx_i+b_j)\right)-y_i\right)^2 \quad (2)$$

여기서 변수들을 늘려가는 이 상황을 우리는 row의 변화라 할 것이다. 여기서 m개의 row를 사용한 것이며, w^1는 layer를 통하여 나온 변수이고, w는 layer를 통과하지 않은 다른 변수이다. 이러한 row의 변화만으로는 논리 연산이 부족하다는 것은 1980년대 초에 잘 알려진 사실이며, 이러한 점을 해결하고자 layer의 수를 늘리는 딥 구조의 연산을 수행해야 했다. layer의 수를 늘리는 것은 수학에서 쓰이는 합성함수의 방법을 이용한다. 계산하는 방법은 다음과 같다.

$y_h^1=\sum_{j=1}^{m}\left(w_j^1\sigma(w_jx_i+b_j)\right)$로 정의하고, 이로부터 $y_h^2=\sum_{j=1}^{m}\left(w_j^2\sigma(w_j^{11}y_{hi}^1+b_j^1)\right)$의 방법으로 얼마든지 늘려나갈 수 있다. 이것을 수학에서 쓰이는 Vector의 계산 방법으로 간단하게 기술한다. 우리는 layer와 row의 값에 대하여 정의했으며, 데이터의 개수에 따른 문제 등에 대해서도 알아보았다. 여기에 추가로 데이터의 크기에 따른, 즉 입력 데이터 x의 크기(size)에 따른 딥 learning의 상관관계에 대하여 알아본다.

Ⅲ. 데이터 및 Neural Network의 군 관계

수학에서 사용하는 군에 관하여 기본적인 설명과 이를 이용한 디지털 데이터의 계산과 인공신경망의 구조를 계산한다.

3.1. 수학의 군 계산

$<1>_2=Z_2$, $Z_2=\{0, 1\}$의 원소에 + 연산을 수행하는 것으로써 집합 안에 있는 원소를 2로 나누어 나머지를 모아 놓은 것이다. 즉 2는 2로 나누어 0이므로 2는 다시 0이 되고 1은 2로 나누면 0.5이나 정수값을 정의함으로 1이라 생각한다. 그러므로 Z_2 안에서의 연산은 0+1, 1+1, 0+0의 연산이 이루어진다[9]. 이러한 설명을 표현하면 그림 1처럼 나타낼 수 있다.

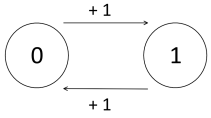

Fig. 1 Structure of $\mathbb{Z}_2$

3.2. 군을 이용한 디지털 데이터 계산

디지털 데이터는 0과 1로 이루어진 신호이므로 Z_2로 표현할 수 있다. 예를 들어 직사각형의 사진은 행렬과 행렬의 원소들로 표현할 수 있다. 사진의 화소 수는 행렬의 행의 수와 열의 수로, 그리고 행렬의 원소가 0과 1의 값이라 할 때, 사진은 행의 수, 열의 수 그리고 원소의 값으로 표현 가능하여 $x=[a_{i,j}]_{m_1\times m_2}$, $a_{i,j}$는 0 또는 1의 값을 갖는다. 우선은 흑백 사진으로 생각하며, 밝은 곳을 1로 어두운 곳을 0으로 생각 할 수 있다. (컬러 사진은 RGB 구조로 레드, 그린, 블루 빛의 3원소로 구성되며, 각 색상은 보통 0부터 255로 2^8으로 쓰이며, 이는 역시나 군의 연산으로 표현할 수 있다). 여기서 원소$a_{i,j}$의 값이 0 또는 1을 갖는다고 했으므로 이것 또한 수학의 군 $<1>_2=Z_2$로 표현할 수 있다. 따라서 $x=[a_{i,j}]_{m_1\times m_2}$로, 이것은다시 $<1>_2\times\cdots\times<1>_2$, 그리고= $\prod^{m_1m_2}<1>_2$로 수식으로 표현할 수 있다. 경우의 수로 계산하면 $2^{m_1m_2}$의 가짓수로 계산되며, 여기서 m_1m_2를 Order라고 이야기할 것이다. 즉 디지털 데이터x의 Order는 m_1m_2이다. 컬러 사진은 각 색상에서 $<1>_2^8\times\cdots\times<1>_2^8=\prod^{m_1m_2}<1>_2^8$로 생각할 수 있다. 여기서 $<1>^8=\prod^{8}<1>_2$이다. 화소에서의 각 경우의 수와 같은 결과를 얻는다.

3.3. 인공신경망과 군 관계

sigmoid σ는 0 또는 1을 표시하는 연속 함수이므로, 학습이 잘 이루어졌다는 가정하에 하나의 σ는 수학의 군 $<1>_2=Z_2$로 표현할 수 있다. 따라서 디지털 데이터와 인공신경망 사이에 다음과 같은 상관관계를 생각할 수 있다. 인공 신경 하나는 활성화 함수 하나로 생각할 수 있고, 활성화 함수 하나는 군 $<1>_2=Z_2$로 생각할 수 있다. 군 $<1>_2=Z_2$는 $a_{i,j}$ 로 생각할 수 있다. 수식으로 표현하면 다음과 같다.

$$\sigma \approx \text{군} <1>_2=Z_2 \approx a_{i,j}.$$

따라서 인공신경망 안에서 신경세포와 같은 역할에 해당하는 활성화 함수 σ 가 0과 1을 표현함으로 군 $<1>_2=Z_2$로 대응할 수 있고 이는 영상에서 하나의 화소와 연관 지을 수 있다.

3.4. 군을 이용한 인공신경망 구조 계산

인공신경망에서는 +연산과 $\times$이 이루어지며, 그 안에서 연산이 이루어질 때 군의 변화가 어떻게 되는지 계산한다. 동일 layer에서의 계산과 layer를 이동하면서 이루어지는 기본 계산을 하고 이를 바탕으로 디지털 데이터를 표현하는 Order와 그것을 학습하기 위한 인공신경망의 Order를 계산한다. 아래 3.4.1부터 3.4.8까지는 활성화 함수를 통하여 나온 값을 군과 대응하여 계산한다.

3.4.1. $\sigma(wy^1)\approx<1>_2$. $\sigma\rightarrow\sigma$: 신경망에서 신경망으로 파라미터 하나 결합

가장 기본적인 변화 없는 연산, w의 값이 변하여도 기본적인 군의 값에는 변화가 없다.

3.4.2.

$\sigma(w^{11}y^{11}+w^{12}y^{11})\approx\sigma((w^{11}+w^{12})y^{11})\approx<1>_2$.

$\sigma\rightarrow\sigma$: 동일 layer에서 +연산 수행 후 layer 변환

같은 layer에서 동일 Neural(y^{11}) 값의 합은 군에는 변화를 주지 못한다.

3.4.3.

$\sigma(w^{11}y^{11}+w^{12}y^{12})\approx<1>_2\times<1>_2$ 또는 $\sigma(w^{11}y^{11}+w^{12}y^{12})\approx<1>_2$. $\sigma,\sigma\rightarrow\sigma$: 서로 다른 layer로부터의 입력 후 +연산 수행

같은 layer에서 다른 신경망 (y^{11},y^{12}) 값의 합은 군에는 변화를 줄 수 있다. 그러나 입력되는 신경망의 수를 늘릴 수는 없다.

3.4.4.

$y^{21}=\sigma(w^{11}y^{11}),y^{22}=\sigma(w^{12}y^{11})$

$\Rightarrow\sigma(w^{21}y^{21}+w^{22}y^{22})\approx<1>_2\times<1>_2$.

$\sigma\rightarrow(\sigma,\sigma)\rightarrow\sigma$: 하나의 입력값이 2번의 layer 변환 과정을 거치는 과정

입력되는 신경망의 수가 다른 값이 들어올 때, 2번 거친 layer에 의해서 군에 변화를 줄 수 있다.

3.4.5.

$y^{21}=\sigma(w^{11}y^{11}),y^{22}=\sigma(w^{12}y^{11}),y^{23}=\sigma(w^{13}y^{11})$

$\Rightarrow\sigma(w^{21}y^{21}+w^{22}y^{22}+w^{23}y^{23})$

$\approx<1>_2\times<1>_2\times<1>_2$. $\sigma\rightarrow(\sigma,\sigma,\sigma)\rightarrow\sigma$: 하나의 입력값이 3개의 row로 입력되어 연산 후 출력되는 과정

2번 거친 layer에 의해서 군의 수는 row의 개수만큼 늘릴 수 있다. 일반적인 경우로 확대해 보자.

3.4.6. layer와 row의 수를 늘린 일반적인 경우 : 일반적인 과정으로 딥 구조에서의 다수의 layer와 row의 경우

W^1을 벡터 (vector, size $1\times m_y$)로 Y를 벡터(vector, size $m_y\times 1$)일 때, 인공신경망을 통과한 값은 $\sigma(W^1Y)$ 상숫값으로 계산된다. 만약 $Y\approx\prod^{m_y}<1>_2$라 하면, $\sigma(W^1Y)$의 값은 $\sigma(W^1Y)\approx\prod^{m_y}<1>_2$의 경우의 값으로 계산된다. 이는 3.4.3의 조건에 의하여 성립한다.

3.4.7. $W^1\rightarrow W^{11}$, 벡터에서 행렬로 변화 : 벡터 계산을 행렬 계산을 위한 변환 그리고 군 계산

W^{11}을 행렬(matrix, size $m_{m_{11}}\times m_y$)라 하고, Y는 벡터(vector, size $m_y\times 1$)로 두자. 그러면 $\sigma(W^{11}Y)$ 는 벡터(size $m_{m_{11}}\times 1$)로 계산된다. 여기서 각 원소별 활성화 $\sigma(W^{11}Y)$ 함수를 수행하는 것으로 계산한다.

만약 $Y\approx\prod^{m_y}<1>_2$이라 하면,

$\sigma(W^{11}Y)\approx\prod^{m_y}<1>_2$ 이다. 단 W^{11} rank $m_{11}>m_y$이어야 한다(rank는 책[9]를 참고하기 바란다). 행렬과 layer의 변화 경우를 계산해 본다.

3.4.8. 행렬과 layer 변화 계산

W^{11}는 행렬로 (matrix, size $m_{m_{11}}\times m_y$), 그리고 Y와 W^{21}는 벡터(vector, size $m_y\times 1$, $1\times m_{11}$각각) 으로 정의한다. $\sigma(W^{11}Y)$ 는 vector (size $m_{m_{11}}\times 1$), 각 원소별 활성화 함수를 수행한다. 만약 $Y\approx\prod^{m_y}<1>_2$이라 하면, $\sigma(W^{21}\sigma(W^{11}Y))\approx\prod^{m_{11}m_y}<1>_2$이다. 단 W^{11} rank $m_{11}>m_y$ 이어야 한다.

3.5. Neural Network Order 계산

기계학습이 잘 이루어졌다는 가정하에 인공신경망이 처리할 수 있는 군의 크기는 최소한 2-layer 이상의 구조에 입력 데이터의 크기만큼의 row 수를 유지하는 것을 필요로 한다. 따라서 우리는 다음과 같은 가설을 세울 수 있다.

가설: Order n의 디지털 데이터를 구분하기 위해서는 적어도 2-layer 구조에 n이상의 Order를 가지는 인공신경망 구조를 이루어야 한다.

가설을 증명하기 위한 수치적 실험을 진행한다.

Ⅳ. 수치적 실험

우리는 Google이 제공하는 Tensor flow와 Adam 최적화 방법을 기본으로 사용한다[3, 13-18]. 모든 실험에서의 학습 강도는 0.01이고 학습 횟수는 200번이다(여러 학습의 결과 오판율(cost값)이 1% 이하인 경우의 학습 횟수는 200이었다). 이 논문의 각 표에서 %의 의미는 학습 데이터 중 숫자만큼의 %로 임의로 선택한 것이다. 선택되지 못한 데이터를 이용하여 정확도를 확인한다.

4.1. O와 X문제

사람이 손으로 쓴 O와 X를 기계학습 시킨다. 데이터의 크기는 64×64 size 500개의 데이터를 가지고 있다. 그림 2는 손으로 쓴 O와 X의 데이터들의 몇 가지 예를 보여준 것이다.

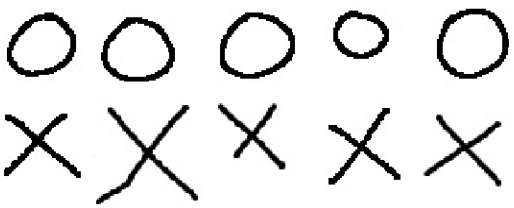

Fig. 2 Example of binary dataset

흑백의 이미지라 가정했을 때, 64×64 size의 흑백 이미지의 경우의 수는 $\prod^{4096}<1>_2$이다. 따라서 신경망 구조의 구조 또한 4096 이상이어야 한다. 표 1은 실험 4.1.1과 4.1.2에 쓰인 layer, row, parameter의 값을 나타낸 것이다.

Table. 1 Initial parameters in 4.1.1 and 4.1.2

4.1.1 Full 1, layer 1, row 4096 $size(W)=2\times 4096$		
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	8194

```
=================================================================
Total params: 8,194
Trainable params: 8,194
Non-trainable params: 0
```

4.1.2 Conv1, Full 1, layer 2, row 1024
$size(W) = 64 \times 1024$, Conv1(3×3 filter 1, 64×2)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 1)         10
dense_1 (Dense)              (None, 2)                 8194
=================================================================
Total params: 8,204
Trainable params: 8,204
Non-trainable params: 0
```

4.1.1. 실험 출력값이 O와 X구분이므로, 이진 분류

출력은 두 가지이며 디지털 데이터의 order가 64×64=4096이므로 row의 개수를 4096으로 하였다. 표 2는 4.1.1과 4.1.2의 실험의 결과를 나타낸 것이다.

Table. 2 Results 4.1.1 and 4.1.2

	20% (400, 1600)		40% (800, 1200)	
	cost	acc	cost	acc
4.1.1	1.0×10^{-5}	1.0	8.0×10^{-6}	1.0
4.1.2	3.0×10^{-5}	1.0	3.9×10^{-7}	1.0
	60%(1200, 800)		80%(1600, 400)	
	cost	acc	cost	acc
4.1.1	2.9×10^{-6}	1.0	1.1×10^{-6}	1.0
4.1.2	2.3×10^{-7}	1.0	1.8×10^{-6}	1.0

cost 학습의 정도는 대부분 0.000001 이하의 값이 나왔으며, acc 다른 학습에 사용되지 않은 데이터를 실험했을 때는 100의 정확도를 보였다.

4.1.2. 실험 출력값이 O와 X구분의 동일 실험에서, CNN 방법을 사용하여 layer의 수를 2로 하고 row의 수를 줄여 실험

64×64 size의 반에 해당하는 32×32 size에 해당하는 order 1024로 실험하였다. 실험 결과는 표 2를 통하여 확인하면 500개의 손으로 쓴 O와 X의 20%, 40%, 60%, 80%의 각 선택의 모든 학습에서 99% 이상의 적중률을 보였다.

두 실험은 가장 단순한 구조의 실험값으로 우리가 가설에서 이야기한 것처럼 디지털 데이터의 order와 인공신경망의 order가 일치하더라도 구분해 낼 수 있음을 실험을 통하여 확인하였다.

4.2. 0부터 9까지의 수 분류

데이터의 크기가 28×28의 영상 실험으로 70,000개의 손으로 쓴 데이터를 가지고 실험한다. 그림 3은 손으로 쓴 0부터 9까지의 수 중 몇 가지 예를 제시한 것이다.

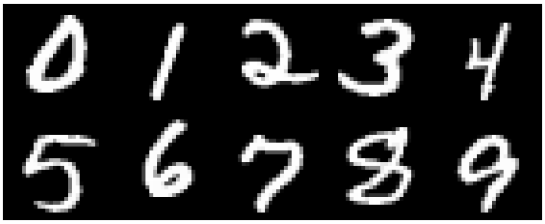

Fig. 3 Example of MNIST(Modified National Institute of Standards and Technology) dataset

10가지로 구분하여 출력해야 하므로 최소 단위는 10×784이어야 한다. 4.2.1이 가장 기본으로 1-layer로 실험하였다. 4.2.2는 이전과 같은 2-layer로 CNN 방법을 사용하였다. 표 3은 4.2.1과 4.2.2 실험의 layer, row, parameter의 값을 나타낸 것이다.

Table. 3 Initial parameters in 4.2.1 and 4.2.2

4.2.1 Full 1, layer 1, row 784
$size(W) = 10 \times 784$

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

4.2.2 Conv1, Full 1, layer 2, row 784
$size(W) = 10 \times 784$, Conv1(3×3 filter 1)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 1)         10
_________________________________________________________________
dense_1 (Dense)              (None, 10)                7850
=================================================================
Total params: 7,860
Trainable params: 7,860
Non-trainable params: 0
```

Table. 4 Results 4.2.1 and 4.2.2

	20% (14000, 56000)		40% (28000, 42000)	
	cost	acc	cost	acc
4.2.1	0.0677 (1.2046)	0.9773 (0.8772)	0.2231 (0.7295)	0.9398 (0.8887)
4.2.2	0.0203 (2.5423)	0.9927 (0.8748)	0.1603 (0.5354)	0.9524 (0.8998)

	60% (42000, 28000)		80% (56000, 14000)	
	cost	acc	cost	acc
4.2.1	0.2720 (0.5857)	0.9285 (0.8959)	0.2946 (0.5213)	0.9233 (0.9050)
4.2.2	0.1905 (0.4171)	0.9443 (0.9084)	0.1458 (0.2687)	0.9537 (0.9349)

표 4는 4.2.1과 4.2.2의 실험의 결과를 나타낸 것이다. cost 학습의 정도는 대부분 0.1 이하의 값이 나왔으며, acc 다른 학습에 사용되지 않은 데이터를 실험했을 때는 0.95의 정확도를 보였다.

출력값이 10가지이므로 다소 정확도가 떨어지는 결과를 얻었으나, 각 경우마다 약 90%의 정확도를 보였다. 흑백의 이미지로 사진이 구분 가능한 것은 우리가 추론한 영상을 구분하는 order와 인공신경망의 order가 일치한다.

4.3. 컬러 이미지 데이터

Size 32×32×3의 영상과 컬러는 256 level 이미지로 개와 고양이를 구분하는 실험을 시행한다. 그림 4는 실험에 사용한 개와 고양이의 사진 중 일부를 가지고 온 것이다.

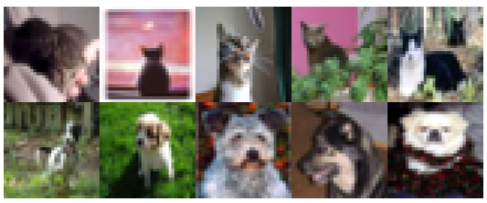

Fig. 4 Example of RGB CAT&DOG image in CIFAR-10

컬러 사진으로 각 색감의 level이 256 즉 2^8이다. 따라서 계산되는 order는 $8^{32*32*3}(2^{8^{32*32}})^3$이다. 이전에 기계학습 방법과 동일하게 3072=32*32*3의 입력값으로 인공신경망을 구축하여 단계별로 layer 또는 row를 늘려가며 실험하였다. 표 5는 4.3.1 ~ 4.3.7 실험의 layer, row, parameter의 값을 나타낸 것이다.

Table. 5 Initial parameters in 4.3.1,... and 4.3.7

4.3.1 Full 1, layer 1, row , $size(W) = 3072 \times 2$

```
Layer (type)                 Output Shape              Param #
=================================================================
(Dense)                      (None, 2)                 6146
=================================================================
Total params: 6,146
Trainable params: 6,146
Non-trainable params: 0
```

dense

4.3.2 Full 2, layer 2, row , $size(W) = 3072 \times 512, 512 \times 2$

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               1573376
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 1,574,402
Trainable params: 1,574,402
Non-trainable params: 0
```

4.3.3 Full 3, layer 3, row , $size(W) = 3072 \times 1024, 1024 \times 2, 512 \times 2$

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 1024)              3146752
_________________________________________________________________
dense_5 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,672,578
Trainable params: 3,672,578
Non-trainable params: 0
```

4.3.4 Full 4, layer 4, row , $size(W) = 3072 \times 1024, 1024 \times 512, 512 \times 128, 128 \times 2$

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 1024)              3146752
_________________________________________________________________
dense_14 (Dense)             (None, 512)               524800
_________________________________________________________________
dense_15 (Dense)             (None, 128)               65664
_________________________________________________________________
dense_16 (Dense)             (None, 2)                 258
=================================================================
Total params: 3,737,474
Trainable params: 3,737,474
Non-trainable params: 0
```

4.3.5 Conv1, Full 1, layer 1, row , $size(W) = $Conv1($3 \times 3$ filter 1, 64×2)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 1)         28
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 2050
=================================================================
Total params: 2,078
Trainable params: 2,078
Non-trainable params: 0
```

4.3.6 Conv2, Full 1, layer 2, row , $size(W) =$ Conv1(3×3 filter 1, 64×2)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 3)         84
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 3)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 1)         28
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 514
=================================================================
Total params: 626
Trainable params: 626
Non-trainable params: 0
```

4.3.7 Conv4, Full 2

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 8)         608
```

```
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 8)         32
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 16)        1168
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 16)        64
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 32)          4640
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 32)          128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 32)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 4, 64)          18496
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dense_12 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_13 (Dense)             (None, 2)                 66
=================================================================
Total params: 27,282
Trainable params: 27,170
Non-trainable params: 112
```

Table. 6 Results 4.3.1,... and 4.3.7

training data rate	40% (4800, 7200)		60% (7200, 4800)		80% (9600, 2400)	
	cost	acc	cost	acc	cost	acc
4.3.1	0.6672 (1.0637)	0.6964 0.5644	0.7831 (1.0913)	0.6509 (0.5629)	0.8222 (1.4022)	0.6440 (0.5379)
4.3.2	0.7153 (0.7219)	0.5160 (0.5008)	0.7202 (0.8050)	0.5205 (0.5058)	0.7295 (0.7763)	0.5158 (0.4925)
4.3.3	0.6991 (0.6941)	0.5177 (0.4992)	0.7161 (0.7702)	0.5179 (0.5060)	0.7252 (0.7207)	0.5116 (0.4925)
4.3.4	0.6978 (0.6941)	0.5216 (0.4992)	0.6996 (0.7010)	0.5159 (0.4940)	0.6997 (0.7122)	0.5156 (0.4925)
4.3.5	0.5295 (1.1734)	0.6981 (0.5540)	0.6929 (0.6934)	0.5173 (0.4940)	0.6931 (0.6947)	0.5104 (0.4925)
4.3.6	0.6930 (0.6934)	0.5133 (0.5008)	0.6597 (0.7399)	0.5666 (0.5327)	0.6105 (0.6578)	0.6698 (0.6317)
4.3.7	0.0024 (3.5652)	0.9989 (0.6774)	0.0064 (3.1161)	0.9977 (0.7371)	0.0074 (2.3207)	0.9975 (0.7275)

표 6은 4.3.1 ~ 4.3.7의 실험의 결과를 나타낸 것이다.

실험 4.3.1은 입력되는 데이터의 크기인 3072로 인공신경망을 구축하여 가장 기본이 되는 출력값을 실험하였다. 우리의 예측대로 인공신경망의 크기는 입력되는 컬러 이미지를 구분하기에 턱없이 부족한 값이다. 데이터의 선택 방법을 40% 60% 80%로 변화하며 실험하여도, 50%를 넘지 못함이다. 또한 layer를 4회 통과하여도 인공신경망의 계산 방법 3.4.3과 3.4.4 등에 의하여 군의 값이 줄 수 있다. 이러한 이유와 row의 값이 layer를 통과할수록 줄어드는 이유로 인하여 군의 감소로 이어진다. 실험 4.3.2부터 4.3.5까지의 결과가 더 나아지지 못하는 이유이다. 반면에 CNN의 방법을 사용함으로 layer에 따른 인위적인 인공 신경 합성이 발생하고 그것으로 인하여 군의 계산 방법과 다르다. CNN을 통한 layer의 변화로 학습률의 성과는 CNN을 사용하지 않은 방법에 비하여 20% 이상의 상승효과가 있었다. CNN에서 Convolution의 역할은 특정 부분의 연결이므로 어떻게 계산되는지는 다음 논문에서 다루고자 한다. 그러나 Full 연결된 인공신경망에서 '일부분의 부분을 모아 계산하는 방법과 CNN 방법 안에서 쓰이는 방법이 기계학습에 효과적이다'라는 것은 기존의 논문[10]으로부터 잘 알려진 사실이다. 계산 과정을 통하여 이 사실은 다시 증명되었다.

Ⅴ. 결 론

수학에서 정의된 군을 이용하여 디지털 데이터의 크기를 계산하고, 데이터를 기계학습 할 때 필요로 하는 인공신경망의 크기를 알아보고자 하였다. 기본적인 흑백 데이터에서 계산되는 군의 order로 실험 4.1과 4.2를 통하여 우리는 우리의 가설을 확인하였다. 컬러 데이터에서는 RGB(레드, 그린, 블루)의 혼합으로 판단되는 색의 메커니즘을 우리가 군으로 아직 표현할 수 없어서 계산상에 다소 차이가 있었다. 그러나 데이터의 계산 order에 상응하는 인공신경망의 order를 사용하면 RGB 혼합의 컬러 이미지의 구분도 가능하다. 즉 실험 4.3.7을 통하여 40%의 경우 cost 값이 0.0024, 그리고 정확도를 나타내는 acc 0.9989의 결론을 통하여 RGB컬러 이미지도 구분할 수 있다. 그러나 정확한 해석을 위해서는 딥 구조의 CNN과 같은 인공신경망에 대한 기하학적인 요소를 설명할 수 있는 수학적 요소의 연구가 필요하겠다. 이것은 군의 표현만으로 부족하다는 것이다. 군에서 확장된, '환 구조를 이용한 추가 연구가 필요하다'라고 생각한다(환: 집합에 두 가지 연산자를 가지고 있는 구조 [9] 참조). 딥 learning과 Graph 이론 그리고 여기에 기하에 따른 신경망을 설명할 수 있는 수학의 방법이 더 요구된다. 이것은 앞으로의 연구를 통하여 해결해야 하는 내용이라 하겠다. 우리가 이 논문에서 이야기한 기계학습이 디지털 데이터를 학습하는 데 있어서, 최소한에 필요한 인공신경망의 구조 및 크기에 대한 기본적인 설명은 이루어졌다. Stanford 앤드류 응 교수[5]가 앞으로 필

요로 하는 설명 가능한 인공지능을 위한 하나의 방법을 우리가 제시한다.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF-2017R1E1A 1A03070311)

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas: NV, pp. 770-778, 2016.

[2] I. Goodfellow, Y. Bengio, and A. Courville, “Regularization for deep learning,” in *Deep Learning*, Cambridge : MIT Press, 2016.

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 618-626, 2017.

[4] M. W. Huang, C. W. Chen, W. C. Lin, S. W. Ke, and C. F. Tsai, “Svm and svm ensembles in breast cancer prediction,” *PLoS ONE*, vol. 12, no. 1, pp. 1-14, Jan. 2017.

[5] A. Ng. The most frequently themes(AItimes) [Internet]. Available: http://www.aitimes.com/news/articleView.html ?idxno=131542.

[6] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, “Exploring Generalization in Deep Learning,” in *the 31st Conference on Neural Information Processing Systems (NIPS)*, Long Beach: CA, 2017.

[7] Machine Learning Mastery. A Gentle Introduction to the Rectified Linear Unit (ReLU) [Internet]. Available: https:// machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/?nowprocket=1.

[8] D. X. Zhou, “Deep distributed convolutional neural networks: Universality,” *Analysis and Applications*, vol. 16, no. 6, pp. 895-919, 2018.

[9] L. Serge, *Algebra*, New York, NY: Springer-Verlag, 2002.

[10] A. Krizhevsky, H. Suskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, Jun. 2017.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, NY: Springer-Verlag, 2006.

[12] M. Kirby, *Geometric Data Analysis*, New York, NY: Wiley-Interscience, 2001.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278 - 2324, 1998.

[14] G. Montavon, S. Bach, A. Binder, W. Samek, and K. Müller, “Explaining nonlinear classification decisions with deep taylor decomposition,” *Pattern Recognition*, vol. 65, pp. 211-222, May. 2017.

[15] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*, Cambridge: The MIT Press, 2019.

[16] H. Zulkifli, Understanding Learning Rates and How It Improves Performance in Deep Learning. Towards Data Science [Internet]. Available: https://towardsdatascience.com/ understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

[17] S. Lau. Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning. Towards Data Science [Internet]. Available: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1.

[18] G. Aurélien, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media Inc, 2017.

이덕균(Dokkyun Yi)

한국과학기술원 수학과 이학박사
現, 대구대학교 성산교양대학 부교수
※관심분야 : 수치해석, 이미지처리, 인공지능

박지은(Jieun Park)

이화여자대학교 과학교육학과 이학박사
現, 대구대학교 성산교양대학 부교수
※관심분야 : 문제해결력, 표상, 인공지능