



## 프로젝트 주제

다양한 ANN을 이용한 손글씨 숫자 인식 알고리즘

2416 채지민 [cjm\\_0112@naver.com](mailto:cjm_0112@naver.com)

2509 오경준 [frankok@naver.com](mailto:frankok@naver.com)

### 초록

ANN(Artificial Neural Network, 인공 신경망)은 우리 뇌를 구성하고 있는 뉴런을 모방한 것으로, 우리가 알고리즘을 찾기 어려운 문제를 해결하는 데 잘 사용될 수 있다. ANN에는 다양한 Topology가 있는데, 우리는 가장 간단한 FF(Feed Forward)를 대상으로 프로젝트를 진행하였다. 우리 프로젝트의 주제는 숫자 인식 알고리즘에서 최적의 활성화함수를 찾는 것이다. 활성화함수는 ANN에서 Node 사이의 신호를 변환하는 함수를 말하는데, 이 활성화함수가 ANN의 학습을 가능하게 하는 핵심 요소인만큼, 우리는 프로젝트가 ANN의 학습과 관련해 큰 의미가 있다고 생각했다. 또한, ANN의 학습의 시작 단계인 숫자 인식 신경망을 친구들에게 소개함으로써 친구들의 ANN에 대한 관심을 이끌어내려 하였다. 또한, 우리는 ANN을 주로 구현하는 언어인 python의 실행속도보다 C++의 실행속도가 월등히 빠르다는 점에 주목했다. 빠른 실행속도는 같은 시간 내 더 많은 양의 데이터를 학습할 수 있다는 것을 의미하므로, 우리는 C++을 통한 ANN의 구현 또한 의미가 있다고 판단하였다.

주제어: ANN, 활성 함수



♡ 주요 참고문헌: 신경망 첫걸음 / 타리크 라시드

♡ 주요 알고리즘: Artificial Neural Network (인공 신경망)

## I. 프로그램 개발의 동기 및 목적

2 이 글씨는 그림판으로 직접 쓴 숫자이다. 조금이라도 교육을 받은 사람에게 이 숫자를 읽어보라고 하면 모두 2라고 대답할 수 있다. 그만큼 정상적인 손글씨로 쓴 숫자를 분간하는 일은 정말 어려운 작업이 아니다. 하지만 이는 사람에게만 해달될 뿐, 컴퓨터에게는 아니다. 컴퓨터의 입장에서는 2와 2는 그저 int형 데이터와 그림일뿐이고 아무런 연관이 없다. 2라는 손글씨를 인식하기 위해서 알고리즘을 짜기란 상당히 어려운 작업이다. 왜냐하면 사람마다 손글씨가 다르기 때문에 명확한 규칙성을 찾기가 힘들고 3과 8처럼 비슷하게 생긴 다른 숫자 등 수많은 예외들이 존재하기 때문이다. 그래서 이러한 문제들을 해결하기 위해 '인간의 뇌처럼 컴퓨터를 '학습'시킨다.'라는 인공신경망을 도입하게 되었다. 우리도 이 인공신경망을 통해 컴퓨터를 학습시켜 손글씨 숫자 인식 프로그램을 만들 계획이며 기존의 방법에서 더 발전시켜 시그모이드 함수뿐만 아니라 다양한 활성화 함수에 대해서도 알고리즘을 만들어서 가장 적합한 활성화 함수를 찾는 것을 목표로 하였다.

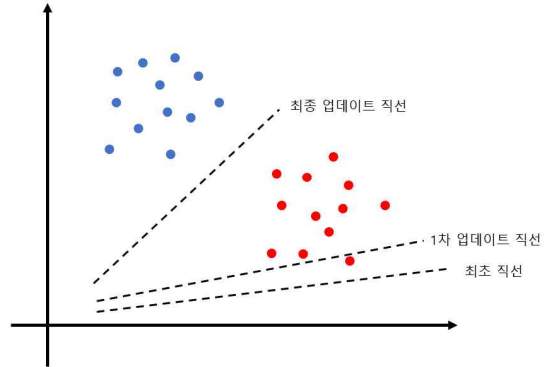
## II. 프로그램 소개



### Ⅲ. 인공지능 소개

#### 1. 오차분석법

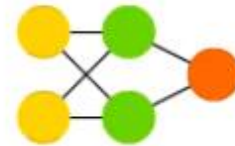
그래프에 직선을 하나 그려서 2개의 그룹으로 분류한다고 하자. 컴퓨터를 통해서 분류를 하기 위해서는 학습을 통해 기울기를 반복적으로 조금씩 바꿔나가는 방법으로 분류를 할 수 있다. 임의의 분할선으로 시작을 한 뒤, 뒤에 입력하는 데이터를 바탕으로 오차를 점점 줄여가는 방향으로 분할선을 수정해간다. 이를 통해 두 그룹의 분류를 위한 직선을 찾을 수 있다.



#### 2. ANN(Artificial Neural Network)

ANN(artificial neural network, 인공신경망)은 기계학습과 인지과학에서 생물학의 신경망에서 영감을 얻은 통계학적 학습 알고리즘이다. 인공신경망은 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화시켜, 문제 해결 능력을 가지는 모델 전반을 가리킨다.

Feed Forward (FF)

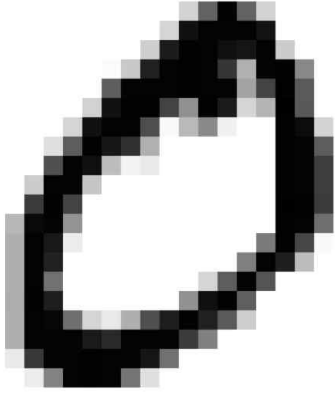


인공신경망에는 교사 신호(정답)의 입력에 의해서 문제에 최적화되어 가는 교사 학습과 교사 신호를 필요로 하지 않는 비교사 학습이 있다. 명확한 해답이 있는 경우에는 교사 학습이, 데이터 클러스터링에는 비교사 학습이 이용되는데, 우리 연구에서는 명확한 답(어느 숫자인지)가 존재하므로, 교사 학습의 알고리즘을 사용하면 된다. ANN은 일반적으로 입력으로부터 값을 계산하는 뉴런 시스템의 상호연결로 표현되고 적응성이 있어 패턴 인식과 같은 기계학습을 수행할 수 있다.

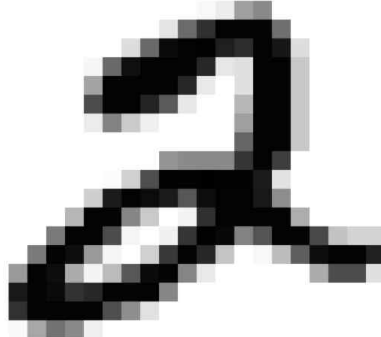
인공 신경망의 구조를 자세히 살펴보자. 인공 신경망은 여러 개의 인공 뉴런으로 구성되어 있고 각각의 뉴런은 여러 개의 입력을 받아 더한 다음 활성화 함수에서 더한 신호를 변환하여 다음 뉴런에 전달한다. 위의 그림처럼 뉴런을 여러 계층에 걸쳐 위치시키고 각각의 뉴런은 직전 계층과 직후 계층에 있는 모든 뉴런들과 상호 연결되어 있는 식으로 표현하면 뉴런을 인공적으로 모델화할 수 있다. 각각의 인공 뉴런들을 노드라고 하는데 노드 간 연결의 강도인 가중치라는 것을 통해 낮은 가중치는 신호를 약화하고 높은 가중치는 신호를 강화한다. 학습을 시킬 때는 오차를 역전파 시켜가면서 분류자를 학습할 때와 동일하게 오차에 대해서 각각의 가중치를 변화시켜나가면 된다.



## IV. 메뉴 구성도



The answer is 0  
Computer answer is 0



The answer is 2  
Computer answer is 2

0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127
0	0	0	0	0	0	0	0	0	0	0	48	238	252	252	252	237	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	232	39	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	253	255	63	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	13	25	100	122	7	0	0	0	0
0	0	0	0	0	0	0	0	32	237	253	252	71	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	38	43	105	255	253	253	253	253	253	174	6	0	0
0	0	0	0	0	0	0	0	0	5	63	197	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	143	247	153
0	0	0	0	0	0	0	103	242	254	254	254	254	254	66	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
99	91	142	155	246	182	155	155	155	155	131	52	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	38	178	252	253	117	65	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	168	242	28	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	93	164	211	250	250	194	15	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	203	229	32	0
0	0	0	0	0	0	0	0	0	0	75	247	143	10	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
25	10	0	0	0	0	0	0	0	0	0	0	0	0	112	252	125	4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	96	205	251	253	205	111	4	0	0

## V. 프로그램 분석 및 주요 알고리즘

### • ANN에서의 신호 전달

각각 2개의 노드를 가지는 3개의 계층으로 구성된 신경망에 대하여 생각을 해보자. 일반적으로 첫 번째 계층을 입력 계층, 두 번째 계층을 은닉 계층, 마지막 계층을 출력 계층이라고 표현한다. 초기 입력값을  $input_1, input_2$ 라고 하고 입력 계층과 은닉 계층 사이의 각각의 가중치 값을  $w_{1,1}, w_{1,2}, w_{2,1}, w_{2,2}$ 라고 하자. 계층 2의 첫 번째 노드에서의 입력의 합을 구하면  $input_1 \times w_{1,1} + input_2 \times w_{2,1}$ 이 되고 두 번째 노드에서의 입력의 합은  $input_1 \times w_{1,2} + input_2 \times w_{2,2}$ 가 된다. 이처럼 노드도 2개밖에 없고 계층도 2개로 매우 단순한 신경망에서는 위와 같이 일일이 식을 적어줄 수 있지만 노드와 계층의 수가 많아지면 많아질수록 코드를 짜면서 실수가 발생할 확률이 높고 컴퓨터도 처리하는데 오래 걸릴 수밖에 없다. 따라서 행렬이라는 것을 이용하여 식을 단순하게 만든다.

$$\begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} \begin{pmatrix} input_1 \\ input_2 \end{pmatrix} = \begin{pmatrix} input_1 \times w_{1,1} + input_2 \times w_{2,1} \\ input_1 \times w_{1,2} + input_2 \times w_{2,2} \end{pmatrix} \text{로 표현을 하면 입력의 합을 쉽게 나타}$$

낼 수 있다.  $W$ 를 가중치의 행렬,  $I$ 를 입력의 행렬,  $X$ 를 입력합의 행렬이라고 하면  $X = WI$ 라는 식 하나면 된다. 계층 2에서의 출력은 입력의 합에 활성화 함수를 적용시켜주기만 하면 된다. 이는 행렬  $X$ 의 각각의 원소에 활성화 함수를 적용시켜주기만 하면 되므로  $O$ 를 최종 결과 행렬이라고 하면  $O = func(X)$ 가 된다.

### • 오차의 역전파

은닉 계층과 출력 계층 사이의 각각의 가중치 값을  $w_{1,1}, w_{1,2}, w_{2,1}, w_{2,2}$ 이라고 하자. 출력 계층에서의 오차를 각각  $e_{output1}, e_{output2}$ 이라고 하면 은닉 계층의 오차는 이 오차를 재조합한 값이 된다. 첫 번째 노드의 오차값  $e_{hidden,1}$ 은  $w_{1,1}, w_{1,2}$ 와 연결되어 있으므로

$$e_{hidden,1} = e_{output1,1} \times \frac{w_{1,1}}{w_{1,1} + w_{2,1}} + e_{output2,1} \times \frac{w_{1,2}}{w_{1,2} + w_{2,2}} \text{이고 두 번째 노드의 오차값도 같}$$

은 방법으로 구할 수 있다. 하지만 식을 이렇게 구성할 경우 행렬로 표현하기 어렵다.

$$error_{hidden} = \begin{pmatrix} \frac{w_{1,1}}{w_{1,1} + w_{2,1}} & \frac{w_{1,2}}{w_{1,2} + w_{2,2}} \\ \frac{w_{2,1}}{w_{2,1} + w_{1,1}} & \frac{w_{2,2}}{w_{2,2} + w_{1,2}} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \text{ 위에서 신호 전달할 때처럼 간단하게 표현하기}$$

어렵다. 따라서 위의 수식에서 핵심인  $e_n$ 과  $w_{ij}$ 의 곱 부분만 남기고 분모 부분의 정규화

인자를 그냥 무시하면  $error_{hidden} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ 로 매우 간단히 표현할 수 있다. 가중치

행렬  $W$ 의 전치행렬을  $W^T$ 로 표현하면  $error_{hidden} = W_{hidden\_output}^T \cdot error_{output}$ 이 된다.



#### • 가중치 업데이트

가중치 업데이트는 '경사 하강법'을 통해서 진행이 된다. 가중치에 따른 오차의 함수를 모르기 때문에 어떤 지점에서 미분값을 구해서 미분값에 따라 가중치를 조금씩 바꿔나가는 것이다. 그렇게 하면 가중치에 따른 오차 함수의 최저점을 구할 수 있고 결국 가중치가 어떨 때 오차가 가장 적은지 구할 수 있다.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (t_n - o_n)^2$$

여러 가지 오차함수가 있지만 일반적으로 제곱오차 방식으로 오차를 계산한다.

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{jk}} \text{sigmoid}(\sum w_{jk} \cdot o_j)$$

미분한 형태가 상대적으로 간단한 시그모이드 함수를 사용하여 표현하였다.

$$\frac{\partial}{\partial x} \text{sigmoid}(x) = \text{sigmoid}(x)(1 - \text{sigmoid}(x))$$

$$\begin{aligned} \therefore \frac{\partial E}{\partial w_{jk}} &= -2(t_k - o_k) \cdot \text{sigmoid}(\sum w_{jk} \cdot o_j)(1 - \text{sigmoid}(\sum w_{jk} \cdot o_j)) \cdot \frac{\partial}{\partial w_{jk}} (\sum w_{jk} \cdot o_j) \\ &= -2e_j \cdot \text{sigmoid}(\sum w_{jk} \cdot o_j)(1 - \text{sigmoid}(\sum w_{jk} \cdot o_j)) \cdot o_j \end{aligned}$$

가중치는 기울기와 반대 방향으로 진행되어야 하고 학습률 인자를 통해 변화의 정도를 조절할 수 있다. 학습률을  $\alpha$ 라고 하면  $new w_{jk} = old w_{jk} - \alpha \times \frac{\partial E}{\partial w_{jk}}$ 가 된다. 위의 식이

매우 복잡해 보이지만 행렬을 이용하면 이를 매우 간단히 나타낼 수 있다.

$$\Delta w_{jk} = \alpha \cdot E_k \cdot O_k (1 - O_k) \cdot O_j^T$$

## VI. 본 연구의 결과에 대한 고찰 및 결론

본 프로젝트에서 FF(Feed Forward)형태의 숫자 인식 ANN에 대해 최적화된 활성화 함수를 찾아내었다. 이를 통해 활성화 함수가 학습에 미치는 영향을 체감할 수 있었다.

본 프로젝트에서는 ANN의 다양한 Topology 중 FF(Feed Forward)에 대해서만 최적화된 활성화 함수를 찾아내었다. 이후, 프로젝트를 더 진행하여 DFF(Deep Feed Forward), RNN(Recurrent Neural Network), LSTM(Long / Short Term Memory)등 다양한 Topology에 대해 최적화된 활성화 함수를 찾아내어, 이를 비교하면 각각의 topology에 최적화된 활성화 함수를 찾아낼 수 있을 것이다. 또한 기존에 있는 데이터 말고도 약간 변화시켜 새로운 데이터를 만들어 낸다면 더 최적화된 ANN을 만들어낼 수도 있을 것이다.



## VIII. 참고문헌

1. 신경망 첫걸음 / 타리크 라시드
2. 딥러닝 첫걸음 / 김성필
3. MNIST data  
<http://yann.lecun.com/exdb/mnist/>
4. Topology of ANN  
<http://www.asimovinstitute.org/neural-network-zoo/>
5. 인공 신경망  
<https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B3%B5%EC%8B%A0%EA%B2%BD%EB%A7%9D>