인공신경망을 이용한 숫자인식에 관한 연구

전민혁*, 김병욱*‡
*동국대학교 경주캠퍼스 컴퓨터공학과
e-mail: gur6289@naver.com, bwkim@dongguk.ac.kr

A Study on Numerical Recognition Using Artificial Neural Network

Min-Hyeok Jun*, Byoung-Wook Kim*‡
*Dept of Computer Engineering, Dongguk University-Gyeongju

요 약

인공지능이 정형화된 수치 데이터뿐만 아니라 비정형 데이터까지도 인식해야하는 시대가 왔다. 보안 분야 이외에도 사회 전반에서 숫자 인식을 활용하고 점차 확대되고 있다. 숫자인식을 위해 인공신경망을 이용하였다. 인공신경망은 입력 층, 중간 층, 출력 층으로 이루어져 있다. 각 층은 노드와 노드들을 연결하는 가중치로 구성되어 있다. data set을 입력 값으로 하여 각각의 가중치를 곱한다. 오차역전파법을 이용하여 가중치 값을 갱신한다. 갱신하는 과정에서 학습률과 가중치 조정을 통해 결과 값의 정확도를 연구한다. 궁극적으로 학습된 data set과 인공신경망 알고리즘을 이용하여 손 글씨로 된 숫자를 인식한다. 실험에서 학습률과 중간층의 노드 개수를 조정하여 인식률을 높여간다.

1. 서론

최근 4차 산업으로 인공지능에 많은 관심이 주목되고 있다. 사회적 기대에 부응하는 인공지능이 되기 위해 많은 기술적 요소들이 필요하다. 이제는 인공지능이 정형화된 수치 데이터뿐만 아니라 비정형 데이터까지도 인식해야하는 시대가 왔다. 보안 분야 이외에도 사회 전반에서 숫자 인식을 활용하고 점차 확대되고 있다. 본 논문에서는 손 글씨로 된 숫자들을 3계층으로 된 인공신경망을 통해서 학습 시킨다. 또 학습된 data를 가지고 이후에 손 글씨로 된 숫자들을 인공신경망 알고리즘을 통해 인식시키는 과정을 연구한다.

2. 인공신경망

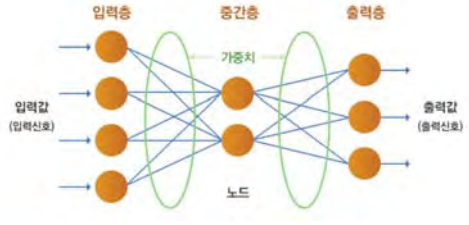
2.1 인공신경망

인공지능의 한 분야인 인공신경망(ANN)은 생물학(통상 인간)의 뇌 구조(신경망)를 모방하여 모델링한 학습 알고리즘이다. 사람의 뇌는 약 250억 개의 신경세포로 구성 되어있다. 신경세포는 1개의 세포체와 세포체의 돌기인 1개의 축색 및 보통 여러 개의 수상 돌기를 포함 한다. 신경세포들 간의 정보교환은 시냅스라고 부르는 전기적 신호이다. 시냅스는 일정한 세기 이상이 되어야 수상돌기로 전달이 된다. 인공 신경망은 이러한 생물학적 신경세포의 정보처리 및 전달 과정을 모방하여 구현한 것이다.

<표1> 신경망과 인공신경망의 비교

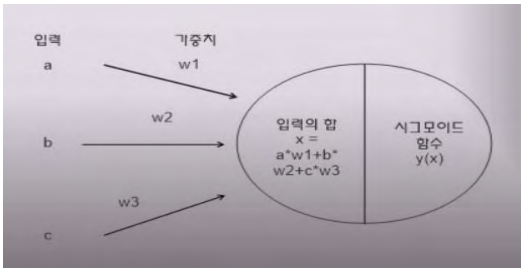
신경망	인공신경망
세포체(cell body)	노드(node)
수상돌기(dendrite)	입력(input)
축색돌기(Axon)	출력(output)
시냅스(synapse)	가중치(weight)

인공 신경망은 입력 신호를 받는 입력 층, 가중치와 활성화 함수를 이용하여 예측 값을 생성하는 중간 층, 출력신호를 출력하는 출력 층으로 구성된다. 각 층은 노드와 각 노드들을 연결하는 가중치로 구성된다. 중간층의 경우 여러 층으로 나뉠 수 있는데, 이를 다층 퍼셉트론이라고 말한다.



(그림 1) 인공신경망 모델

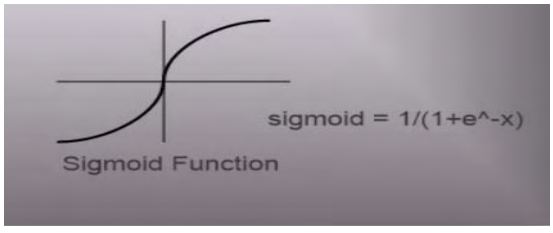
인공신경망을 이용하여 학습하는 경우 각 노드로 입력받는 값과 각각의 연결에 가중치의 값을 이용하여 입력의 합을 먼저 계산한다. 입력의 합을 활성화 함수인 시그모이드 함수를 이용하여 계산하여 예측 값을 도출한다. 이후 중간층에서도 입력 층과 가중치를 계산하여 나온 값을 입력으로 가지고 같은 방법으로 반복적으로 계산한다. 중간층의 개수와 노드의 개수와는 상관없이 동일한 방법으로 계산이 적용된다. 마지막 중간층과 가중치의 계산으로 나온 합과 시그모이드 함수를 계산하면 예측 값 계산을 위한 한번의 학습이 완료된다. 이후 오차의 역전파와 예측 값 계산을 반복 하게 된다.



(그림 2) 노드의 입력과 가중치 계산

2.2 활성화 함수

입력 신호를 받아 특정 분계점을 넘어서는 경우에 출력 신호를 생성해 주는 함수이다. 인공신경망에서 중요한 개념이다. 활성화 함수로는 step function, linear function, threshold logic function, sigmoid function이 있다. 이 중에서 인공신경망은 sigmoid function을 주로 사용한다. 부드러운 형태로 자연스럽고 현실에 가깝기 때문이다. sigmoid function의 그래프와 식은 아래의 그림과 같다.



(그림 3) sigmoid function

2.3 오차역전파법

인공신경망의 학습은 입력 데이터의 값들을 연결해서 모형의 예측 값을 생성하는 과정과 예측 값과 실제 값의 차이를 최소화하기 위해 연결의 가중치를 갱신하는 학습 과정으로 나누어진다. 인공신경망을 통해 출력된 예측 값과 실제 값을 비교하여 가중치를 갱신하고 조정하여 전체 모델을 훈련 시킨다. 이러한 과정을 오차역전파법이라 한다. 대부분의 인공신경망 알고리즘은 가중치 갱신을 위해 경사 하강법을 사용한다. 경사 하강법은 비용함수를 가중치에 대해 편미분한 다음 가중치를 기울기 방향으로 조금씩 시동하는 과정을 반복함으로써 실제값과 예측값이 차이를 최소화하는 가중치를 찾는 학습방법이다.

2.4 인공신경망의 장점

인공신경망은 각각의 노드가 독립적으로 동작하기 때문에 병렬성이 뛰어나다. 학습에서 각각의 노드와 노드에 연결된 가중치가 다르기 때문에 빠르게 학습이 가능하다. 또 많은 연결선에 정보가 분산되어 있어서 몇몇 노드에 문제가 발생해도 큰 영향을 주지 않아 일정수준 오류에 강하다는 장점이 있다.

2.5 인공신경망 학습 노하우

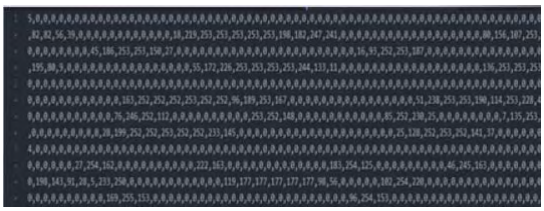
인공신경망 학습을 통해 정확한 예측 값을 도출하기 위해서는 많은 데이터를 준비하고, 많은 파생변수를 생성해야한다. 적절한 데이터의 정제도 학습에 영향을 끼친다. 또 최적의 중간층과 노드 수를 결정해 줘야 한다. 학습률을 조정해서 local minima를 수렴하는 것을 방지해야 한다.

3. 인공신경망을 이용한 숫자인식

3.1 python

인공신경망을 학습시키기 위해 프로그래밍 언어로 python을 이용한다. python은 스크립트 언어로 현재 많은 관심을 받고 있는 언어이다. 데이터처리를 위한 많은 라이브러리를 보유하고 있어 mnist의 많은 data를 학습시키기에 편리하다. 또 그래픽 처리에도 강점을 가지고 있다.

3.2 숫자인식을 위한 data set 준비



(그림 4) MNIST data set

먼저 숫자들을 인식하기 위해 인공신경망에 숫자들을 학습시킨다. 그러기 위해 MNIST를 이용했다. MNIST(Modified National Institute of Standards and Technology database)는 손으로 쓴 숫자들로 이루어진 대형 데이터베이스이다. MNIST의 특징은 흑백 그림들이 28x28 픽셀로 구성되어 있다는 점이다. 785개의 데이터로 하나의 숫자가 이루어지는데 0번 인덱스는 나타내는 숫자

를 의미한다. 1번 인덱스부터 784번 인덱스는 각 픽셀의 흑백 정도에 따라 0에서 255의 값을 가진다.

```
all_values = data_list[1].split(',')
image_array = numpy.asfarray(all_values[1:]).reshape((28,28))
mpl.pyplot.imshow(image_array, cmap = 'Greys', interpolation = 'None')
plt.show()
```

(그림 5) 숫자의 그래픽 출력

MNIST의 숫자를 사람이 알아 볼 수 있도록 그래픽으로 표현해 준다. 첫 번째 행에서 첫 번째 레코드인 data_list[0]을 불러와서 쉼표로 구분하여 분리한다. 함수 reshape((28,28))은 784개의 숫자를 28x28 형태의 정방 행렬로 만들어 준다. matplotlib.pyplot.imshow 함수를 이용해서 시각화 한다.

3.3 신경망 학습

주어진 data set을 가지고 신경망에 학습을 시킨다. 입력 층 노드의 수는 픽셀의 수만큼 생성한다.

```
scaled_input = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
print(scaled_input)
```

(그림 6) 입력 색상 값 조정

data set의 입력 색상 값들의 범위를 0.01에서 1.0사이로 조정해야 한다. 입력 값들을 255로 나누면 0에서 1까지의 범위를 가지게 될 것이다. 그리고 여기에 0.99를 곱하면 범위는 0.0에서 0.99사이의 값으로 변하게 된다. 0.0은 가중치 업데이트를 잃어버리기 때문에 0.01을 더해주어야 한다.

```
onodes = 10
targets = numpy.zeros(onodes) +0.01
targets[int(all_values[0])] = 0.99
```

(그림 7) 출력노드와 값 준비

출력계층의 노드의 수를 10개로 설정한다. 숫자는 0에서 9까지 10개의 레이블을 가지기 때문이다. numpy.zeros() 함수를 이용하여 0으로 채워진 행렬을 생성한다. 이 함수는 매개변수로 행렬의 크기와 형태를 받는다. 여기에서는 최종 출력 계층의 노드의 수를 의미하는 onodes변수를 행렬의 길이로 사용한다. 여기서도 0값을 피하기 위해 0.01을 더해준다. MNIST의 첫 번째 원소를 취한 다음 int()함수를 이용해 그 문자열을 정수로 변환한다.

```
inputs = numpy.array(inputs_list, ndmin = 2).T
targets = numpy.array(targets_list, ndmin = 2).T

hidden_inputs = numpy.dot(self.wih, inputs)
hidden_outputs = self.activation_funtion(hidden_inputs)

final_inputs = numpy.dot(self.who, hidden_outputs)
final_outputs = self.activation_funtion(final_inputs)
```

(그림 8) 신경망 학습

입력 값들을 numpy.array()함수를 이용하여 2차원 배열로 만든다. numpy.dot() 함수를 이용하여 입력 값과 가중치를 행렬 곱을 하여 인공신경망 학습을 효율적으로 진행한다. 하나하나 계산해주어야 할 값들을 행렬곱을 이용하면 한번에 각 노드에 들어오는 입력의 합을 구할 수 있다. 도출된 값을 가지고 시그모이드 함수를 가지고 계산한다. 실험에서 입력 층, 중간층, 출력 층 3개의 층으로 구성했다. 마찬가지로 중간층과 출력 층 사이에서도 가중치와 중간 층 입력 값을 가지고 행렬 곱과 시그모이드 함수 계산을 해준다.

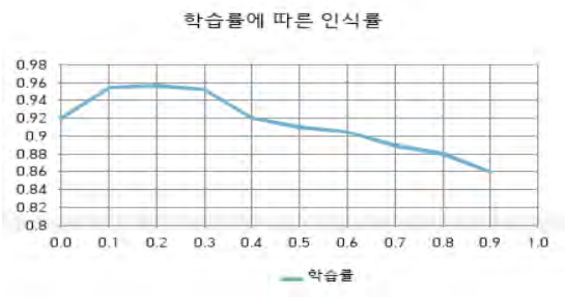
```
self.who += self.lr*numpy.dot((output_errors*final_outputs*(1.0-final_outputs)),numpy.transpose(hidden_outputs))

self.who += self.lr*numpy.dot((output_errors*final_outputs*(1.0-final_outputs)),numpy.transpose(hidden_outputs))
```

(그림 9) 가중치 업데이트

실험에서 초기 가중치는 0.01에서 1사이의 값으로 랜덤하게 할당했다. 여기서 0은 가중치와 입력의 값으로 계산할 때 계산이 없는 것으로 되기 때문에 실험에 적합하지 않다고 판단하여 제외했다. 출력된 오차와 다음 계층으로부터의 시그모이드 함수에서 비롯된 행렬과 이전 계층으로부터의 결과 값을 전치한 것을 행렬 곱을 해준다. 그 다음 학습률을 곱해준다.

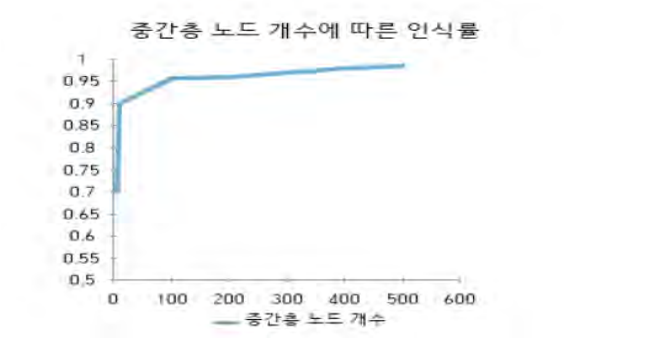
3.4 실험 결과

실험에서 중간층의 노드 개수와 학습률을 조정하여서 인식률을 조정 할 수 있다. 먼저 실행에서 학습률의 경우 0.1에서 0.9까지의 경우에서 0.1에서 0.3의 경우 인식률이 가장 높다는 것을 확인 할 수 있다. 0.1에서의 경우 인식률은 0.952로 나온다. 0.6에서는 0.904의 인식률이 나온다. 자세한 수치는 아래 그림과 같다.



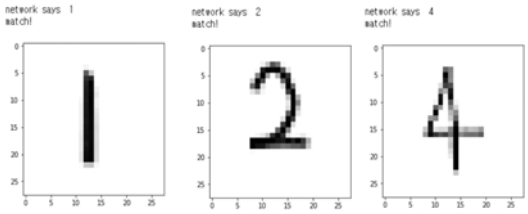
(그림 10) 학습률에 따른 인식률

중간층 노드 개수의 경우는 중간층의 노드 개수가 적을 때에는 결과가 좋지 않다. 노드가 5개일 때 성능은 0.7이다. 노드가 10개일 때는 0.9의 인식률을 보인다. 노드가 100개 이상일 경우 거의 유사한 정확도를 보인다. 자세한 내용은 아래 그림과 같다.



(그림 11) 중간층 개수에 따른 인식률

data set을 가지고 학습을 시켜보고 충분한 학습이 된 이후 여러 이미지를 가지고 실험했다. 노이즈가 심한 경우 인식을 하지 못하는 경우가 많았다.



(그림 12) 숫자인식 일치 결과

뚜렷한 글씨이고, 노이즈가 없는 이미지 일수록 숫자인식이 잘되는 것을 확인 할 수 있다.

4. 결론

앞으로 4차 산업이 발전되고 고도화 되어 갈수록 비정형 데이터의 인식이 중요해 질 것이다. 현재 사회전반에서 사용하는 기술보다 더욱 많은 분야에서 비정형데이터 인식은 활용될 것이다. 이러한 점에서 손 글씨로 적은 숫자인식은 더욱 많은 곳에서 활용 될 수 있다. 향후 학습률과 학습의 횟수 조정을 통해 더욱 인식률을 높여갈 것이다. 또 한글이나 영어에 대한 손 글씨 인식을 추가 할 것이다. 이 연구를 통해 사람들이 보다 안전하고 편리하게 생활할 수 있으리라 기대한다.

5. 참고 문헌

[1] 신경망 첫걸음- 타리크라시드

[2] http://creativeprm.tistory.com/82 [Creative Programmer]

[3] https://giantpark197cm.tistory.com/133?category=784464