

인공신경망의 기초와 소성가공해석에의 응용 II

김영석^{1, #}, 김진재²

1. 경북대학교 기계공학부 교수
2. 경북대학교 기계공학과 박사과정

Artificial Neural Network and Its Applications to Plastic Forming Process Analyses II

Y. S. Kim, J. J. Kim

1. School of Mechanical Engineering, Kyungpook National University
2. Graduate School of Mechanical Engineering, Kyungpook National University

1. 서 론

인공신경망(artificial neural network (ANN))는 인간의 뇌의 원리와 정보처리 과정을 모방해서 만든 머신러닝 알고리즘이며, 딥러닝의 기본을 이룬다. 이 인공신경망은 최근에 소성가공분야에서 최근 최적가공조건의 탐색, 파단 및 스프링백 불량 원인분석 그리고 고부가가치 신소재의 개발과 재질예측 등에 활발히 적용되고 있다.[1-10]

최근 소성가공학회에 게재된 논문들의 데이터 분석으로부터 알 수 있듯이 소성가공 분야의 연구에 인공신경망의 적용이 증가하고 있다. 전 회[11]에서는 인공신경망의 개요와 기본 알고리즘에 대해서 알기 쉽게 설명하였다. 본 회에서는 소성가공 문제에서와 같이 입력 변수(입력 뉴런)가 한정된 개수로 정해져 있으며 반복 실험의 결과값들을 이용하여 새로운 입력변수에 대한 결과값을 추론하기 위한 파이썬 프로그램을 이용하는 방법에 대해서 설명한다.

2. 반복적 입력값을 갖는 경우

간단한 문제로 다음과 같은 단층 퍼셉트론의 신경망 구조에서 입력값이 Table 1 에서와 같이 4 회에 걸쳐서 반복적으로 들어오는 경우에 각 뉴런의 파라미터값 (가중값과 바이어스)을 갱신하는 방법에 대해서 생각해 보자.

여기서는 바이어스는 없고 활성화 함수로 시그모이드 함수를 사용한다. 또한 초기 가중값은 0.1, 0.2, 0.3 이며 학습률은 0.05 로 한다.

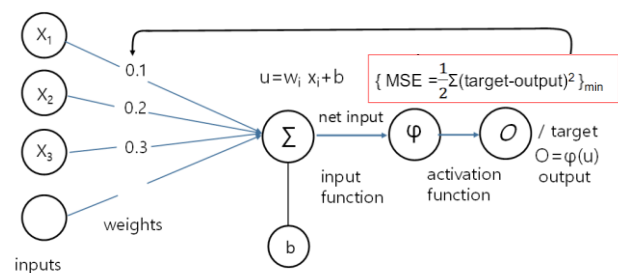


Fig. 1 Simple artificial neural network

Table 1 Training data set including input and output for artificial neural network

	Input			Output
	X ₁	X ₂	X ₃	
Training data 1	0	0	1	0
Training data 2	1	1	1	1
Training data 3	1	0	1	1
Training data 4	0	1	1	0
New data	1	0	0	?

(a) 통상의 구배하강법 (gradient descent)에 의해 학습 데이터 1 (0,0,1)에 대해서 전 호 [11]의 식 (7), (12), (13)을 이용하여 순 방향 계산을 수행하면

$$u = x_1 w_1 + x_2 w_2 + x_3 w_3 \\ = 0 * 0.1 + 0 * 0.2 + 1 * 0.3 = 0.3$$

$$\varphi(u) = \frac{1}{1 + e^{-u}} = \frac{1}{1 + e^{-0.3}} \approx 0.5744 (= O)$$

$$E = \frac{1}{2}(t - O)^2 = \frac{1}{2}(0 - 0.5744)^2 \approx 0.1649$$

$$\frac{\partial E}{\partial O} = O - t \approx 0.5744 - 0 \approx 0.5744$$

$$\frac{\partial O}{\partial u} = \varphi(1 - \varphi) = 0.5744 * (1 - 0.5744)$$

$$\approx 0.2444$$

$$\frac{\partial E}{\partial O} \frac{\partial O}{\partial u} = 0.5744 * 0.2444 \approx 0.1404$$

학습 데이터 2 (1,1,1)에 대해서 순방향 계산을 수행하면

$$u = x_1 w_1 + x_2 w_2 + x_3 w_3 \\ = 1 * 0.1 + 1 * 0.2 + 1 * 0.3 = 0.6$$

$$\varphi(u) = \frac{1}{1 + e^{-u}} = \frac{1}{1 + e^{-0.6}} \approx 0.6456 (= O)$$

$$E = \frac{1}{2}(t - O)^2 = \frac{1}{2}(1 - 0.6456)^2 \approx 0.0627$$

$$\frac{\partial E}{\partial O} = O - t \approx 0.6456 - 1 \approx -0.3544$$

$$\frac{\partial O}{\partial u} = \varphi(1 - \varphi) = 0.6456 * (1 - 0.6456) \\ \approx 0.2288$$

$$\frac{\partial E}{\partial O} \frac{\partial O}{\partial u} = -0.3544 * 0.2288 \approx -0.0810$$

학습 데이터 3 (1,0,1)과 학습 데이터 4 (0,1,1)에 동일한 방법으로 순방향 계산을 수행한다. 따라서 파라미터값은 역방향 계산에서 전 호의 식 (17), (18)을 이용하여 다음과 같이 갱신된다.

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial w_1} = \left\{ \frac{\partial u}{\partial w_1} \right\}^T \left\{ \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right\} \\ = (0 \ 1 \ 1 \ 0) \begin{pmatrix} 0.1404 \\ -0.0810 \\ -0.0964 \\ 0.1467 \end{pmatrix} = 0.1774$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial w_2} = \left\{ \frac{\partial u}{\partial w_2} \right\}^T \left\{ \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right\} \\ = (0 \ 1 \ 0 \ 1) \begin{pmatrix} 0.1404 \\ -0.0810 \\ -0.0964 \\ 0.1467 \end{pmatrix} = 0.0657$$

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial w_3} = \left\{ \frac{\partial u}{\partial w_3} \right\}^T \left\{ \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right\} \\ = (1 \ 1 \ 1 \ 1) \begin{pmatrix} 0.1404 \\ -0.0810 \\ -0.0964 \\ 0.1467 \end{pmatrix} = 0.1097$$

$$w_1^{t+1} = w_1^t - \alpha \frac{\partial E}{\partial w_1} = 0.1 - 0.05 * 0.1774$$

$$\approx 0.0911$$

$$w_2^{t+1} = w_2^t - \alpha \frac{\partial E}{\partial w_2} = 0.2 - 0.05 * 0.0657$$

$$\approx 0.1967$$

$$w_3^{t+1} = w_3^t - \alpha \frac{\partial E}{\partial w_3} = 0.3 - 0.05 * 0.1097$$

$$\approx 0.2945$$

(b) 한편 다른 방법으로 오차함수를 미분하여 파라미터값을 갱신할 때 개별 데이터로 하는 것이 아니고 N 개의 모든 학습 데이터에 대해서 오차함수의 기울기의 평균을 사용하여 일괄적으로 파라미터값을 갱신하는 방법을 생각할 수 있다. 즉, 학습하고자 하는 모델의 손실함수는 다음과 같이 전체 데이터에 대한 평균으로 표현되므로

$$E(w) = \frac{1}{N} \sum_{i=1}^N E_i(w)$$

이 경우에 구배하강법에서 최적의 파라미터값을 다음과 같이 나타낸다.

$$w_i^{t+1} = w_i^t - \alpha (\nabla E(w)) = w_i^t - \alpha \left(\frac{1}{N} \sum_{i=1}^N \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right)$$

여기서

$$\left(\frac{1}{N} \sum_{i=1}^N \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right) = \frac{1}{4} (0.1403 - 0.0810 - 0.0964 + 0.1467) = 0.0274$$

따라서

$$w_1^{t+1} = w_1^t - \alpha \left(\frac{1}{N} \sum_{i=1}^N \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \right)$$

$$= 0.1 - 0.05 * 0.0274 = 0.0986$$

$$w_2^{t+1} = 0.2 - 0.05 * 0.0274 = 0.1986$$

$$w_3^{t+1} = 0.3 - 0.05 * 0.0274 = 0.2963$$

이 방법을 배치구배하강법 (Batch gradient descent, BGD) 또는 전배치구배하강법 (Full-batch gradient descent)이라고 한다. 여기서 배치란 모델 학습의 반복 1 회, 즉 파라미터값의 갱신 1 회에 사용되는 학습 데이터의 집합이다.

(c) 한편 각각의 개별 학습 데이터에 대해서 파라미터값을 구하여 갱신해가는 과정을 반복하는 방법을 생각할 수 있다. 이 경우에는 학습데이터 1 (0,0,1)에 대해서

$$w_1 = w_1 - \alpha * \frac{\partial E}{\partial w_1} = w_1 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \left(\frac{\partial u}{\partial w_1} \right)$$

$$= 0.1 - 0.05 * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_1$$

$$= 0.1 - 0.05 * 0.1404 * 0 = 0.1$$

$$w_2 = w_2 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_2$$

$$= 0.2 - 0.05 * 0.1404 * 0 = 0.2$$

$$w_3 = w_3 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_3$$

$$= 0.3 - 0.05 * 0.1404 * 1 \approx 0.2929$$

이렇게 갱신된 파라미터값을 다음의 학습 데이터 2 (1,1,1)에 대해서 다시 갱신해간다. 즉,

$$w_1 = w_1 - \alpha * \frac{\partial E}{\partial w_1} = w_1 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} \left(\frac{\partial u}{\partial w_1} \right)$$

$$= 0.1 - 0.05 * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_1$$

$$= 0.1 - 0.05 * (-0.0810) * 1 \approx 0.1040$$

$$w_2 = w_2 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_2$$

$$= 0.2 - 0.05 * (-0.0810) * 1 \approx 0.2040$$

$$w_3 = w_3 - \alpha * \frac{\partial E}{\partial O} \frac{\partial O}{\partial u} x_3$$

$$= 0.2929 - 0.05 * (-0.0810) * 1 \approx 0.2969$$

이 과정들을 모든 학습 데이터에 대해서 반복해간다. 이 방법을 확률적 구배하강법 (stochastic gradient descent, SGD)라고 한다. 여기서 확률적이란 표현은 학습 데이터를 랜덤하게 뽑아서 학습시킨다는 의미이다.

3. Python 프로그램 예

인공신경망 계산은 Matlab SW 를 통해서 수행할 수 있지만 알고리즘의 구체적인 작동 기구를 파악하기 어려운 단점이 있다. 최근에는 파이썬[12]을 이용한 인공신경망 계산이 널리 이용되고 있어서

이하에서는 파이썬을 이용한 간단한 문제와 소성 가공문제에의 적용 예에 대해서 설명한다.

파이썬 문법이 매우 쉬워서 초보자들이 처음 프로그래밍을 배울 때 추천되는 언어로 타 프로그래밍 언어 대비 접근성과 응용력이 좋다고 알려져 있다. 파이썬을 작동시키기 위해서는 사이파이(SciPy), 넘파이(NumPy, www.numpy.org)와 심파이(SymPy, www.sympy.org) 등의 패키지들이 제공하는 기능을 불러와야 한다.

넘파이는 파이썬에서 수치 해석, 특히 선형대수 계산 기능을 제공한다. 특히 다차원의 행렬 자료구조(n-dimensional array, ndarray) 클래스와 벡터연산, 데이터 정렬, 인덱싱 등과 같은 수학연산에서 가장 기본적인 기능을 제공하는 중요한 패키지이다. 반면 고급 수학 함수, 수치적 미적분, 미분 방정식 계산, 최적화, 신호 처리 등에 사용하는 다양한 과학 기술 계산 기능을 제공한다.

넘파이는 선형 대수, 푸리에 변환 등을 해결하는 데 도움이 될 수 있는 여러 함수를 제공하며 사이파이는 실제로 이러한 함수의 모든 기능을 다른 많은 기능(학습알고리즘 및 최적화기법 등)과 함께 포함하는 파이썬 패키지이다. 파이썬을 사용하여 과학적 분석을 수행하는 경우 사이파이가 넘파이를 기반으로 작동되므로 넘파이와 사이파이를 모두 설치해야 한다. 또한 심파이는 기호를 사용하여 다양한 수학 방정식, 미분방정식 등을 계산할 수 있는 파이썬 패키지이다.

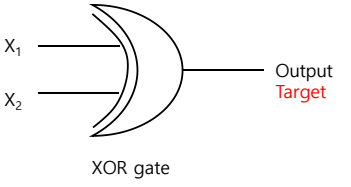
또한 팬더스(pandas, pandas.pydata.org)는 테이블 형태의 데이터를 다루는 데이터프레임 자료형을 제공하고, 사이킷런 (sklearn, <https://scikit-learn.org/stable/>)은 파이썬에서 머신러닝 분석을 할 때 유용하게 사용할 수 있는 라이브러리이고, 맷플롯립(Matplotlib, matplotlib.org)은 파이썬에서 데이터를 차트나 플롯(Plot)으로 그려주는 데이터 시각화(Data Visualization) 패키지이다.

본 해설에서는 구글 클라우드 상에서 운영하는 프리웨어 소프트웨어 <https://colab.research.google.com/>를 이용하였다. Colab은 colaboratory의 약어로 구글에서 제공하는 대화형 환경인 Jupyter 메모장에서 파이썬 코드를 작성하고 실행할 수 있도록 되어 있다.

부록 1에 첨부한 파이썬 프로그램을 이용하여 Table 2의 XOR 논리회로에 대한 오차 역전파에서

초기 임의로 부여한 가중값이 어떻게 갱신되고 해가 얻어지는지를 Fig. 2에 나타내었다. 인공신경망은 입력층 2개의 뉴런, 은닉층에 2개의 뉴런, 출력층에 하나의 뉴런 구조를 갖는 것으로 하였고, 활성화 함수로는 은닉층과 출력층에서 모두 시그모이드 함수를 사용하였다.

Table 2 Schematic of XOR gate with inputs and target

	Input X ₁	Input X ₂	Target
	0	0	0
	0	1	1
	1	0	1
	1	1	0
	1	0	?

Epoch 0에서는 초기에 부여한 파라미터값(가중값과 바이어스)와 입력값들로부터 순방향 정보전달에 의해서 예측한 결과값을 나타내었다. 입력값의 차이에도 불구하고 결과값이 0.7 근방으로 거의 모두 동일한 것을 나타내고 있음을 알 수 있다. Epoch 1000 회 계산에서는 예측한 값의 개선이 전혀 보이지 않음을 알 수 있다. Epoch 10,000 회 계산에서는 목표값(Target)과 비교하면 예측한 결과값 (0.0764, 0.9271, 0.9271, 0.0817)이 목표값(0,1,1,0)에 근접하고 있음을 알 수 있다.

여기서는 어떻게 계산이 진행되는지를 나타내기 위하여 초기 파라미터(가중값과 바이어스)를 일정한 값으로 지정하였지만 일반적으로 이들 값을 랜덤하게 부여한다. 파라미터값을 랜덤하게 부여한 경우에는 10,000 반복 계산에서 예측값이 (0.0635, 0.9411, 0.9412, 0.0637)으로 가중값을 임의로 부여한 경우의 예측값 (0.0764, 0.9271, 0.9271, 0.0817) 보다 약간 해가 개선되고 있음을 알 수 있다. 또한 학습률을 $\alpha=0.3$ 으로 하고 100,000 회 계산을 하면 (0.0073, 0.9938, 0.9938, 0.0063)이 얻어지며 목표값에 보다 가까워지는 것을 알 수 있다. 한편 은닉층과 출력층에서 바이어스를 고려하지 않은 경우에는 10,000 반복 계산에서 예측값이 (0.4927, 0.5020, 0.5069, 0.5128) 로 바이어스를 고려한 경우보다 해의 개선 속도가 매우 느리다는 것을 알 수 있다.

부록 1의 프로그램에 전 보[11]에서 사용한 식들이 어떻게 이용되고 있는지를 나타내었다.

여기서 random 은 균등분포에 따른 난수발생을 나타내고, dot 는 두 벡터의 내적을, ().T 는 행렬의 전치를 나타낸다.

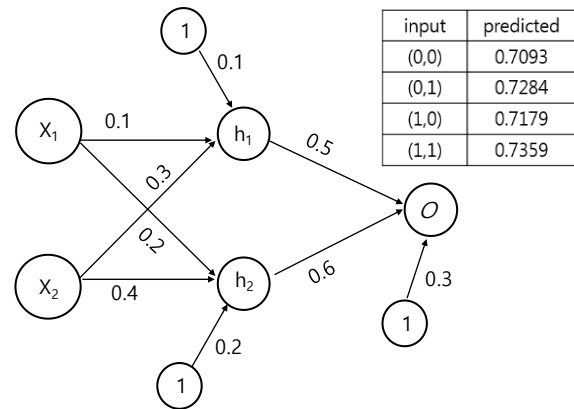
이상에서 기술한 파이썬 코드에서는 ANN 에서 순방향과 역방향 계산 과정을 상세하게 코딩하였지만 케라스(Keras) 라고 하는 오픈 소스 신경망 라이브러리를 이용하면 ANN 을 한층 간단하게 구현할 수 있다. 케라스는 텐서 플로의 공식 고수준 API 로 딥 러닝 모델 구축 프로세스를 가속화하기 위해 파이썬으로 작성된 오픈 소스 신경망 라이브러리이다.[13] 이 케라스는 더 높은 수준의 직관적인 추상화 집합을 표현함으로써 Matlab 과 같이 신경망의 설계와 계산을 쉽게 만들어준다.

인공신경망 적용결과에의 우열은 입력데이터의 개수, 손실함수 및 최적화방법의 적정성 등에 의존한다. 신경망 알고리즘에서 손실함수는 반복 계산에 대한 손실을 최소화하기 위해(손실 함수의 전역 최소값으로 이동시키는 것) 역전파를 사용하여 가중치 매개변수를 업데이트하는 함수이다. 케라스에서는 손실함수로 linear, sigmoid, ReLU, tanh, softmax, ELU 등등을 지원한다.

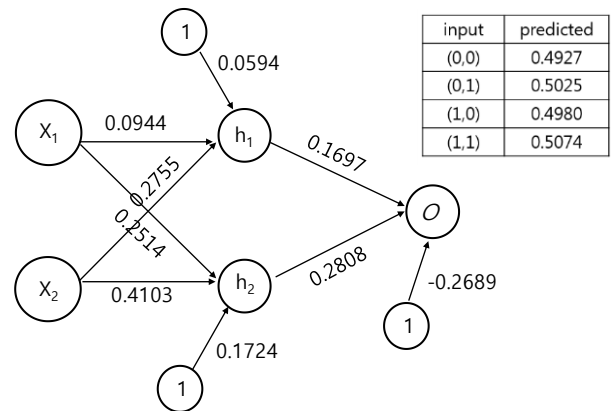
최적화 방법(optimizer 라고 불림)으로 위에서 사용한 경사하강법은 신경망에서 사용되는 가장 오래된 최적화 방법이지만 그 외에도 많은 방법들이 제안되어 있다. 케라스에서 자주 사용되고 있는 최적화 방법들로는 SGD(stochastic gradient descent, 확률적 구배하강법), Adam (adaptive moment estimation), RMSprop, Adamax, Adagrad 등이 있다.[14] Adam 최적화기법은 SGD 방법으로써 일차 및 이차모멘트 측정에 기반을 둔 것으로 계산이 효율적이고 메모리 사용량이 적은 특징이 있다.

SGD 는 얇은 네트워크에서 잘 작동하며, RMSprop 은 순환 신경망에서 잘 작동한다. 적합한 최적화 방법의 선택은 개인의 경험에 따른다. Keras 에서는 기본적으로 "Adam" 옵티마이저를 사용한다.

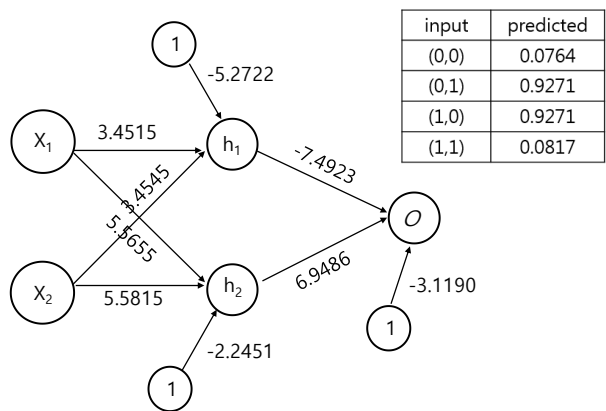
많은 인공신경망 교재에서 활성화 함수로 이진 분류 문제의 출력층에는 시그모이드 함수를 사용하고, 은닉층에는 가능한 렐루 계열 함수(ReLU, LeakyReLU, ELU)를 사용하고, 다중 클래스 분류 문제의 출력층에는 소프트맥스 함수를 사용하는 것을 추천하고 있다.



(a) epoch 0



(b) epoch 1000



(c) epoch 10000

Fig. 2 Shape of sigmoid, tanh, ReLu, and leaky ReLu function: (1) epoch 0; (2) epoch 1000; (3) epoch 10000

4. 소성가공문제에의 적용

인공신경망 알고리즘이 다양한 소성가공 문제에 적용되어 가공성 및 가공불량 및 재질 예측 등에 널리 사용되고 있다.[15-17] 여기서는 최근에 널리 연구되고 있는 점진판재성형(incremental sheet forming) 공정에서 판재의 가공한계와 성형 후의 스프링백의 예측에 적용한 사례를 설명하고자 한다. 점진판재성형은 상하형의 금형을 사용하지 않기 때문에 무다이 성형(die-less forming)이라고도 불린다.

점진판재성형 기술은 기존의 프레스 공정과는 달리 상하형의 금형을 사용하지 않고 Fig. 3 에서와 같이 외곽으로 판재를 고정된 상태에서 CNC 머신 또는 로봇 팔에 부착된 공구이동을 이용하여 국부적 점진적으로 판재를 성형하는 기술이다. 이 기술은 프레스 기술과 같이 대량 생산에는 적합하지 않지만 고가의 금형과 프레스 기계를 사용하지 않기 때문에 저렴하게 소량의 판재 제품이나 시제품 생산에 적합하다고 알려져 있다.[17]

이 점진성형공정에서 성형제품의 가공성과 성형 후 제품의 형상 정밀도는 제품의 성형 각도, 공구의 직경, 공구의 z 방향 이동피치, 공구의 이동속도 등등 공정변수에 크게 의존한다.

이 점진판재성형에는 재료의 외곽이 고정되어 재료의 유입이 제한되는 한계가 있기 때문에 성형성이 낮은 단점이 있다. 성형성을 향상시키기 위한 대책으로 플랜지부에 홀을 가공하는 홀랜싱(hole lancing) 공정이 제안되어 있다. [18] 여기서는 스프링백 크기에 미치는 홀랜싱의 크기를 포함한 공정 변수들의 영향을 평가하고 스프링백을 예측하는 수단으로 인공신경망 기법을 적용한 사례를 이 하에 설명한다.

Fig. 4 에 홀랜싱을 갖는 두께 0.8mm 의 AL5052-O 판재 시편을 이용하여 사각 피라미드 형상 제품을 깊이 40mm 까지 점진판재성형한 형상을 나타내었다. Table 3 에 실험계획법으로 설계한 실험표에 따라서 다양한 조건 하에서 총 18 회의 점진판재성형 실험을 수행한 후 성형된 시편을 CMM(coordinate measurement machine)으로 스프링백 크기를 측정된 결과를 나타내었다.

이 문제를 풀기 위한 ANN 구조는 입력층에 성형각도, 공구직경, 홀의 폭, 공구의 z 방향 피치, 공구 이송속도에 5 개의 뉴런, 2 개의 은닉층에 각각 10개와 30개의 뉴런, 출력층에 1개의 뉴런을 갖는 구조이다.

Table 3 에 나타낸 실험계획법으로 설계한 실험표에 따른 입력 데이터를 그대로 파이썬 프로그램에 입력하면 계산 수행이 제대로 이루어지지 않는다. 왜냐하면 시그모이드 함수를 사용하는 경우에 입력 데이터 값이 0 과 1 사이값이 아닌 매우 큰 값을 가지면 시그모이드 함수 값이 근사적으로 1 이 되기 때문에 파라미터값의 갱신이 잘 이루어지지 않는다. 따라서 입력값, 출력값 들은 모두 변환하여 0 과 1 사이의 값을 갖도록 정규화(Normalized) 하거나

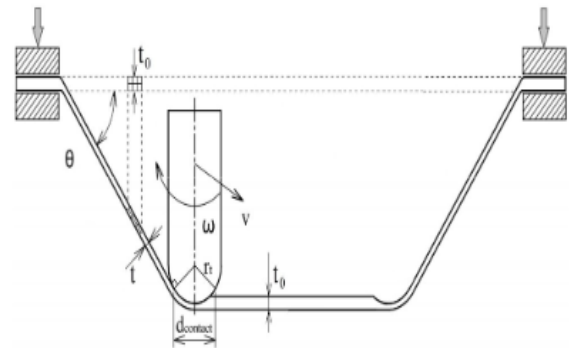


Fig. 3 Schematic of single-point incremental sheet forming

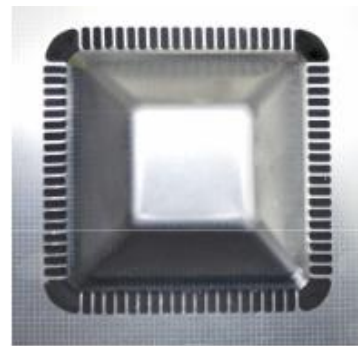
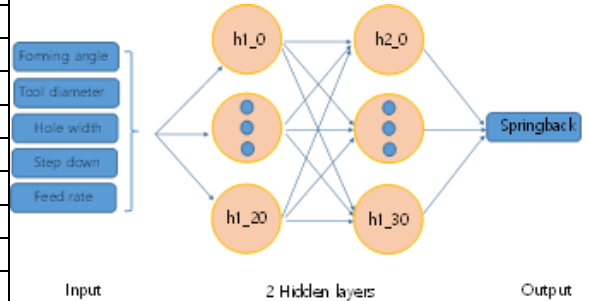


Fig. 4 Specimen formed by incremental sheet forming process with hole lancing

Table 3 Design of experiments for incremental sheet forming

Exp.	Forming angle [°]	Tool diameter [mm]	Hole width [mm]	Step down [mm]	Feed rate [mm/min]	Spring-back [mm]
1	50	12	3	0.4	1500	0.040
2	60	16	5	1	1500	1.355
3	60	12	3	1	1000	1.021
4	55	12	0	1	500	0.961
5	50	16	5	0.4	1000	0.085
6	50	16	3	1	500	0.383
7	55	16	0	0.4	1500	1.496
8	55	8	5	1	1000	1.965
9	55	8	3	0.4	500	1.598
10	60	12	5	0.4	500	0.624
11	55	12	5	0.7	1500	0.854
12	50	8	5	0.7	500	1.003
13	50	8	0	1	1500	1.367
14	55	16	3	0.7	1000	0.524
15	60	8	3	0.7	1500	2.393
16	60	16	0	0.7	500	1.458
17	50	12	0	0.7	1000	0.711
18	60	8	0	0.4	1000	2.116



$Z = \frac{x-m}{s}$ (m 은 평균, S 는 표준편차) 인 관계를 이용하여 평균이 0 이고 표준편차가 1 인 데이터로 표준화(standardization) 시켜 입력해야 한다는 것에 주의해야 한다.

표준화도 정규화라고 자주 불린다. 이 방법은 머신러닝이나 딥러닝의 경우에도 동일하게 적용된다.

부록 2 에 점진판재성형에서 공정변수들의 변화에 따른 스프링백 예측을 위한 인공신경망 프로그램을 나타내었다.

ANN 에서는 실험 데이터를 충분한 정확도로 근사적으로 잘 따라가는 최적의 모델을 결정하고 새로운 데이터에 대한 근사값을 추정, 판단하는 것이 목적이다. 그런데 경우에 따라서는 학습 데이터를 다 외워버려서 실험 데이터를 너무 완벽하게 따라가는 경우가 있고(과적합(overfitting, 오버 피팅)이라고 함) 이 경우에는 새로운 데이터나 학습이 되지 않은 데이터에 대해서는 추정이 어려워진다. 이와 반대의 경우로 모델이 학습 데이터를 잘 따라가지 못하여 데이터 전체의 (underfitting, 언더 피팅)이 있을 수도 있다. (Fig. 5)

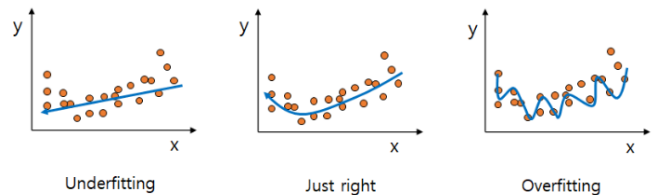
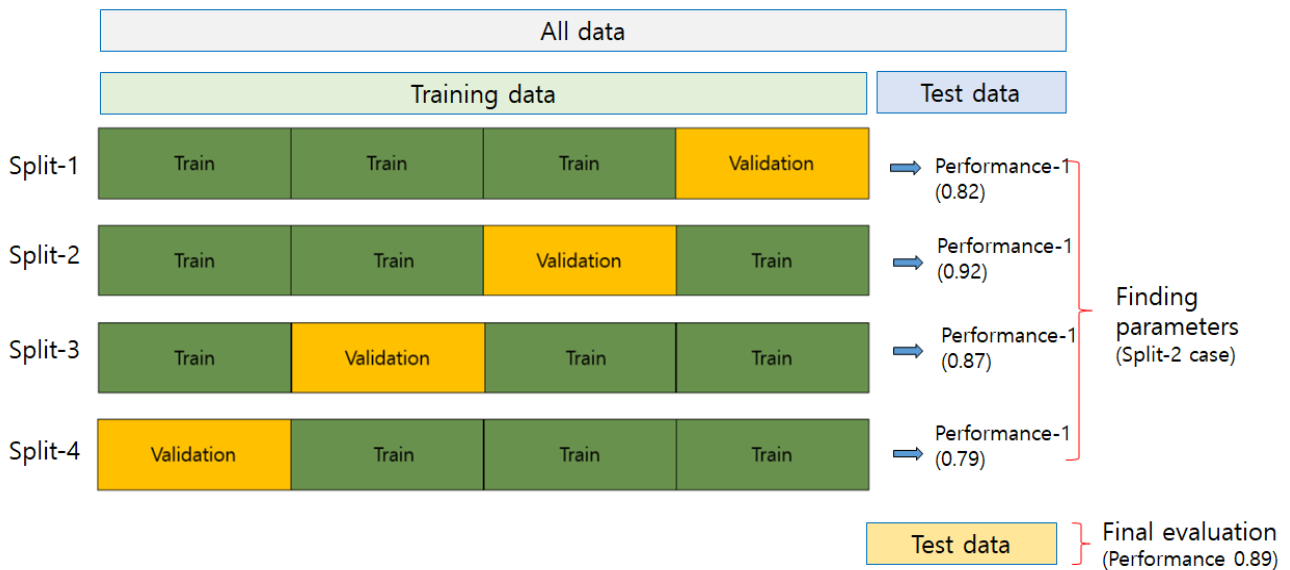


Fig. 5 Result cases after training the data set: under fitting; just right; over fitting

따라서 이런 과적합과 과소적합을 피하고 균형있게 학습된 적절한 모델을 찾기 위해서는 ANN 사용에 있어서는 반드시 학습을 수행하면서 선택한 인공신경망 구조와 결과의 유효성을 검사하여야 한다. ANN 에서는 통상 과적합과 과소적합을 피하기 위해 학습-검증-테스트 방법 (train-validation-test method)에 따라서 학습 데이터 셋트와 테스트 데이터 셋트 (또는 학습 데이터 셋트, 검증 데이터 셋트와 테스트 데이터 셋트)로 분할한다. 일반적으로 입력 데이터 셋트의 60%를 ANN 학습(training)에 사용하고 나머지 40%를 학습된 모델의 검증(validation)과 테스트(testing)에 사용한다.

Fig. 6 Four-fold cross validation with the performance



검증 데이터 셋트는 ANN 모델의 성능을 평가하기 위해 입력 데이터의 일부를 사용하며 학습한 모델이 잘 예측을 하는지 평가하는데 쓰인다. 한편 소성가공문제에서와 같이 학습 데이터가 적은 경우에 학습-테스트검증 방법에서는 검증 데이터 셋트가 고정될 가능성이 높기 때문에 정확도가 비현실적으로 나올 가능성이 높다. 따라서 검증 데이터 셋트를 고정하지 않고 데이터의 모든 부분을 사용하여 검증하고 그 중에서 가장 성능이 우수한 모델을 결정하는 방법이 자주 이용된다. 이 방법을 k-겹 교차 검증(k-fold cross validation)이라고 한다. [19]

k-겹 교차 검증의 순서는 다음과 같다. 무작위로 입력 데이터를 k 개의 겹(fold)로 나눈 후에 각 집단 중 하나의 집단을 검증 데이터 셋트로, 그리고 나머지는 학습을 위한 학습 데이터 셋트로 사용한다. 학습한 모델을 기준으로 테스트 셋트로 평가한 에러를 기록한다. 이 과정을 돌아가면서 k 번 실시한다. 기록한 에러를 평균(cross-validation error, 교차검증에러)하고, 모델을 평가하는 하나의 지표로 사용한다. k는 보통 10 정도로 하며, k가 작을수록 결과가 정확하지 않다고 알려져 있다.

이들 학습 데이터 셋트와 검증 데이터 셋트를 어떻게 선택하느냐에 따라서 ANN에서 예측한 결

과가 크게 달라진다. 따라서 실제 응용에서는 k-겹 교차 검증과 같이 학습 데이터 셋트와 검증 데이터 셋트의 묶음 조합들에 대해서 가장 정밀도가 높은 ANN 모델의 파라미터들을 결정 한 후에 테스트 데이터 셋트로 결과를 최종 평가한다. Fig. 6에서 예들 든 4 겹 교차검증에서 두번째 검증에서 가장 성능이 좋은 0.92가 얻어졌다고 하면 이때 결정된 파라미터를 테스트 데이터 셋트에 적용하고 그 결과 성능이 0.89인 것을 나타내었다.

부록 2의 프로그램의 validation_data(x_val, y_val)에서는 무작위로 추출한 x_train, y_train 데이터를 이용하여 학습한 결과의 파라미터들을 이용하여 그 결과를 검증하기 위해 x_val, y_val 데이터를 사용한 것이다. 이 과정에서 검증 데이터를 사용하면 각 에포크마다 검증 데이터의 정확도 (accuracy 또는 mse)도 함께 출력되는데, 이 정확도는 훈련이 잘 되고 있는지를 나타낸다. 이 과정에서 검증 데이터의 손실이 낮아지다가 높아지기 시작하면 모델이 과적합되어 있다는 신호이므로 다른 학습데이터와 검증 데이터를 사용하는 반복계산을 통해서 성능이 우수한 모델을 선택 하여야 한다.

점진관재성형 실험에서 얻어진 스프링백 값과 설계한 인공신경망 구조에 대한 파이썬 계산에서 예측된 값의 정확도를 Table 4에 나타내었다.

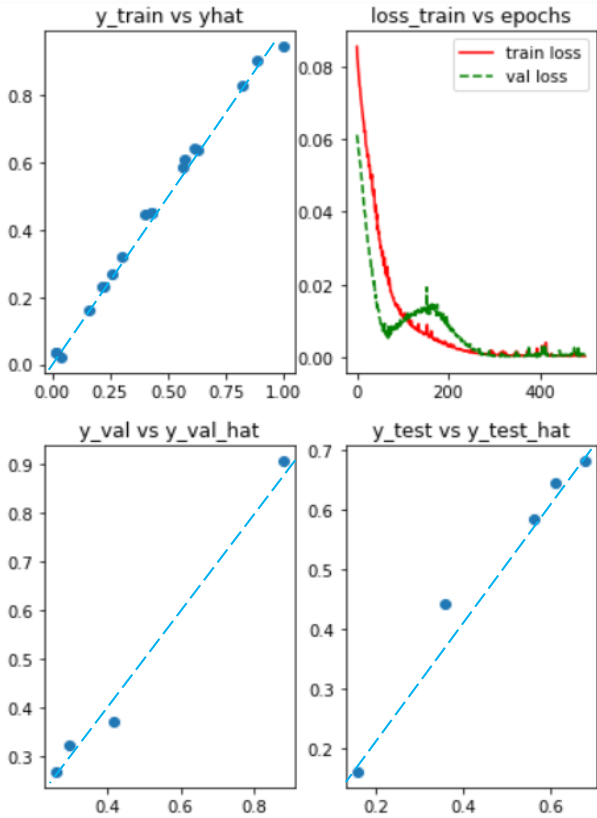


Fig. 7 Springback prediction in incremental sheet forming

본 예제에서는 입력 데이터 세트가 적어서 적절한 수렴이 얻어지지 않았기 때문에 일부 반복실험을 수행하여 모두 26 개의 입력 데이터 세트를 사용하였다. ANN 계산에서 무작위로 선정한 학습 데이터 세트와 검증 데이터 세트의 선택에 따라서 훈련 정확도가 서로 상이하므로 반복 계산을 통해서 검증데이터 세트에 약 95% 정도의 정확도를 갖는 경우만을 선택하여 적정 ANN 모델로 하는 것이 바람직하다.

부록 2의 프로그램에 첨부한 그림에 에포크에 따른 학습 데이터 세트의 손실과 검증 데이터 세트의 손실을 나타내었다. (Fig. 7)

반복 계산의 진행에 따라 500 에포크에서 손실이 모두 급격하게 0 근방으로 감소하고 있어 훈련이 잘 이루어 졌음을 알 수 있다. 파이썬 계산에서 예측한 스프링백 값은 최종 RMSE 값과 R 값으로부터 실험결과를 약 95% 정도의 신뢰도로 잘 예측하고 있음을 알 수 있다.

한편 부록 3에 나타난 입력 데이터가 60 개 정도로 많은 절삭력 예측문제에서는 수십회 정도의 에포크에서 손실이 거의 0에 근접하여 만족할만한 결과가 얻어지는 것을 알 수 있다. (Fig. 8)

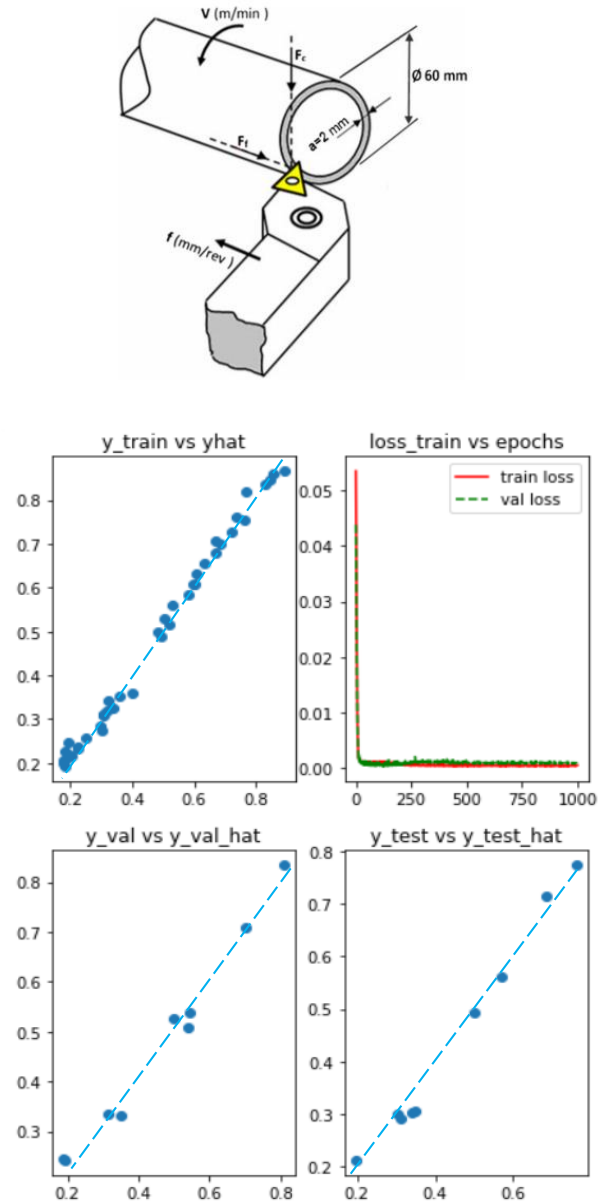


Fig. 8 Cutting force prediction in orthogonal machining

독자들은 소성가공학회 홈페이지의 일반자료실에 올린 첨부한 프로그램을 수정하여 사용할 것을

추천한다. 또한 소프트웨어 개발 프로젝트용 웹기반 호스팅 서비스인 깃허브(GitHub, <https://github.com/>)에서 관련된 프로그램 예제 소스를 받아 독자들의 목적에 맞게 수정하면 다양한 소성가공문제에 적용할 수 있을 것이다.

동일한 문제에 쌍곡 탄젠트 활성화 함수를 적용하고 역전파 계산과정에서 레벤버그-마퀴드(Levenberg-Marquardt, LM) 근사를 적용한 경우 Matlab SW 활용결과에 대한 비교를 다음 Table 4에 나타내었다. 이 LM 방법을 사용하면 역전파 과정에서 수렴이 늦어지거나 국부적인 최소값에 해가 빠져버리는 것을 방지하며 전체적으로는 SGD, Adam을 사용하는 경우보다 수렴성이 우수하다고 알려져 있다.[20]

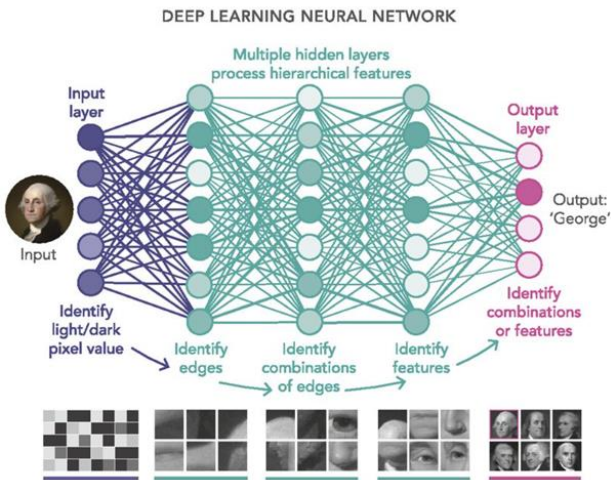


Fig. 9 Schematic of deep neural network with 2 hidden layers [21]

Table 4 The results predicted by Python and Matlab

		Test1	Test2	Test3	Test4
Experiment		0.542	1.325	2.056	0.861
Python	ANN	0.524	1.433	1.985	0.766
	Error(%)	3.3	8.1	3.4	11.0
Matlab	ANN	0.559	1.289	2.141	0.808
	Error(%)	3.2	2.7	4.2	6.1

위에서 설명한 인공지능의 다층 계층구조인 다층 퍼셉트론의 경우에 입력층의 노드수가 예를 들면 수십억 개로 매우 커지는 빅데이터 문제에서는 최적해를 구하는데 여러 문제점들을 포함하고

있다. 예를 들면 문제에 적합화되어 일반적인 답을 구할 수 없는 과적합 문제, 최적해에 이르기 전에 중간 언덕에서 해가 멈추는 문제, 위에서 설명한 것과 같이 시그모이드 함수를 사용하는 경우에 정보전달이 소실되는 문제, 수렴이 아닌 발산으로 피드백이 발생하는 문제 등등이다. 이런 문제점을 개선하여 빅데이터에 대한 최적해를 구해가는 인공지능 방법인 딥러닝(deep learning)이다. 통상 딥러닝은 Fig.9과 같이 은닉층이 2개 이상의 심화신경망(deep neural network, DNN)을 말한다.

한편 이 인공지능과 딥러닝 문제에서 입력 데이터가 너무 작은 경우에는 입력 데이터를 과도 평가(overfitting)하는 문제점이 있기 때문에 충분히 많은 데이터를 이용하여 학습시켜야 한다. 그러나 입력 데이터가 너무 많은 경우에는 입력 데이터 모두에 대해서 가중값의 변화에 따른 손실함수의 구배를 구하고 구배의 역방향으로 가중값을 갱신하는데 너무 많은 연산시간이 소요되기 때문에 비효율적이다.

이런 기본적인 달콤한 구배하강법에 대한 개선방법으로 앞에서 언급한 배치 구배하강법과 확률적 구배하강법 그리고 이 두 가지 방법의 단점을 보완한 미니배치 구배하강법(Mini-Batch (Stochastic) Gradient descent, MBGD 또는 MGD) 등이 제안되어 있다.[22]

MGD는 실제 모든 입력 데이터에 대한 편미분을 구하는 대신에 무작위로 선택된 데이터의 미니배치에 대한 편미분을 구하여 전체 가중값을 갱신하는 방법이다. 즉,

$$\begin{aligned}
 E(w) &= \frac{1}{m} \sum_{i=1}^m E_i(w) \quad (m < N) \\
 w_i^{t+1} &= w_i^t - \alpha(\nabla E(w)) \\
 &= w_i^t - \alpha \left(\frac{1}{m} \sum_{i=1}^m \frac{\partial E}{\partial w} \frac{\partial \sigma}{\partial u} \right)
 \end{aligned}
 \tag{1}$$

이 MGD에서는 미니 배치 크기를 어떻게 결정하느냐에 따라 학습 과정에 차이가 발생한다. 배치 크기가 크면 당연히 경사값이 정확해지지만 1에포크에 대한 계산량이 늘어나게 된다. 그러나 1에포크에서 각 미니 배치에 대한 경사값 $\frac{\partial E}{\partial w}$ 는 병렬 계산이 가능하므로, 많은 배치를 사용하고

병렬연산에 최적화된 그래픽처리장치(graphics processing unit, GPU)를 사용하면 연산속도를 가속시킬 수 있다.[23]

한편 가중값의 변화가 인공신경망 결과에 매우 작은 변환만을 미친다면 가중값을 효과적으로 학습시킬 수 없다. 이에 대한 해결방안으로 배치 정규화(batch normalization) 방법이 이용되기도 한다. 이 방법은 미니 배치의 평균과 분산을 이용해서 정규화한 뒤에 γ 값과 β 값을 통한 변환을 통해 비선형 성질을 유지하면서 학습을 하는 방법을 말한다. 이 방법은 전역 최적값을 구하지 못하고 국소 최적값 문제에 빠지는 가능성을 줄이기 위해서 사용하기 때문에 학습을 더 빨리 한다.[24] 또한 다수의 은닉층이 존재할 때 은닉층 간의 가중값의 차이가 너무 크면 해가 수렴하지 않는 경우가 발생한다. 이에 대한 대책으로 모든 층에서의 가중값의 변화가 매끈하게 되도록 하는 가중값 표준화 방법(weight standardization, WS)이 사용되기도 한다. 이 방법에서 가중값 W_{ij} 는 다음과 같이 정의된 \hat{W}_{ij} 로 치환된다.[25]

$$\hat{W}_{ij} = \frac{W_{ij} - \mu W_i}{\sigma W_i}$$

$$\mu W_i = \frac{1}{N} \sum_{i=1}^N W_{ij} \quad (2)$$

$$\sigma W_i = \sqrt{\frac{1}{N} \sum_{i=1}^N (W_{ij} - \mu W_i)^2}$$

이상에서 기술한 최적화 방법으로써 구배하강법의 단점으로는 (1) 각 파라미터의 중요도나 스케일 등이 모두 다른 경우에도 모든 파라미터에 동일한 학습률을 적용해야 하는 불합리한 문제점, (2) 수렴지점에서 더 정밀한 최소값을 찾기 위해서는 학습 중간에 학습률을 바꿀 필요가 있는데 구배하강법에서는 같은 값을 유지하는 문제점, (3) 최저점 근방에서 이동 관성을 고려하지 못하는 점 등이 있는 것으로 알려져 있다.

이에 대한 대책으로 구배의 모멘텀(momentum, 관성)을 고려하여 현재 파라미터를 업데이트해줄 때 이전 구배들도 계산에 포함해주는 방법이 있다. 즉, 구배가 0 인 평면을 만났다고 해도 전 단계의 구배가 남아 있어서 SGD에서 전역 최적해를 찾아갈 수 있게 만들어 준다. 대표적인 방법으로

Momentum, NAG, Adagrad, Adadelata, RMSprop, Adam, NAdam 등등이 널리 알려져 있다.[26-28]

5. 결론

본 해설에서는 인공신경망 알고리즘에 대한 설명과 XOR 논리회로에 대한 Python 프로그램과 실제 소성가공문제에의 적용 프로그램 설명을 통해 인공신경망 프로그램의 작동에 대한 이해를 높이고 소성가공분야에의 적용력을 높이도록 하였다. 본 해설을 통해 독자들은 인공신경망에 대한 최소한의 지식을 갖추어 소성가공분야의 산업현장과 연구에 도움이 되길 바란다. 본 해설에 사용한 python 프로그램(본문에 의거한 부록 1, 2, 3)을 한국소성가공학회 홈페이지의 일반자료실에 올려놓았다. 인공신경망을 소성가공 연구에 활용하고자 하는 독자들은 프로그램을 각자의 문제에 맞게 수정하여 사용하길 추천한다.

후 기

이 논문은 2021 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1A2C 1011224).

REFERENCES

- [1] D. H. Kim, D. J. Kim, B. M. Kim, J. C. Choi, 1997, Process Design of Multi-Step Drawing using Artificial Neural Network, Conf. on Trans. Mater. Process, pp. 144-147
- [2] A. J. Hong, K. D. Cheol, L. C. Joo, B. M. Kim, 2008, Springback Compensation of Sheet Metal Bending Process Based on DOE & ANN, Trans. Kor. Soc. Mech. Eng., Vol. 32, No. 11, pp. 990~996.
- [3] S. K. Lee, S. M. Kim, S. B. Lee, B. M. Kim, 2010, Optimization of Process Variables of Shape Drawing for Steering Spline Shaft, Trans. Mater. Process, Vol. 19, No. 2, pp. 132-137.
- [4] D. T. Nguyen, Y. S. Kim, D. W. Jung, Formability Predictions of Deep Drawing Process for Aluminum Alloy A1100-O Sheets by Using Combination FEM with ANN, 2012, Advanced Mat. Research, Vol. 472, pp. 781~786.
- [5] J. I. Choi, J. M. Lee, S. H. Baek, B. M. Kim, D. H. Kim, 2015, The Shoe Mold Design for Korea Standard Using

- Artificial Neural Network, *Trans. Mater. Process*, Vol. 24, No. 3, pp. 167-175.
- [6] V. C. Do, Y. S. Kim, Effect of Hole Lancing on the Forming Characteristic of Single Point Incremental Forming, 2017, *Proc. Eng.*, Vol. 184, pp. 35-42.
- [7] M. A. Woo, S. M. Lee, K. H. Lee, W. J. Song, J. Kim, 2018, Application of an Artificial Neural Network Model to Obtain Constitutive Equation Parameters of Materials in High Speed Forming Process, *Trans. Mater. Process*, Vol. 27, No. 6, pp. 331-338.
- [8] S. C. Ma, E. P. Kwon, S. D. Moon, Y. Choi, 2020, Prediction of Springback after V-Bending of High-Strength Steel Sheets Using Artificial Neural Networks, *Trans. Mater. Process*, Vol. 29, No. 6, pp. 338-346.
- [9] D. C. Yang, J. H. Lee, K. H. Yoon, J. S. Kim, 2020, A Study on the Prediction of Optimized Injection Molding Condition using Artificial Neural Network (ANN), *Trans. Mater. Process*, Vol. 29, No. 4, pp. 218-228.
- [10] E. S. Noh, S. R. Yi, M. S. Kim, S. M. Hong, 2020, Identification of Bolt Coating Defects Using CNN and Grad-CAM, *Trans. Kor. Soc. Mech. Eng. A*, Vol. 44, No. 11, pp. 835-842.
- [11] Y. S. Kim, J. J. Kim, 2021, Basics of artificial neural network and its applications to plastic forming process analyses I, *Trans. Mater. Processing*, Vol. 30, No. 4, pp.
- [12] <https://www.python.org/>
- [13] <https://keras.io/>
- [14] <https://keras.io/ko/optimizers/>
- [15] M. J. Kwak, J. W. Park, K. T. Park, B. S. Kang, 2020, A Development of Optimal Design Model for Initial Blank Shape Using Artificial Neural Network in Rectangular Case Forming with Large Aspect Ratio, *Trans. Mater. Process*, Vol. 29, No. 5, pp. 272-281.
- [16] M. J. Kwak, J. W. Park, K. T. Park, B. S. Kang., 2020, A Development of Longitudinal and Transverse Springback Prediction Model Using Artificial Neural Network in Multipoint Dieless Forming of Advanced High Strength Steel, *Trans. Mater. Process*, Vol. 29, No. 2, pp. 76-88.
- [17] S. H. Oh, X. Xiao, Y. S. Kim, 2021, Modeling of AA5052 Sheet Incremental Sheet Forming Process Using RSM-BPNN and Multi-optimization Using Genetic Algorithms, *Trans. Mater. Process*, Vol. 30, No. 3, pp. 125-133.
- [18] D. V. Cuong, D. C. Ahn, Y. S. Kim, Formability and effect of hole bridge in the single point incremental forming, 2017, *Int. J. Pre. Eng. Mat.*, vol. 18, pp. 453-460
- [19] https://scikit-learn.org/stable/modules/cross_validation.html
- [20] <https://pythonq.com/so/matlab/1814337>
- [21] M. M. Waldrop, 2019, News Feature: What are the limits of deep learning?, *PNAS*, Vol. 116, No. 4, pp. 1074-1077.
- [22] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. T. P. Tang, 2017, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, *arXiv preprint arXiv:1609.04836*.
- [23] https://ai-benchmark.com/ranking_deeplearning.html
- [24] S. Loffe, C. Szegedy, 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *PMLR*, pp. 448-456.
- [25] S. Santurkar, D. Tsipras, A. Ilyas, A. Mądry, 2018, How does batch normalization help optimization?, In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488-2498.
- [26] S. Ruder, 2016, An overview of gradient descent optimization algorithms, *arXiv preprint arXiv:1609.04747*.
- [27] Y. J. Lee, 2008, Optimization of weights for good generalization performance of neural networks, Thesis in KAIST.
- [28] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A. Y. Ng, 2011, On optimization methods for deep learning, *Proceedings of the 28 th International Conference on Machine Learning*, Bellevue, WA, USA.