

# 1장 텐스플로 2.0으로 신경망 구현

# 1장 목차

## 1장. 텐서플로 2.0으로 신경망 구현

\_\_텐서플로(TF)란?

\_\_케라스란?

\_\_텐서플로 2.0의 가장 중요한 변화

\_\_신경망 소개

\_\_퍼셉트론

\_\_텐서플로 2.0 코드 첫 번째 예제

\_\_다층 퍼셉트론: 신경망 첫 번째 예제

\_\_퍼셉트론 훈련의 문제점과 해결책

\_\_활성화 함수: 시그모이드

\_\_활성화 함수: tanh

\_\_활성화 함수: ReLU

\_\_추가적인 두 개의 활성화 함수: ELU와 LeakyReLU

\_\_활성화 함수

\_\_간단히 말해: 결국 신경망이란?

\_\_실제 예제: 필기체 숫자 인식

\_\_원핫 인코딩(OHE)

\_\_텐서플로 2.0으로 단순 신경망 정의

\_\_단순 텐서플로 2.0 신경망 실행과 베이스라인 구축

\_\_텐서플로 2.0의 단순 신경망을 은닉층으로 개선

\_\_텐서플로에서 드롭아웃으로 단순망 개선

\_\_텐서플로 2.0에서 여러 최적화기 테스트

\_\_에폭 수 증가시키기

\_\_최적화기 학습률 조절

\_\_내부 은닉층 개수 증가

\_\_배치 계산 크기 증가

\_\_필기체 인식 실행 차트 요약

\_\_정규화

\_\_과적합을 피하기 위한 정규화 적용

\_\_배치 정규화의 이해

\_\_구글 Colab 사용: CPU, GPU, TPU

\_\_감정 분석

\_\_초매개변수 튜닝과 AutoML

\_\_출력 예측

\_\_역전파에 대한 실용적 개괄

\_\_정리

\_\_딥러닝 접근법을 향해

가급적 수학적 이해는 간단히 하고, 개념적 이해를 위주로

# 텐서플로(TensorFlow)/케라스 (Keras)

- TensorFlow: 구글 브레인팀이 개발, Deep neural networks을 위한 오픈소스 소프트웨어 프레임워크
- Keras : 거의 모든 종류의 딥러닝 모델을 간편하게 훈련시킬수 파이썬을 위한 API,
- TensorFlow2.0: Keras를 내부에 구현하여 외부 호출 없이 사용가능

```
import tensorflow.compat.v1 as tf

in_a = tf.placeholder(dtype=tf.float32, shape=(2))

def model(x):
    with tf.variable_scope("matmul"):
        W = tf.get_variable("W", initializer=tf.ones(shape=(2,2)))
        b = tf.get_variable("b", initializer=tf.zeros(shape=(2)))
        return x * W + b

out_a = model(in_a)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    outs = sess.run([out_a],
                     feed_dict={in_a: [1, 0]})
```

```
import tensorflow as tf
W = tf.Variable(tf.ones(shape=(2,2)), name="W")
b = tf.Variable(tf.zeros(shape=(2)), name="b")

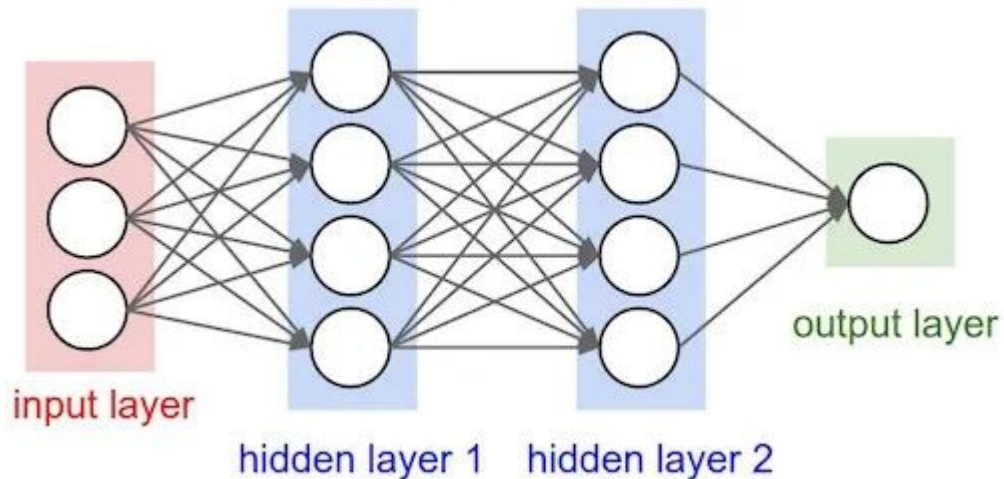
@tf.function
def model(x):
    return W * x + b

out_a = model([1,0])

print(out_a)
```

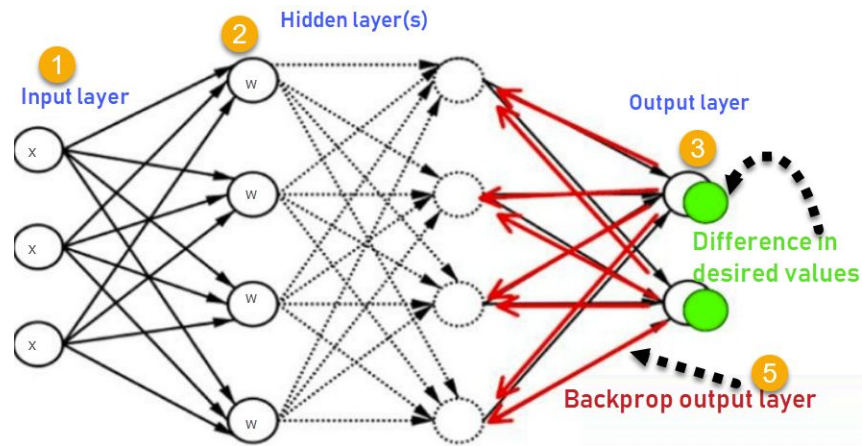
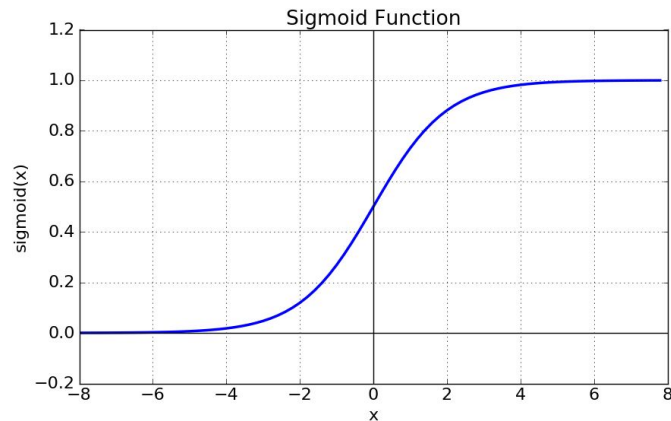
# 신경망(Neural Network) 개요

- 신경망 소개: 신경세포를 모방한 인공신경망을 연결한 머신러닝 모델
- 퍼셉트론: 단위 인공신경망세포, 입력층과 출력층만 있음, XOR 연산불가
- 다층퍼셉트론: 입력층과 출력층 사이에 은닉층(hidden layer), XOR 연산가능



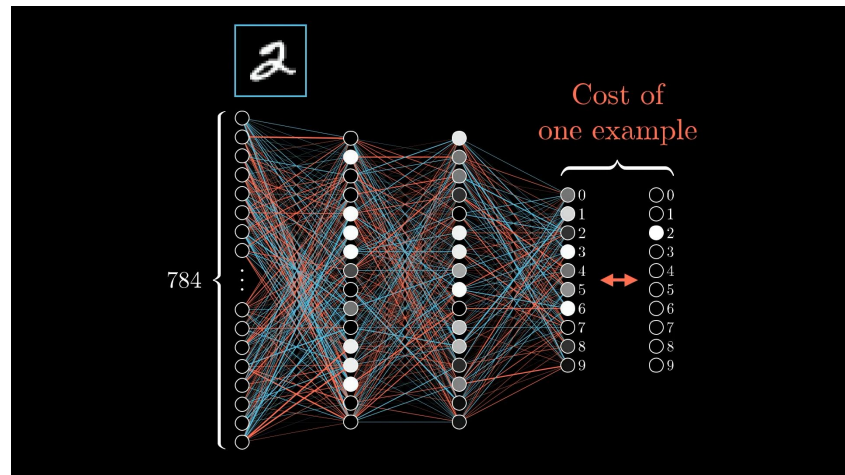
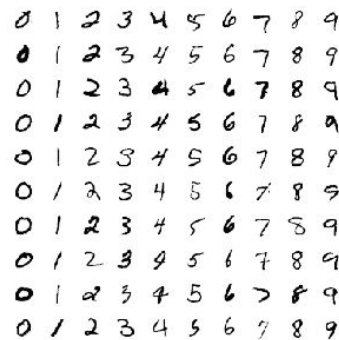
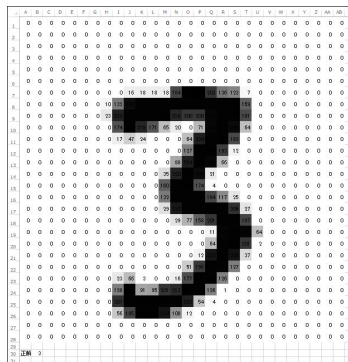
# 다중퍼셉트론의 학습

- 활성화 함수(Activation Function)
  - 시그모이드
  - tanh
  - ReLU
  - ELU와 LeakyReLU
- 역전파 알고리즘(Backpropagation): 먼저 계산 결과와 정답의 오차를 구해 이 오차에 관여하는 값들의 가중치를 수정하여 오차가 작아지는 방향으로 일정 횟수를 반복해 수정하는 방법



# 실제 예제: 필기체 숫자 인식

- MNIST 데이터베이스 (Modified National Institute of Standards and Technology database)는 손으로 쓴 숫자들로 이루어진 대형 데이터베이스이며, 다양한 화상 처리 시스템을 트레이닝하기 위해 일반적으로 사용
- MNIST 데이터베이스는 60,000개의 트레이닝 이미지와 10,000개의 테스트 이미지를 포함



# Hands-on 1

- 단순신경망

# Hands-on 2

- 단순 신경망을 은닉층으로 개선

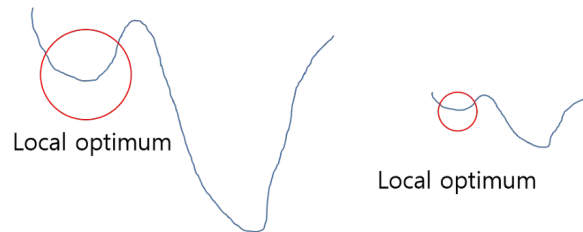
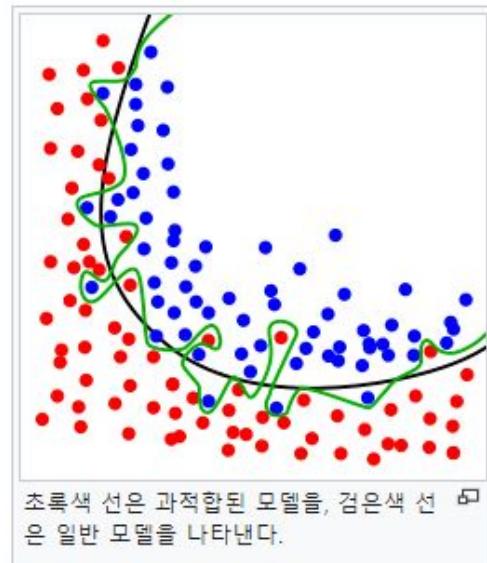


# 훈련의 개선

- 손실함수(Loss function)
  - 예측값과 실제값(레이블)의 차이를 구하는 기준을 의미하는 것으로 머신러닝 모델 학습에서 필수 구성,
  - 손실함수로 인해 모델의 성능이 달라지고 , 머신러닝 모델을 구현 시 손실함수 선택이 중요
- 과적합
- 정규화

# 훈련의 개선

- 손실함수(Loss function)
  - 예측값과 실제값(레이블)의 차이를 구하는 기준을 의미하는 것으로 머신러닝 모델 학습에서 필수 구성,
  - 손실함수로 인해 모델의 성능이 달라지고, 머신러닝 모델을 구현 시 손실함수 선택이 중요
- 과적합(overfitting)
  - machine learning에서 학습 데이터를 과하게 학습(overfitting)하는 것을 뜻함
  - 학습 데이터는 실제 데이터의 부분 집합
  - 학습데이터에 대해서는 오차가 감소하지만 실제 데이터에 대해서는 오차가 증가



# 과적합의 방지

- 정규화 Regularization
  - weight를 조정하는데 규제(제약)를 거는 기법
  - Overfitting을 막기위해 사용함(= 과적합을 완화해 일반화 성능을 높여주기 위한 기법)
    - L1정규화: LASSO(라쏘)
    - L2정규화: Lidge(릿지)
    - 일래스틱 정규화: L1과 L2의 조합
- 배치 정규화 Batch Normalization
  - 활성화함수의 활성화값 또는 출력값을 정규화(정규분포로 만든다)하는 작업
    - 학습 속도 상승
    - 초기 파라미터 의존도 약화
    - 모델 성능 향상

# Google Colab

- Colaboratory(줄여서 'Colab'이라고 함)을 통해 브라우저 내에서 Python 스크립트를 작성하고 실행
  - 구성이 필요하지 않음
  - 무료로 GPU 사용
  - 간편한 공유
- 구글 코랩 주소: <https://colab.research.google.com/>

