지능형 컴퓨팅과정 포트폴리오

PE 20173842 한근희

인공지능응용프로그래밍 강환수 교수님

<목차>

<강의 계획서>

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 2학기	전공	컴퓨터정보공학과(사회맞춤형 지능형 컴퓨팅과정)	학부	컴퓨터공학부
과 목 명	인공지능응용프로그래밍(2019028-PE)			

강의실 과 강의시간	목:2(3-217),3(3-217),4(3-217)	학점	3
교과분류	이론/실습	시수	3

담당 교수	강환수 + 연구실 : 2호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 화요일 14~15, 수요일 14~15

학과 교육목표	
과목 개요	1. 수업 운영방안 : 대면1시간+온라인 2시간 2. 온라인 수업 운영방법 : 시청할 기간을 정하여 동영상 시청 + 질의응답 3. 과목 개요 : 2010년 이후 파이썬의 폭발적인 인기는 제4차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 컴퓨팅 사고력은 누구나가 가져야 할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제4차 산업혁명 시대의 기술을 이끌고 있다. 이러한 제4차 산업혁명 시대에 핵심 기술인 데이터과학과 딥러닝의 붐은 파이썬을 최고의 인기 있는 프로그래밍 언어로 만들었다. 본 교과목은 파이썬 프로그래밍의 기초적이고 체계적인 이해를 바탕으로 딥러닝을 학습한다. 본 교과목을 통하여 인공지능을 이해하고 딥러닝의 기본적인 이론을 학습하고 GPU가 탑재된 컴퓨터와 텐서플로로 딥러닝을 직접 구현해 본다.
학습목표 및 성취수준	1. 컴퓨팅 사고력의 중요성을 인지하고 4차 산업혁명에서 인공지능의 필요성을 이해할 수 있다. 2. 인공지능과 머신러닝, 딥러닝을 이해하고 구글의 코랩을 사용해 딥러닝을 위한 Hello World인 MNIST 분류를 프로그래밍할 수 있다. 3. 구글의 코랩에서 파이썬과 텐서플로 프로그래밍을 수행할 수 있다. 4. 딥러닝 ANN, CNN, RNN을 이해하여 직접 구현할 수 있다.

	도서명	저자	출판사	비고
주교재	파이썬으로 배우는 누구나 코딩	강환수 외 1인	홍릉과학출판사	

수업시 사용도구	구글 코랩 [시작하세요! 텐서플로 2.0 프로그래밍] 등 여러 도서를 참고할 예정임
평가방법	중간고사 30%, 기말고사 40%, 과제물 및 퀴즈 10% 출석 20%(학교 규정, 학업성적 처리 지침에 따름)
수강안내	1. 머시러닝과 딥러닝을 이해하고 구분할 수 있다. 2. 머신러닝과 딥러닝 라이브러리인 텐서플로와 고수준 API인 케라스를 이해할 수 있다. 3. 딥러닝의 이진 분류와 다중 분류를 이해하고 구현할 수 있다. 4. 딥러닝의 선형 회귀를 이해하고 구현할 수 있다. 5. 딥러닝을 이해하고 오픈 데이터를 활용하여 직접 구현할 수 있다.

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

1 주차	[개강일(9/1), 수강정정(9/3 ~ 9/4)]
학습주제	교과목 소개 및 강의 계획 인공지능과 딥러닝 개요
목표및 내용	• 교과목 개요와 성적 산출 방법 등을 소개한다. • 4차 산업혁명 시대를 위한 머신러닝과 딥러닝의 기본을 이해한다. • 인공지능과 머신러닝, 딥러닝을 차이를 이해할 수 있다.
미리읽어오기	인공지능과 딥러닝
과제,시험,기타	#수업에서 제시
2 주차	[2주]
학습주제	텐서플로 이해 텐서플로 기초 프로그래밍
목표및 내용	• 텐서플로를 이해할 수 있다. • 구글의 코랩을 연결하여 실행할 수 있다. • 텐서플로 홈페이지에서 딥러닝 첫 프로그램을 직접 실행해 본다.
미리읽어오기	텐서플로
과제,시험,기타	#수업에서 제시
3 주차	[3주]
학습주제	MNIST 손글씨 프로그래밍
목표및 내용	• MNIST 이해하고 활용할 수 있다. • 텐서플로와 케라스를 이해할 수 있다. • 텐서플로 2.0 방식으로 MNIST 딥러닝 코딩을 구현할 수 있다. • 텐서플로 홈페이지에서 직접 활용할 수 있다.
미리읽어오기	MNIST 손글씨 데이터
과제,시험,기타	#수업에서 제시
4 주차	[4주]
학습주제	인공신경망의 이해
목표및 내용	• 신경망을 이해하고 간단한 신경망을 구현할 수 있다. • 활성화 함수를 이해하고 종류를 알 수 있다. • 신경망에서의 하이퍼 파라미터를 이해할 수 있다. • MNIST 모델에서 가중치와 편향의 수를 이해할 수 있다.
미리읽어오기	인공신경망
과제,시험,기타	#수업에서 제시
5 주차	[5주, 추석연휴(9/30~10/2) -> 보강(9/14~18, 9/21~25, 9/29~10/5, 10/5~8)]
학습주제	회귀와 분류
목표및 내용	• 선형 회귀와 분류의 차이를 이해하고 각각 활성화 함수를 구현할 수 있다. • 간단한 선형 회귀를 구현할 수 있다.
미리읽어오기	텐서플로 홈페이지에서 회귀
과제,시험,기타	#수업에서 제시

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

6 주차	[6주, 한글날(10/9) -> 보강(10/23, 10/6~10/12)]
학습주제	데이터 시각화
목표및 내용	• Matplotlib를 이해하고 필요한 정보를 시각화 할 수 있다. • 하나의 바탕에 여러 그림을 그릴 수 있다.
미리읽어오기	Matplotlib
과제,시험,기타	#수업에서 제시
7 주차	[7주]
학습주제	회귀와 분석 사례 프로그래밍 딥러닝 개인 포트폴리오 대회 설명
목표및 내용	• 패션 MNIST를 구현할 수 있다. • 자동차 연비 예측을 구현할 수 있다.
미리읽어오기	패션 MNIST 데이터
과제,시험,기타	#수업에서 제시
8 주차	[중간고사 기간 : 10/20(화)~30(금)]
학습주제	- 직무수행능력평가 1차(중간고사)
목표및 내용	직무수행능력평가, 서술형 평가
미리읽어오기	1주에서 7주까지의 수업 내용으로 평가
과제,시험,기타	
9 주차	[9주]
학습주제	합성곱의 이해와 구현
목표및 내용	• 합성곱 층의 이해하고 플링과 패딩을 활용할 수 있다. • 합성곱을 활용하여 손글씨 글자 이미지 MNIST 데이터를 딥러닝으로 구현할 수 있다. • 합성곱 구현으로 손글씨 글자 이미지 MNIST 데이터를 정확도 99% 이상 구현할 수 있다.
미리읽어오기	CNN
과제,시험,기타	딥러닝 개인 포트폴리오 준비
10 주차	[10주]
학습주제	순환신경망 RNN
목표및 내용	• 순환신경망 RNN을 이해할 수 있다. • 아나콘다를 설치하고 주피터 노트북을 사용할 수 있다. • 가상환경으로 tf를 만들어 tensorflow keras 등을 설치하고 첫 케라스 프로그램을 실행할 수 있다.
미리읽어오기	RNN
과제,시험,기타	딥러닝 개인 포트폴리오 준비

11 주차	[11주]
학습주제	순환신경망 LSTM과 GRU
목표및 내용	• 장기의존성문제를 이해할 수 있다. • 장기의존성문제를 해결한 LSTM을 이해하고 구현할 수 있다. • LSTM보다 심플하고 성능이 좋은 GRU를 이해하고 직접 구현할 수 있다.
미리읽어오기	LSTM GRU
과제,시험,기타	딥러닝 개인 포트폴리오 준비
12 주차	[12주]
학습주제	딥러닝 개인 포트폴리오 발표
목표및 내용	한 학기 동안 개인별로 작성한 포트폴리오 발표를 통해 심도 있는 딥러닝의 이해를 돕는다.
미리읽어오기	딥러닝 개인 포트폴리오
과제,시험,기타	딥러닝 개인 포트폴리오 발표
13 주차	[13주]
학습주제	딥러닝 구현 프로젝트 발표
목표및 내용	학습한 딥러닝 구현 모델을 활용해 직접 구현한 딥러닝 프로젝트를 직접 발표한다.
미리읽어오기	딥러닝 구현 프로젝트
과제,시험,기타	딥러닝 구현 프로젝트 발표
14 주차	[14주]
학습주제	케라스의 Dense, Flatten, SimpleRNN, LSTM, GRU
목표및 내용	고수준 API에서 케라스의 Dense, Flatten, SimpleRNN, LSTM, GRU를 활용해 딥러닝을 구현할 수 있다.
미리읽어오기	Dense, Flatten, SimpleRNN, LSTM, GRU
과제,시험,기타	딥러닝 구현 프로젝트 발표
15 주차	[기말고사 기간 : 12/14(월)~23(수)]
학습주제	직무수행능력평가 2차(기말고사)
목표및 내용	직무수행능력평가
미리읽어오기	
과제,시험,기타	

수업지원 안내	장애학생을 위한 별도의 수강 지원을 받을 수 있습니다. 언어가 문제가 되는 학생은 글로 된 과제 안내, 확대문자 시험지 제공 등의 지원을 드립니다.

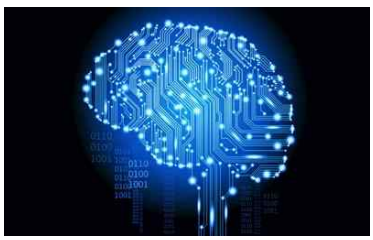
<텐서플로 기초>

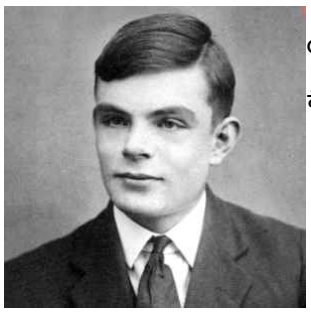
[파이썬]

파이썬은 현재 가장 많이 사용되고 있는 프로그래밍 언어중 하나로서 확장성과 간결함으로 인공지능 관련 기술 코딩에 유용함을 보임

[인공지능 (AI)]

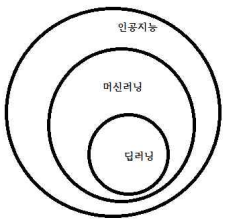
인간이 지닌 지적 능력의 일부 또는 전체, 혹은 그렇게 생각되는 능력을 인공적으로 구현한 것
머신러닝과 딥러닝은 인공지능 기술의 한 범주.



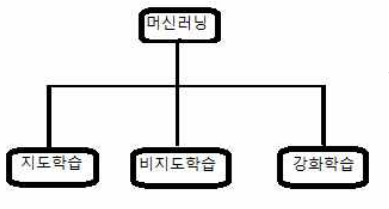


앨런 튜링 (1912-1954)
현대 컴퓨터과학의 아버지, 애니그마 해독, 튜링 머신

[머신러닝]

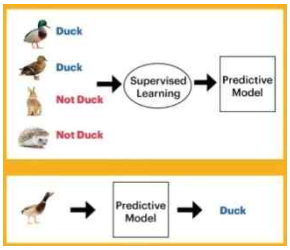


인공지능의 연구 분야 중 하나로, 인간의 학습 능력과 같은 기능을 컴퓨터에서 실현하고자 하는 기술 및 기법.
기계학습이라고도 불리며 기계가 스스로 학습하여 성능을 향상시키거나 최적의 해답을 찾기 위한 학습 지능 방법



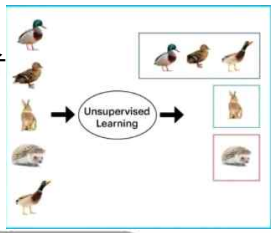
머신러닝은 크게 지도학습, 비지도학습, 강화학습 세가지 부류가 존재한다.

지도학습



올바른 입력과 출력의 쌍으로 구성된 정답의 훈련 데이터로부터 입출력 간의 함수를 학습시키는 방법 (k-최근접 이웃, 선형 회귀 등)

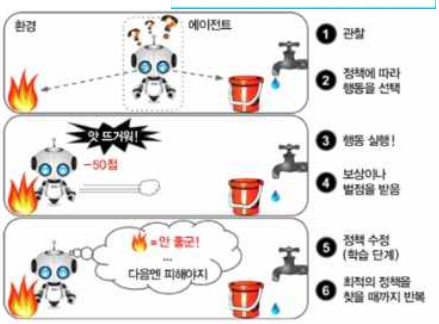
비지도학습



정답이 없는 훈련 데이터를 사용하여 데이터 내에 숨어있는 어떤 관계를 찾아내는 방법 (클러스터링 등)

강화학습

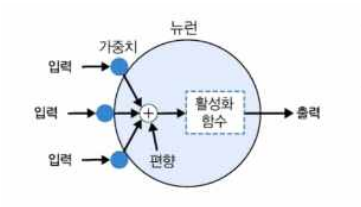
잘한 행동에 대해 보상을 주고 잘못한 행동에 대해 벌을 주는 경험을 통해 지식을 학습하는 방법
(딥마인드의 알파고 등)



머신 러닝과 딥 러닝의 차이점

	기계 학습	딥 러닝
데이터 의존성	중소형 데이터 세트에서 탁월한 성능	큰 데이터 세트에서 뛰어난 성능
하드웨어 의존성	저가형 머신에서 작업하십시오.	GPU가있는 강력한 기계가 필요합니다. DL은 상당한 양의 행렬 곱셈을 수행합니다.
기능 공학	데이터를 나타내는 기능을 이해해야 함	데이터를 나타내는 최고의 기능을 이해할 필요가 없습니다
실행 시간	몇 분에서 몇 시간	최대 몇 주. 신경망은 상당한 수의 가중치를 계산해야합니다.

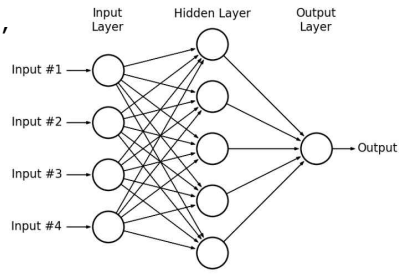
[퍼셉트론]



뉴런의 수학적 모델을 일컫는 말이기도 하며 최초로 제안된 신경망 프로그램 알고리즘 이기도 함

[MLP (Multi Layer Perceptron)]

여러개의 퍼셉트론으로 구성된 신경망 개념이며, 입력층과 출력층 그리고 중간의 은닉층으로 구성되어 있다.

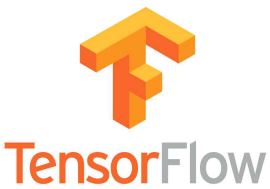


[구글 딥마인드, 알파고]

알파고는 구글 딥마인드에서 개발한 인공지능 바둑 프로그램.
인터넷상에 있는 3000만건의 기보 데이터를 기반으로 1차적으로 학습하였음

[케라스]

원래는 독자적인 고수준 라이브러리 였으나 현재는 Tensorflow의 고수준 API로 사용됨



[텐서플로]

구글에서 만든 연구 및 프로덕션용 오픈소스 딥러닝 라이브러리, 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 라이브러리, 기능 제공
Python을 최우선으로 지원하는 것이 특징

[Tensor (텐서)]

딥러닝에서 데이터를 표현하는 방식
0-D 텐서 : 스칼라
1-D 텐서 : 벡터
2-D 텐서 : 행렬
그리고 텐서는 행렬로 표현할 수 있는 n차원의 형태의 배열을 높은 차원으로 확장함

[Session과 코랩 1.0 코딩]

```
%tensorflow_version 1.x
TensorFlow 1.x selected.

[ ] #텐서플로 1.0 버전 선택
try:
  #%tesnsorflow_version only exists in Colab.
  %tensorflow_version 1.x
except Exception:
  pass

import tensorflow as tf
tf.__version__

'1.15.2'

[ ] import tensorflow as tf

hello = tf.constant('Hello World!')

sess = tf.Session()
print(sess.run(hello))
sess.close()

b'Hello World!'

[ ] a=10
b=10
print(tf.add(a,b))

sess = tf.Session()
print(sess.run(tf.add(a,b)))
sess.close()

Tensor("Add:0", shape=(), dtype=int32)
20
```

텐서플로우에서 텐서는 그래프라고 부르는 객체 내에 저장되어 실행됨
그래프를 계산하려면 외부 컴퓨터에 이 그래프 정보를 전달하고 결과값을 받아야함.
이 통신과정을 담당하는 것이 세션.

코랩 1.x버전에서는 그래프를 실행시키는 객체인 세션을 수동으로 열고 닫아야만 하였다.

- tf.constant : 변수를 설정하는 기본명령어
- shape : 텐서의 형태를 나타냄
- dtype : 텐서의 자료형을 나타냄
- numpy : 텐서의 값을 나타냄

[조건 연산 tf.cond()]

tf.cond

```
[ ]  x=tf.constant(1.)
     bool = tf.constant(False)
     res = tf.cond(bool,lambda:tf.add(x, 1.),lambda: tf.add(x, 10.))

     print(res)
     print(res.numpy())
```

```
tf.Tensor(11.0, shape=(), dtype=float32)
11.0
```

tf.cond(pred, true_fn=None, false_fn=None, name=None)

- pred를 검사해 참이면 true_fc 반환
- pred를 검사해 거짓이면 false_fc 반환

[텐서의 브로드캐스팅 코드]

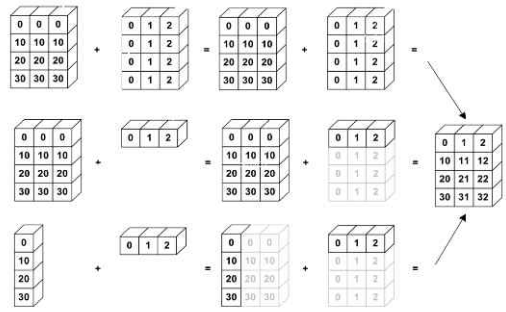
```
[ ]  x= tf.constant([[0], [10], [20], [30]])
     y= tf.constant([0, 1, 2])

     print((x+y).numpy())
```

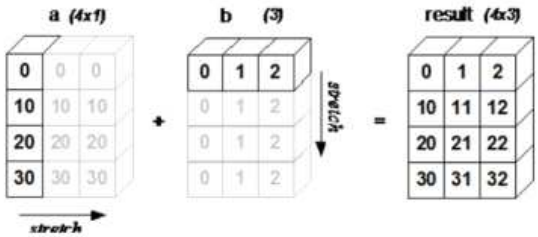
```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

```
[ ]  import numpy as np

     print(np.arange(3))
     print(np.ones((3, 3)))
     print()

     x=tf.constant((np.arange(3)))
     y=tf.constant([5], dtype=tf.int64)
     print(x)
     print(y)
     print(x+y)
```

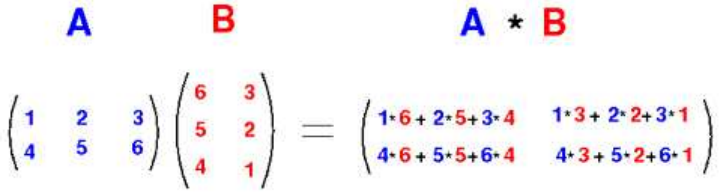
```
[0 1 2]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]

tf.Tensor([0 1 2], shape=(3,), dtype=int64)
tf.Tensor([5], shape=(1,), dtype=int64)
tf.Tensor([5 6 7], shape=(3,), dtype=int64)
```

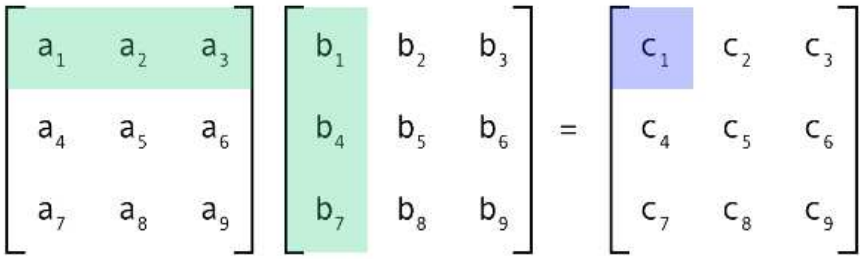
브로드캐스팅은 산술 연산 중에 크기가 다른 두개의 배열을 다루어 호환이 가능한 하나의 형태로 만드는 것





[텐서플로의 행렬 곱]



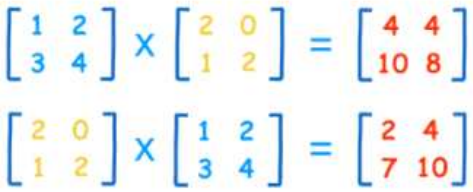


$$C_{ij} = \sum_k A_{ik} B_{kj} = A_{ik} B_{kj}$$

[tf.matmul()]

```
[50] # Matrix multiplications 1
     matrix1 = tf.constant([[1., 2.], [3., 4.]])
     matrix2 = tf.constant([[2., 0.], [1., 2.]])

     gop = tf.matmul(matrix1, matrix2)
     print(gop.numpy())

     # Matrix multiplications 2
     gop = tf.matmul(matrix2, matrix1)
     print(gop.numpy())
```

```
[[ 4.  4.]
 [10.  8.]]
[[ 2.  4.]
 [ 7. 10.]]
```



matmul은 두 배열의 행렬곱을 할 때 사용하는 함수

<텐서플로의 심화 연산>

[tf.rank]

```
[ ]  my_image = tf.zeros([2, 5, 5, 3])
     my_image.shape

     TensorShape([2, 5, 5, 3])
```

변수의 배열이 몇차원 인지를 나타냄.

```
[ ]  tf.rank(my_image)

     <tf.Tensor: shape=(), dtype=int32, numpy=4>
```

```
[ ]  tf.rank(my_image).numpy()

     4
```

[shape, reshape]

```
[ ]  rank_three_tensor = tf.ones([3, 4, 5])
     rank_three_tensor.shape

     TensorShape([3, 4, 5])
```

shape는 변수의 배열 형태를 알려줌

해당 모양의 배열을 생성

```
[ ]  rank_three_tensor.numpy()

     array([[[1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.]],

            [[1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.]],

            [[1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.],
             [1., 1., 1., 1., 1.]]], dtype=float32)
```

```
[ ]  #기존 내용을 6x10 행렬로 형태 변경
     matrix = tf.reshape(rank_three_tensor, [6, 10])
     matrix

     <tf.Tensor: shape=(6, 10), dtype=float32, numpy=
     array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
            [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
            [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
            [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
            [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
            [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]], dtype=float32)>
```

reshape 함수는 배열의 모양을 명시된 형태로 변경함.

```
#기존 내용을 3x20 행렬로 형태 변경
# -1은 차원 크기를 계산하여 자동으로 결정하라는 의미
matrixB = tf.reshape(matrix, [3,-1])
matrixB
```

```
<tf.Tensor: shape=(3, 20), dtype=float32, numpy=
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.]], dtype=float32)>
```

shape에 -1을 명시하면 차원의 크기를 자동적으로 계산하여 자동적으로 reshape 하게 됨

[tf.cast]

텐서를 새로운 형태로 캐스팅하는데 사용되는 함수.
(실수 등)

```
[ ]  #정수형 텐서를 실수형으로 변환
     float_tensor = tf.cast(tf.constant([1, 2, 3]), dtype=tf.float32)
     float_tensor
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dtype=float32)>
```

```
[ ]  float_tensor.dtype
```

```
tf.float32
```

<텐서플로 난수>

[균등분포 난수]

```
#3.7 랜덤한 수 얻기 (균일 분포)
rand = tf.random.uniform([1],0,1)
print(rand)
```

```
tf.Tensor([0.7586086], shape=(1,), dtype=float32)
```

```
[ ]  rand = tf.random.uniform([5,4],0,1)
     print(rand)
```

```
tf.Tensor(
[[0.25572407 0.57340646 0.77737606 0.4770198 ]
 [0.35867608 0.56122077 0.36773717 0.21443188]
 [0.35423803 0.10888362 0.5510478  0.9368199 ]
 [0.07648396 0.9394914  0.2570548  0.65446043]
 [0.7996371  0.5505259  0.4201342  0.64448166]], shape=(5, 4), dtype=float32)
```

```
[ ]  rand = tf.random.uniform([1000],0,10)
     print(rand[:10])
```

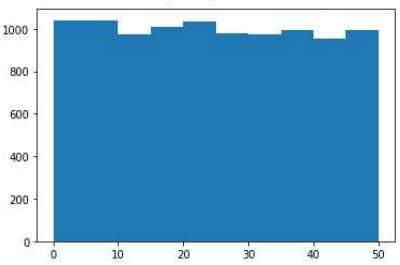
```
tf.Tensor(
[9.723716  2.601993  5.147884  8.559637  0.9290886 4.942019  1.7831147
 3.9651    0.607425  0.5376482], shape=(10,), dtype=float32)
```

균일분포 10000개를 추출하여 그래프로 만든 모습

균등(균일)분포란 주어진 범위 내의 모든 수가 동일한 분포를 갖는 형태를 말함

```
import matplotlib.pyplot as plt
rand = tf.random.uniform([10000],0,50)
plt.hist(rand, bins=10)
```

```
(array([1040., 1041.,  973., 1010., 1035.,  981.,  976.,  994.,  956.,
         994.]),
 array([2.3841858e-04, 4.9999962e+00, 9.9997540e+00, 1.4999513e+01,
        1.9999269e+01, 2.4999027e+01, 2.9998787e+01, 3.4998543e+01,
        3.9998302e+01, 4.4998058e+01, 4.9997818e+01], dtype=float32),
 <a list of 10 Patch objects>)
```



[정규분포 난수]

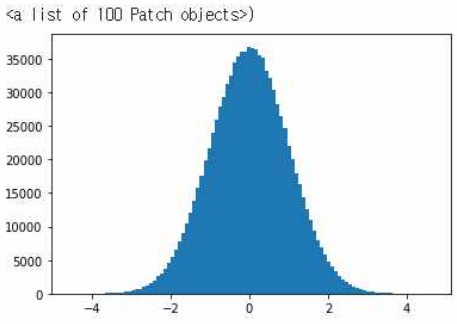
```
# 3.9 랜덤한 수 여러 개 얻기 (정규 분포)
rand = tf.random.normal([2, 4],0,2)
print(rand)
```

```
tf.Tensor(
[[ 0.9057808  -4.2003303   1.3747034   1.9594638 ]
 [ 1.7361418  -2.0234852  -0.33626932 -3.1894848 ]], shape=(2, 4), dtype=float32)
```

```
import matplotlib.pyplot as plt
rand = tf.random.normal([1000000], 0, 1)
plt.hist(rand,bins=100)
```

```
(array([2.0000e+00, 2.0000e+00, 2.0000e+00, 6.0000e+00, 3.0000e+00,
        6.0000e+00, 1.4000e+01, 2.2000e+01, 2.7000e+01, 3.4000e+01,
        6.0000e+01, 7.2000e+01, 1.1300e+02, 1.3400e+02, 2.0000e+02,
        2.3400e+02, 3.5500e+02, 4.5500e+02, 6.0500e+02, 7.2000e+02,
```

정규분포란 연속 확률 분포 중 하나로서 0에 가까운 값을 더 많이 갖게 되는 난수 (0에서 멀수록 확률이 급격히 낮아짐)



[shuffle]

```
import numpy as np
a = np.arange(10)
print(a)
tf.random.shuffle(a)
```

```
[0 1 2 3 4 5 6 7 8 9]
<tf.Tensor: shape=(10,), dtype=int64, numpy=array([5, 7, 1, 6, 8, 0, 3, 9, 4, 2])>
```

```
import numpy as np
a = np.arange(20).reshape(4,5)
a
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

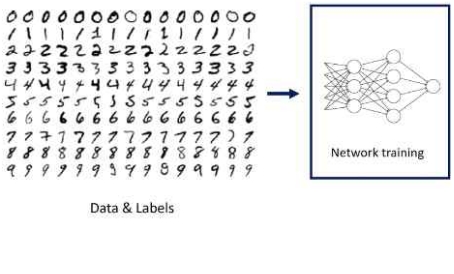
```
tf.random.shuffle(a)
```

```
<tf.Tensor: shape=(4, 5), dtype=int64, numpy=
array([[ 5,  6,  7,  8,  9],
       [15, 16, 17, 18, 19],
       [10, 11, 12, 13, 14],
       [ 0,  1,  2,  3,  4]])>
```

난수의 위치, 순서를 섞어주는 함수

<MNIST 데이터셋>

[MNIST 데이터셋]



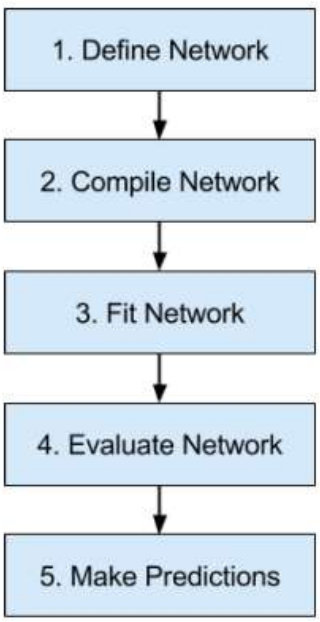
0 1 2 3 4 5 6 7 8 9

0부터 9까지의 손글씨 숫자 이미지로 이루어져 있는 데이터셋으로 딥러닝 손글씨 인식에 사용된다.

28X28=784 픽셀로 이루어져 있으며 각각의 내부 값은 0~255, 이 값을 0~1로 수정하여 사용한다.

[케라스 딥러닝 구현 5개 과정]

1. 딥러닝 모델을 define 하여
2. 주요 훈련 방법을 compile
3. 그 훈련을 시켜서 (fit)
4. 테스트 데이터를 evaluate 하고
5. 정답을 predict 한다.



```
[ ]  # MNIST 데이터셋을 훈련과 테스트 데이터로 로드하여 준비
     (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

훈련 데이터 손글씨와 정답 쌍 60000개와 테스트 데이터 손글씨와 정답 쌍 10000개를 로드

```
[ ]  # MNIST 형태를 알아 봅시다. 데이터 수, 행렬 형태 등
     print(x_train.shape, y_train.shape)
     print(x_test.shape, y_test.shape)
```

```
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)
```

첫 손글씨 데이터와 정답 예

```
x_train.shape
```

```
(60000, 28, 28)
```

```
[ ]  y_train.shape
```

```
(60000,)
```

```
x_train[0]
```

```
...  0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
 205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
  90,   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, 139, 253,
 190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

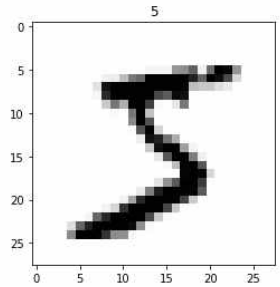
```
import sys

for x in x_train[0]:
  for i in x:
    sys.stdout.write('%3d' %i)
  sys.stdout.write('\n')
```

```
  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   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   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   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   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   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0
  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0
  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0
  0   0   0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0 136 253 253 253 212 135 132  16   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   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   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
```

첫 번째 손글씨 데이터의 정답 행렬 내용을 직접 출력한 것

```
import matplotlib.pyplot as plt

n = 0
ttl = str(y_train[n])
plt.figure(figsize=(6,4))
plt.title(ttl)
plt.imshow(x_train[n], cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7fc48fb83f98>
```



<딥러닝의 구현순서>

가. 0 필요 모듈 임포트

나. 훈련과 정답 데이터 지정

1) 데이터 전처리 (옵션)

다. 모델 구성

라. 학습에 필요한 최적화 방법과 손실 함수 등 설정

1) 구성된 모델 요약(옵션)

마. 생성된 모델로 훈련 데이터 학습

바. 테스트 데이터로 성능 평가

1) 테스트 데이터 또는 다른 데이터로 결과 예측 (옵션)

[훈련과 정답 데이터 지정 + 데이터 전처리]

```
import tensorflow as tf
# mnist 모듈 준비
mnist = tf.keras.datasets.mnist
# MNIST 데이터셋을 훈련과 테스트 데이터로 로드하여 준비
(x_train, y_train), (x_test, y_test) = mnist.load_data()
# 샘플 값을 정수(0~255)에서 부동소수(0~1)로 변환
x_train, x_test = x_train / 255.0, x_test / 255.0
```

[모델 구성]

```
# 층을 차례대로 쌓아 tf.keras.Sequential 모델을 생성
model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(28, 28)),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(10, activation='softmax')
])
```

Flatten - 평탄화
Dense - 완전연결층
Dropout - 뉴런 연결해제로 모델 단순화

[훈련에 사용할 옵티마이저와 손실 함수 선택]

```
# 훈련에 사용할 옵티마이저(optimizer)와 손실 함수, 출력정보를 선택
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

옵티마이저 - 입력된 데이터와 손실 함수를 기반으로 모델을 업데이트 하는 메커니즘

손실 함수 - 훈련 데이터에서 신경망의 성능을 측정하는 방법으로 모델이 옳은 방향으로 학습될 수 있도록 도와주는 기준 값

[모델 요약]

```
# 모델 요약 표시
model.summary()
```

[모델 훈련]

```
# 모델을 훈련 데이터로 총 5번 훈련
model.fit(x_train, y_train, epochs= 5)
```

epochs 에 훈련 횟수 지정

[모델 평가]

```
# 모델을 테스트 데이터로 평가
model.evaluate(x_test, y_test)
```

[실행 결과]

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_3 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2961 - accuracy: 0.9137
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1425 - accuracy: 0.9570
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1074 - accuracy: 0.9674
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0879 - accuracy: 0.9728
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0741 - accuracy: 0.9769
313/313 [==============================] - 0s 1ms/step - loss: 0.0787 - accuracy: 0.9759
[0.07873814553022385, 0.9758999943733215]
```

[원-핫 인코딩]

원-핫 인코딩은 단어 집합의 크기를 벡터의 차원으로 하고 표현하고 싶은 단어의 인덱스에 1의 값, 다른 인덱스에 0을 부여하는 인코딩 방식

ID	과일
1	사과
2	바나나
3	체리

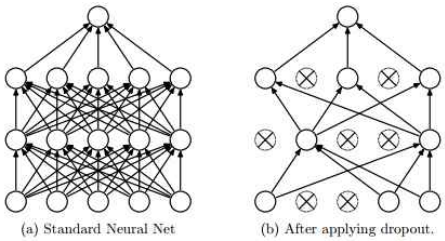
One-Hot Encoding

ID	사과	바나나	체리
1	1	0	0
2	0	1	0
3	0	0	1

LabelEncoder

ID	과일
1	0
2	1
3	2

[드롭아웃]

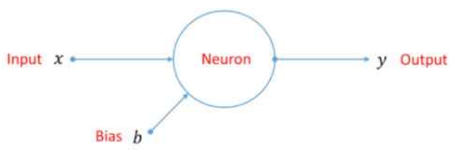


층에서 결과 값을 일정 비율로 제거하는 방법
신경망 모델이 복잡해질 때 뉴런의 연결을 임의로 삭제하여 결과를 도출

BUT, 테스트 단계에서는 어떤 유닛도 드롭아웃 X

<인공신경망>

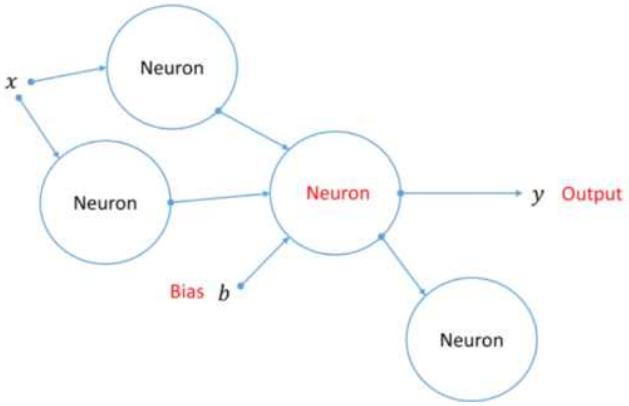
[뉴런]



생물학적인 뉴런을 수학적으로 모델링 한 것.
다른 여러개의 뉴런으로부터 입력값을 받아 자신의 용량을 넘어서면 외부로 출력값을 보내게됨

$$\sigma = w \cdot x + b$$

[신경망, 편향]

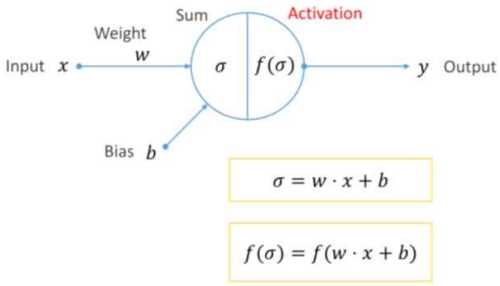


뉴런들의 연결체

편향은 조절하여 출력을
맞춰주는 보조적인 역할을 함

Input x	Output y
Size of house	Price
Time spent for studying	Score in exam

[활성화 함수]

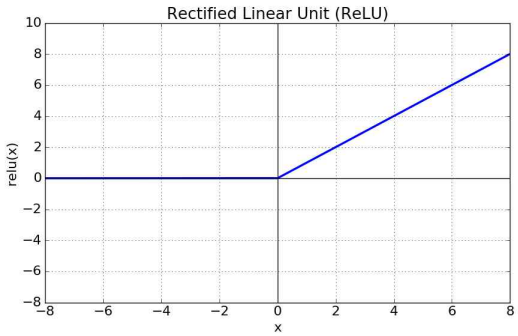


$$\sigma = w \cdot x + b$$

$$f(\sigma) = f(w \cdot x + b)$$

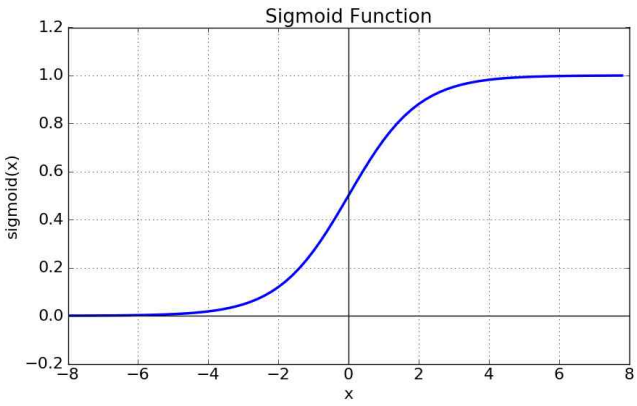
활성화 함수란 뉴런의 출력값을 정하는
함수로서 다양한 형태를 지님

[ReLU 함수]



최근 가장 많이 사용되는 활성화 함수
Rectified(정류된) Linear Unit (선형함수)
max(x, 0) 으로 표현됨.

[Sigmoid 함수]



S자 형태의 곡선이라는 의미로 Logistic 함수라고도 불림.
선형인 멀티퍼셉트론에서 비선형 값을 얻기 위해 사용하기 시작됐었음

그 외로 Maxout, tanh 함수 등이 존재함

<회귀와 분류>

[회귀 모델]

연속적인 값을 예측

[분류 모델]

불연속적인 값을 예측

[회귀 분석]

관찰된 연속형 변수들에 대해 변수 사이의 모형을 구한 뒤 적합도를 측정해 내는 방법

[단순 선형 회귀]

$$H(x) = Wx + b$$

단순 선형 회귀(Simple Linear Regression)은 입력시 특징이 하나이고 출력시 하나의 값을 출력

[다중 선형 회귀]

$$y = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$$

입력시 특징이 여러개이며 하나의 값을 출력

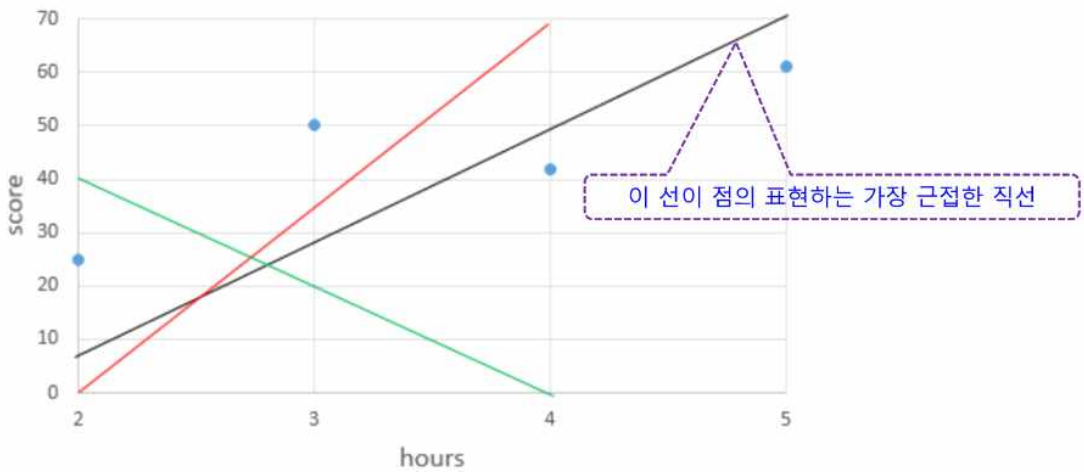
[로지스틱 회귀]

회귀를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1사이의 값으로 예측하고 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘 => 이진 분류 (Binary Classification)

[가설(Hypothesis)]

$$H(x) = Wx + b$$

W:기울기, 가중치
b: 절편, 편향



가중치와 편향/ 기울기와 절편
선형 회귀에서 해야할 일은 결국 적절한 W와 b를 찾아내는 일

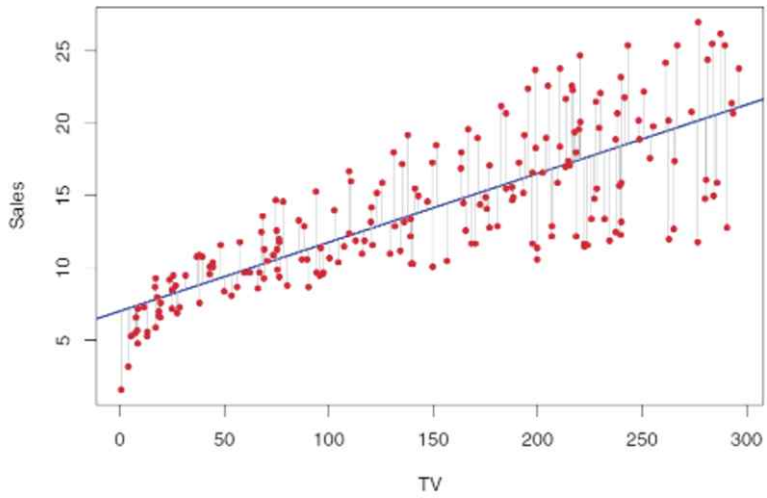
[손실 함수(Loss function)]

$$\frac{1}{n}\sum_{i}^{n}[y_i - H(x_i)]^2$$

실제 값 / 예측 값

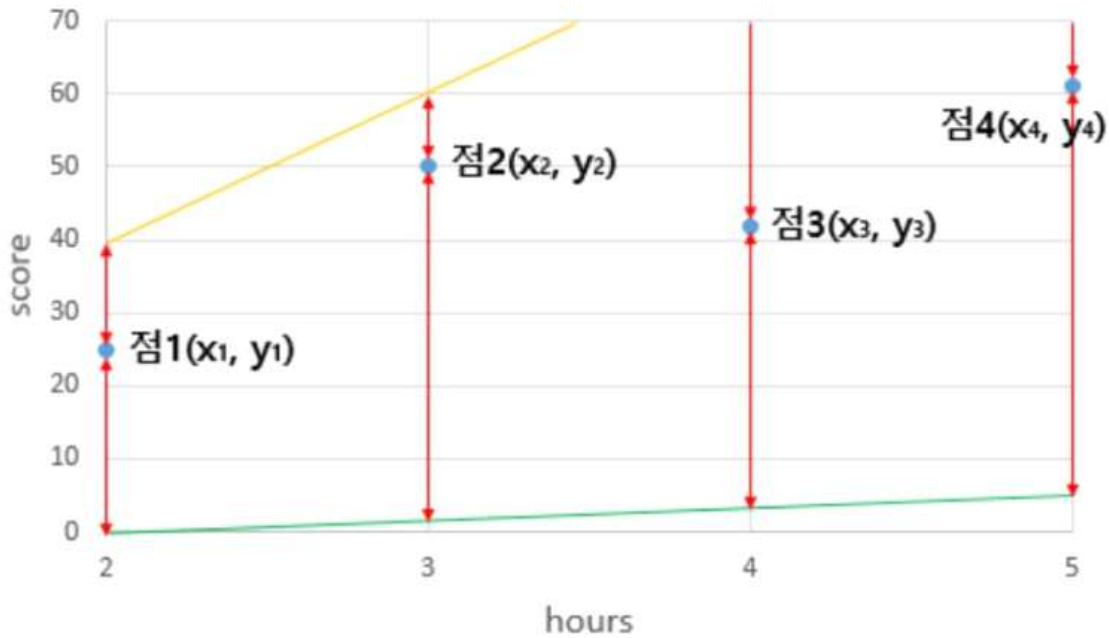
목적 함수 (Objective function) 비용 함수 라고도 불리며 실제 값과 예측 값에 대한 오차에 대한 식
=> 예측 값의 오차를 줄이는 일에 최적화

[MSE]



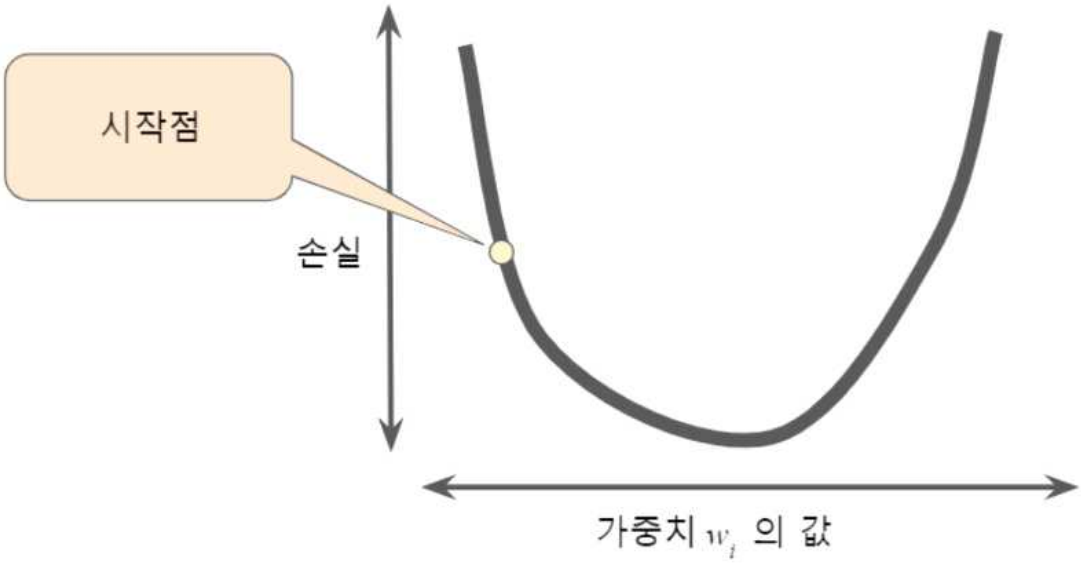
오차는 실제 데이터(빨간 점)와 예측 선(파란 선)의 차이의 제곱의 합

[옵티마이저 : 최적화 과정]



머신 러닝에서 학습은 최적화 알고리즘
적절한 W와 b를 찾아내는 과정임
위의 사례는 경사 하강법(Gradient Descent) 이며 비용 함수의 값을 최소로 하는 W와 b를 찾는 방법임

[경사하강법]

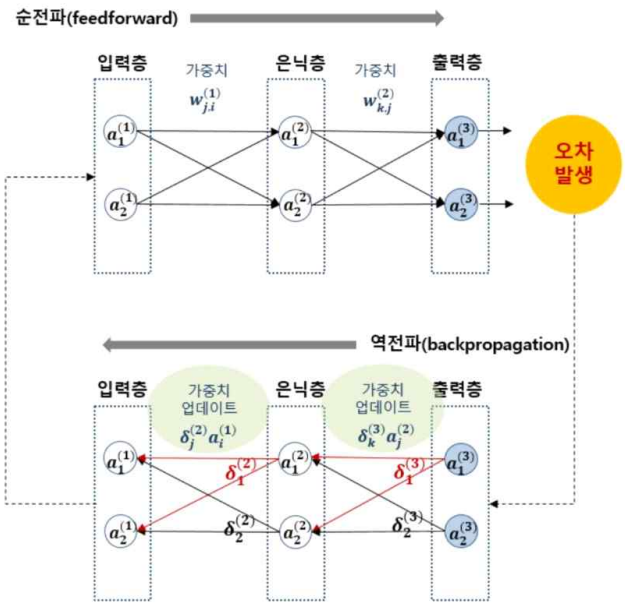


기울기가 0인 지점을 찾기 위하여 시작점을 선택, 곡선의 기울기를 계산하는 방법

[초매개변수]

딥러닝에서 우리가 설정하는 값
예를들어 학습률은 초매개변수 중 하나로 설정됨

[오차역전파]



순전파 - 입력층에서 출력층으로 계산해 최종 오차를 계산하는 방법

역전파 - 오차 결과 값을 통해서 다시 역으로 input 방향으로 오차가 적어지도록 다시 보내며 가중치를 수정하는 방법