

# 오리엔테이션

송태영

# 스터디 개요

- 목적 : 파이토치를 활용한 딥러닝 익숙해 지기
- 일정 : 6월 초 ~ 7월 말
- 스터디 진행 방법
  - 매회 발표자가 담당인 chapter를 요약 정리 (30분)
  - 의견 교환 (20분)
- 교재
  - [<Must Have> 텐초의 파이토치 딥러닝 특강](#)

# 교재 구성 및 스케줄

- 1주

- 1장 : 딥러닝 한눈에 살펴보기
- 2장 : 인공 신경망 ANN 이해하기

- 2주

- 3장 : 간단한 신경망 만들기
- 4장 : 사진 분류하기 : CNN과 VGG

- 3주

- 5장 : 유행 따라가기 : ResNet
- 6장 : 넷플릭스 주가 예측하기 :  
RNN으로 첫 시계열 학습

- 4주

- 7장 : 이미지 세그멘테이션 : U-Net
- 8장 : 이미지 노이즈 제거 : 오토인코더

- 5주

- 9장 : 자동 채색 : Let There be color 모델
- 10장 : 글쓰는 인공지능 : LSTM

- 6주

- 11장 : 직접 만드는 번역기 : 어텐션
- 12장 : 캡처 텍스트 인식 : CRNN+GRU

- 7주

- 13장 : 사람 얼굴 생성하는 GAN
- 14장 : 화질 개선하는 GAN

- 8주

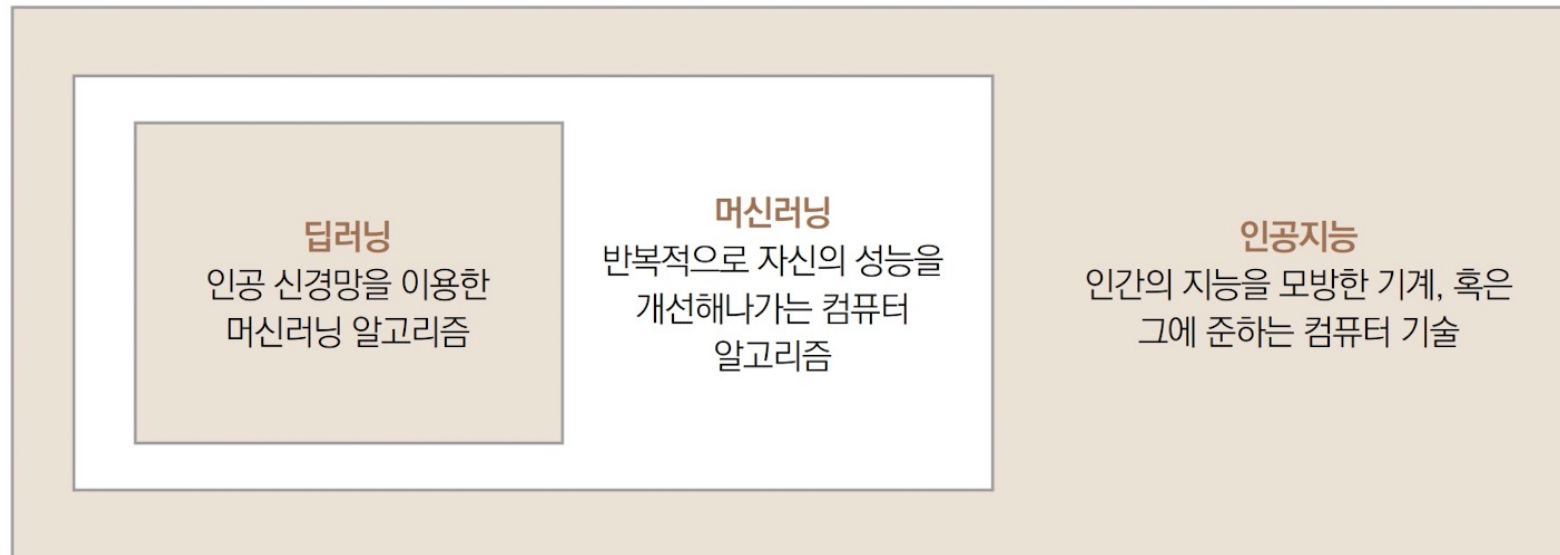
- 15장 : 데이터 없이 학습하는 GAN
- 부록 A : 트랜스포머, GPT, BERT, ViT

# 01 딥러닝 한눈에 살펴보기

# 머신러닝과 딥러닝

- 머신러닝 : 입력 데이터를 이용해 알지 못하는 변수를 반복적으로 학습해 나가면서 예측하는 알고리즘
- 딥러닝 : 인공신경망을 사용한 머신러닝 알고리즘

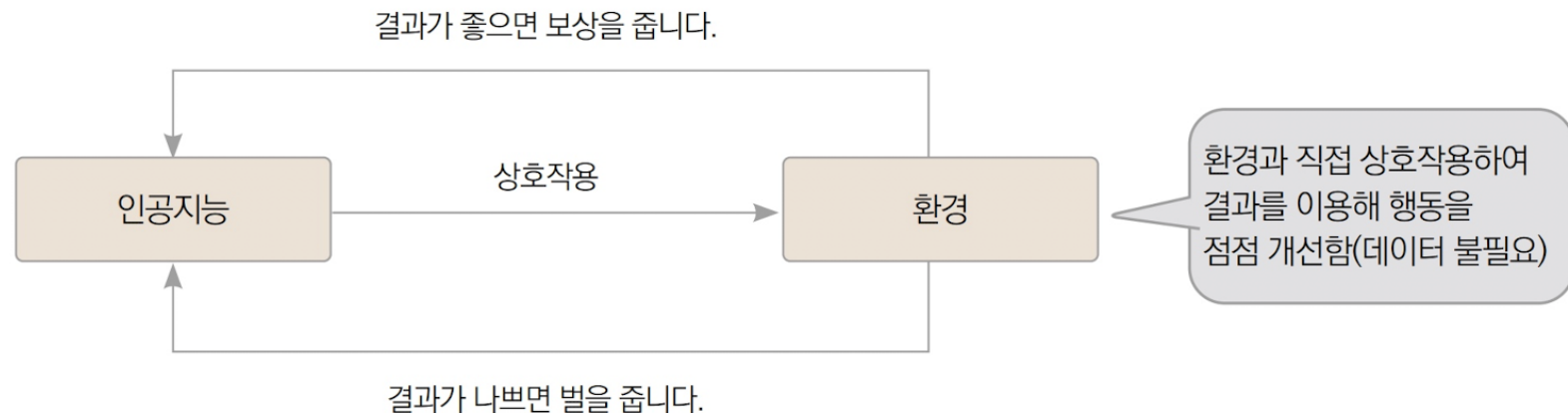
▼ 딥러닝, 머신러닝, 인공지능의 차이



# 지도학습, 비지도 학습, 강화학습

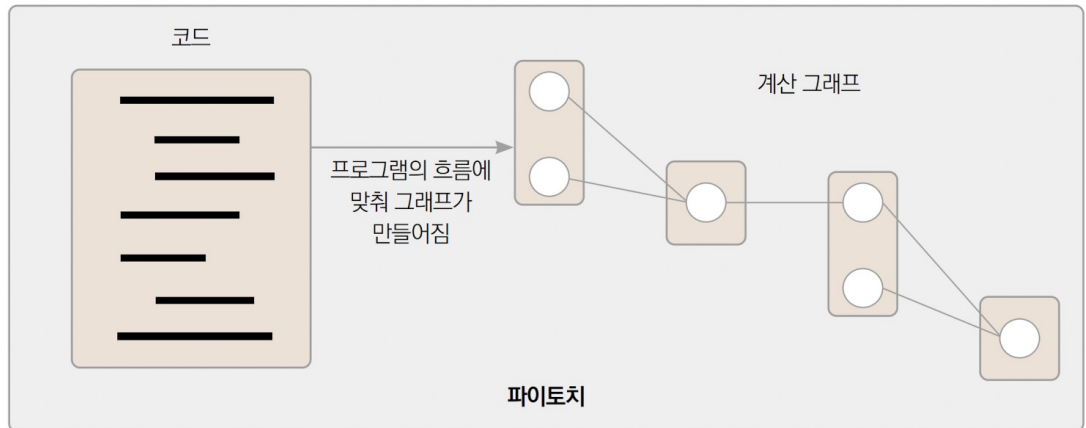
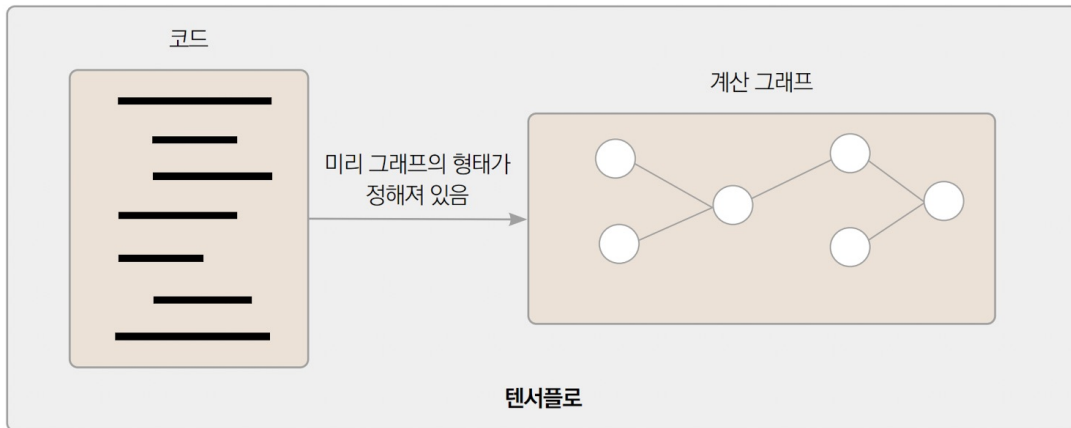
- 지도학습 : 학습용 데이터에 정답 레이블을 제공하는 학습 방법
- 비지도 학습 : 학습용 데이터에 정답 레이블을 제공하지 않는 학습 방법 (구매 이력이 있는 소비자를 구매 취향에 따라 분류)
- 강화 학습 : 인공지능이 환경으로부터 보상과 벌을 받으며 시행착오를 통해 스스로 학습 (e.g. 알파고, GAN)

## ▼ 강화 학습



# 왜 딥러닝에 파이토치인가

- 파이토치는 파이썬 본래의 코드와 유사, 직관적
- 정적 계산 그래프인 텐서플로우와 다르게, 파이토치는 동적으로 계산 그래프를 만들어, 계산 중간에 변수의 값을 수정 가능 (디버깅 용이)
- 동적인 변경이 필요 없을 때는, 정적 계산 그래프가 계산이 빠름



# 파이토치 코딩 권고 스타일

- 클래스 사용 : 모듈 클래스로 신경망을 만들고, 데이터셋 클래스로 데이터를 불러와 학습
- 데이터 로더 : 데이터셋 클래스를 입력 받아, 학습에 필요한 양만큼 데이터를 불러오는 역할

▼ 파이토치 신경망의 기본 구성

```
class Net(nn.Module):
    def __init__(self):
        ...

        # 신경망 구성요소 정의

        ...    미리 정의된 신경망 모듈 불러오기

    def forward(self, input):
        ...

        # 신경망의 동작 정의

        ...    __init__에서 정의한 모듈 연결

    return output
```

```
class Dataset():
    def __init__(self):
        ...

        필요한 데이터 불러오기
        ...

    def __len__(self):
        ...

        데이터의 개수 반환
        ...

        return len(data)

    def __getitem__(self, i):
        ...

        i번째 입력 데이터와
        i번째 정답을 반환
        ...

        return data[i], label[i]
```

```
# 데이터로더로부터 데이터와 정답을 받아옴
for data, label in DataLoader():
    # ❶ 모델의 예측값 계산
    prediction = model(data)

    # ❷ 손실 함수를 이용해 오차 계산
    loss = LossFunction(prediction, label)

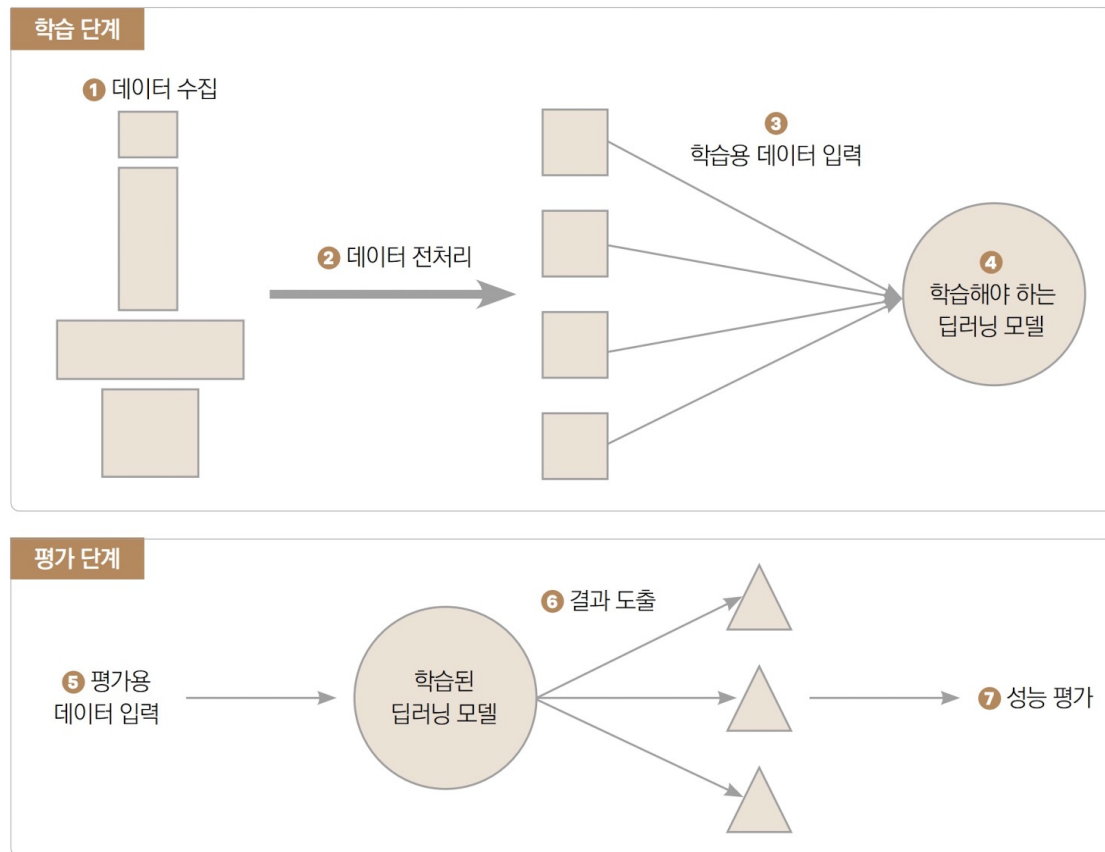
    # ❸ 오차 역전파
    loss.backward()

    # ❹ 신경망 가중치 수정
    optimizer.step()
```



# 딥러닝 문제 해결 프로세스

## ▼ 딥러닝 모델 학습의 흐름



# 딥러닝 문제 해결 체크리스트

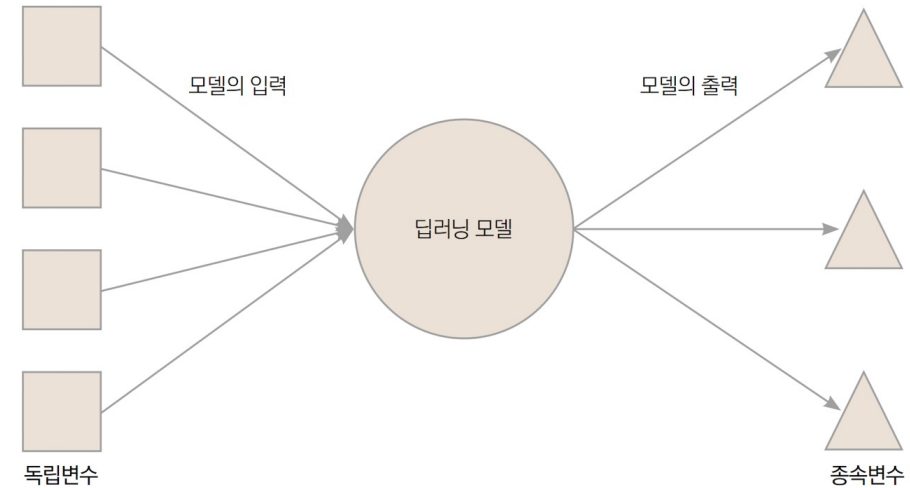
## ▼ 딥러닝 문제 해결 체크리스트

| 문제 풀이 단계       | 항목                                                                                                                                                             |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 풀어야 할 문제 이해하기  | <input type="checkbox"/> 정확한 값을 예측하는 회귀 문제인가?<br><input type="checkbox"/> 입력이 속한 범주를 예측하는 분류 문제인가?                                                             |
| 데이터 파악하기       | <input type="checkbox"/> 입력 자료형과 정답 확인하기<br><input type="checkbox"/> 클래스 간의 불균형은 없는지 확인하기<br><input type="checkbox"/> 누락된 데이터 혹은 자료형에 맞지 않는 데이터가 포함되어 있는지 확인하기 |
| 데이터 전처리        | <input type="checkbox"/> 학습에 필요한 데이터가 부족하다면 데이터 증강하기<br><input type="checkbox"/> 데이터를 정규화해서 값의 범위 맞추기                                                          |
| 신경망 설계         | <input type="checkbox"/> 데이터의 공간 정보가 중요하면 합성곱 적용하기<br><input type="checkbox"/> 데이터의 순서 정보가 중요하면 RNN 적용하기                                                       |
| 신경망 학습         | <input type="checkbox"/> 적합한 손실 함수 찾기<br><input type="checkbox"/> 가중치 수정을 위한 최적화 정하기<br><input type="checkbox"/> 신경망의 성능을 평가하기 위한 평가 지표 정하기                    |
| 손실이 무한대로 발산한다면 | <input type="checkbox"/> 손실 함수 바꿔보기<br><input type="checkbox"/> 데이터에 이상한 값이 섞여 있는지 확인하기<br><input type="checkbox"/> 학습률 줄이기                                    |
| 손실이 0으로 수렴한다면  | <input type="checkbox"/> 데이터가 부족하지 않은지 확인하기<br><input type="checkbox"/> 신경망 크기 줄여보기                                                                            |

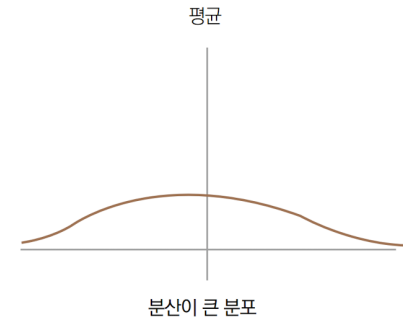
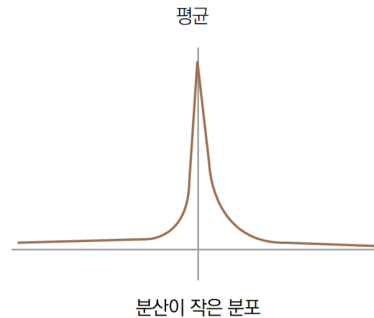
# 딥러닝에 필요한 최소한의 통계 개념

- 독립변수 : 다른 변수의 값을 결정하는 변수, 입력값
- 종속변수 : 독립변수에 의해 값이 결정되는 변수, 출력값
- 평균 : 모든 데이터의 합을 데이터의 수로 나눈 값
- 분산 : 데이터가 얼마나 퍼졌는가를 나타내는 지표
- 표준편차 : 분산의 제곱근

▼ 독립변수와 종속변수

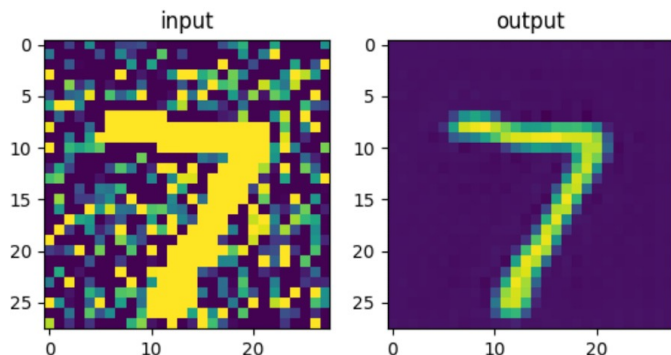


▼ 분산에 따른 데이터 분포의 차이

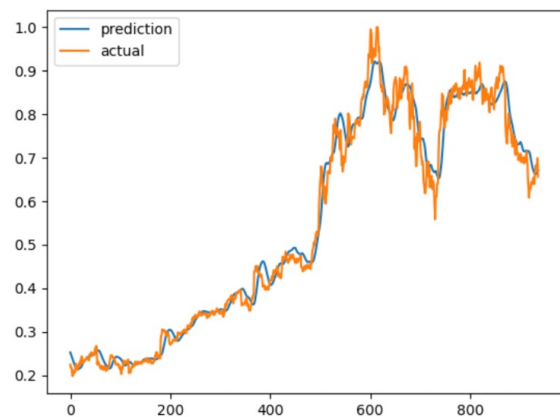


# 직관적 분석에 유용한 시각화

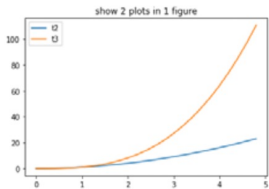
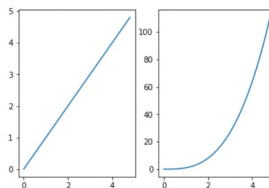
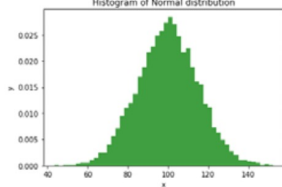
## ▼ 서브플롯 시각화 예



## ▼ 플롯 시각화 예



## ▼ 맷플롯립에서 지원하는 시각화 그래프

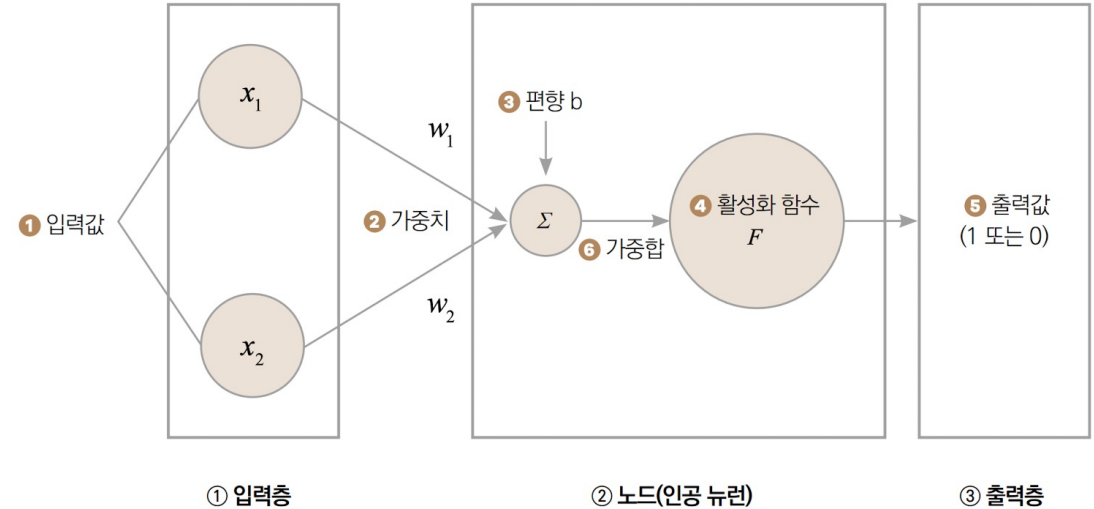
| 구분     | 예시 설명                                                                                                                                                                                                                                                                                | 사용하는 장        |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| 선형 그래프 | <p>여러 개의 선형 그래프를 하나의 그림에 그립니다.</p>  <pre>plt.plot(t, t2, label="t2") plt.plot(t, t3, label="t3") plt.title("show 2 plots in 1 figure") plt.legend() plt.show()</pre>                              | 6장            |
| 서브 플롯  | <p>그래프를 하나의 그림이 아니라 따로따로 볼 때 서브플롯을 이용합니다.</p>  <pre>plt.subplot(1, 2, 1) plt.plot(t, t) plt.subplot(1, 2, 2) plt.plot(t, t3) plt.show()</pre>                                                     | 4, 7, 8, 9장 등 |
| 히스토그램  | <p>데이터의 분포도가 확인할 때 히스토그램을 이용합니다.</p>  <pre>plt.xlabel('x') plt.ylabel('y') plt.title('Histogram of Normal distribution') plt.hist(x, 50, density=1, facecolor='g', alpha=0.75) plt.show()</pre> | 6장            |

## 02 인공신경망 ANN 이해하기

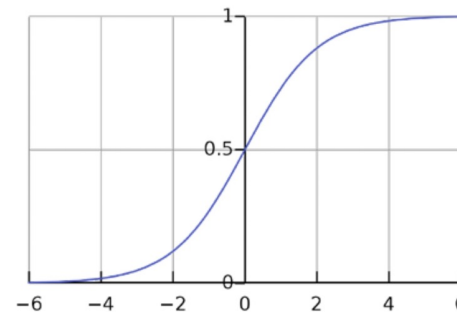
# 퍼셉트론

- 퍼셉트론 : 입력값과 가중치, 편향을 이용해 출력값을 내는 수학적 모델
- 입력값을 표현하는 입력층, 신경망 출력을 계산하는 출력층
- 노드는 입력값에 가중치를 곱해 활성화함수에 입력
- 가중치는 입력의 중요도, 편향은 활성화의 경계가 원점으로부터 얼마나 이동할지 결정
- 활성화 함수는 임계값을 기준으로 노드의 출력을 결정 하는 함수 (비선형성 추가)
- 시그모이드 함수는 뉴런의 출력값을 0과 1 사이로 고정 → 이진분류에 사용

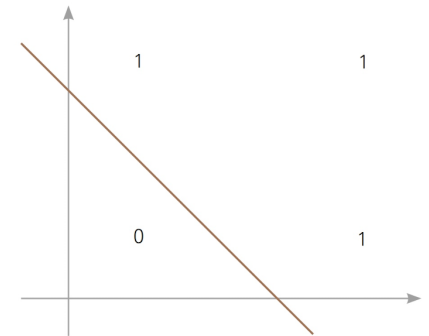
▼ 한눈에 보는 퍼셉트론 동작 원리



▼ 시그모이드 함수



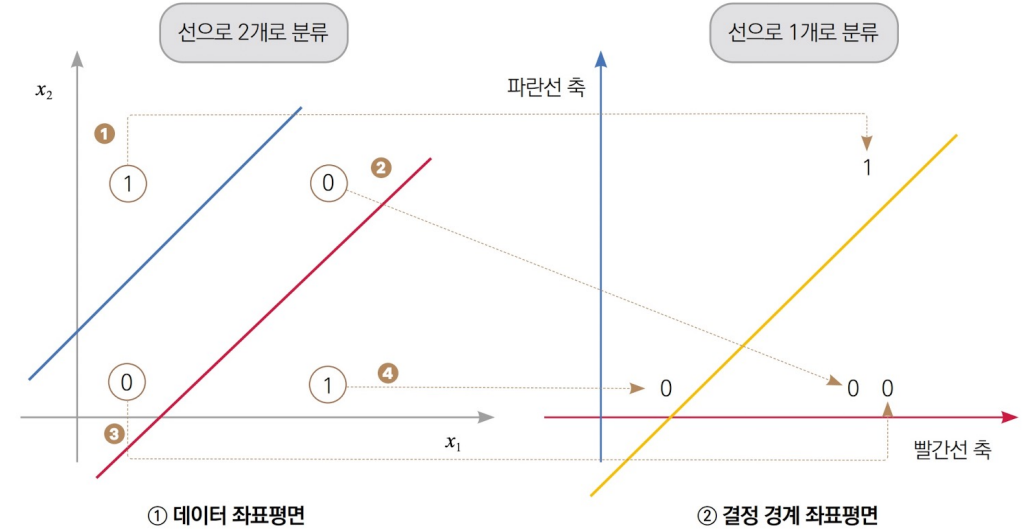
▼ 결정 경계를 이용한 데이터 분류의 예



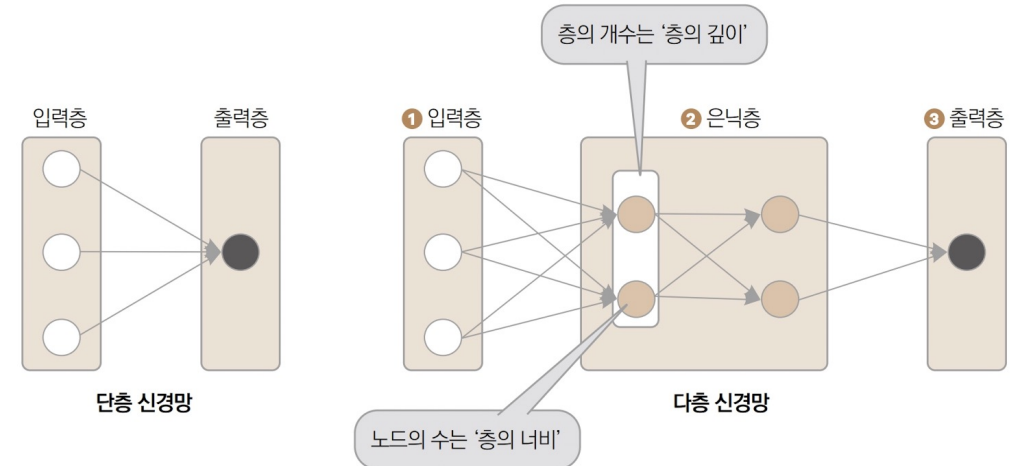
# 다층 신경망

- 다층 인공 신경망 : 퍼셉트론을 여러 개 사용하는 인공 신경망
- (파란, 빨간) 두 직선은 각각이 결정 경계, 선보다 위 값은 1, 아래 값은 0 반환
- 기저 벡터 변환 : 좌표계에서 점을 표현하는 기준(기저 벡터)를 다른 벡터로 변환
- 은닉층은 굳이 값을 알 필요가 없어 출력값을 숨긴다
- 순전파 : 데이터가 입력층으로 부터 출력층까지 순서대로 전달

▼ 다층 신경망이 XOR 데이터를 분류하는 방법



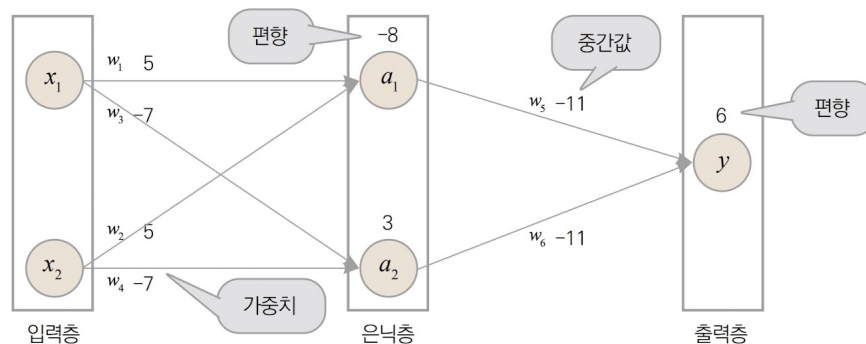
▼ 단층 신경망과 다층 신경망의 구분



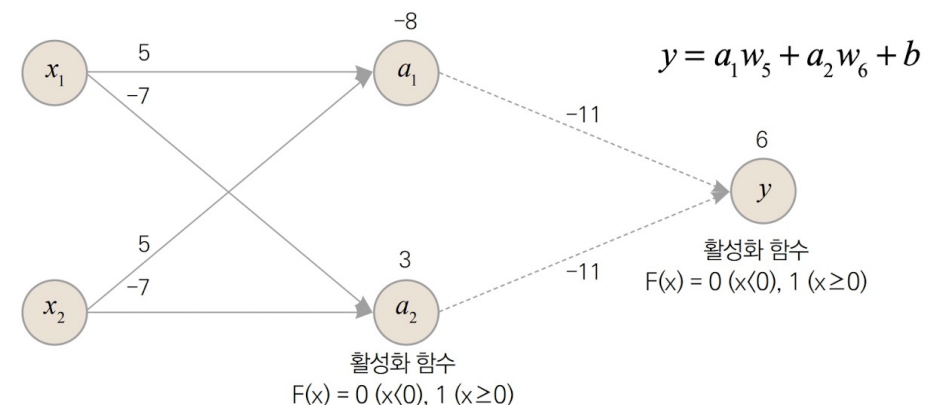
# 인공 신경망의 학습 방법

▼ 실습에 사용할 데이터

| $x_1$ | $x_2$ | $y$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 0     | 1     | 1   |
| 1     | 0     | 1   |
| 1     | 1     | 0   |



$$a = x_1 w_1 + x_2 w_2 + b$$



- 활성함수(F)는 0 이상이면 1출력, 0보다 작으면 0 출력

$$a_1 = F(5x_1 + 5x_2 - 8) = F(5 \times 0 + 5 \times 0 - 8) = F(-8) = 0$$

$$a_2 = F(-7x_1 - 7x_2 + 3) = F(-7 \times 0 - 7 \times 0 + 3) = F(+3) = 1$$

| $x_1$ | $x_2$ | $a_1$     | $a_2$      |
|-------|-------|-----------|------------|
| 0     | 0     | F(-8) = 0 | F(3) = 1   |
| 0     | 1     | F(-3) = 0 | F(-4) = 0  |
| 1     | 0     | F(-3) = 0 | F(-4) = 0  |
| 1     | 1     | F(2) = 1  | F(-11) = 0 |

- 학습 과정은,  $y'$  가  $y$ 와 유사 하도록 가중치를 찾아 가는 것

$$y' = F(-11a_1 - 11a_2 + 6) = F(-11 \times 0 - 11 \times 1 + 6) = F(-5) = 0$$

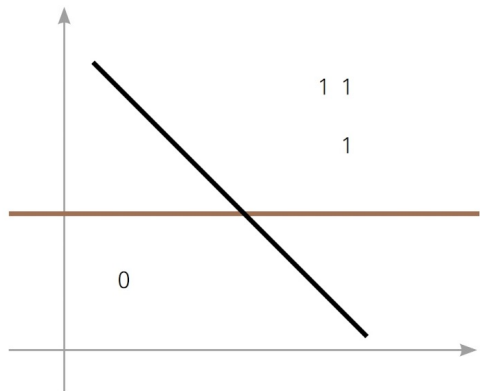
| $x_1$ | $x_2$ | $a_1$     | $a_2$      | $y'$       |
|-------|-------|-----------|------------|------------|
| 0     | 0     | F(-8) = 0 | F(3) = 1   | F(-5) = 0  |
| 0     | 1     | F(-3) = 0 | F(-4) = 0  | F(6) = 1   |
| 1     | 0     | F(-3) = 0 | F(-4) = 0  | F(6) = 1   |
| 1     | 1     | F(2) = 1  | F(-11) = 0 | F(-16) = 0 |



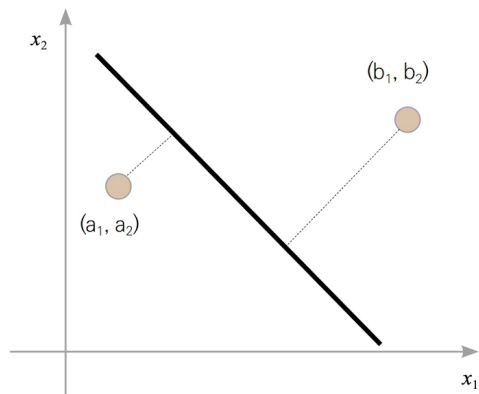
# 손실 함수

- 손실함수는 정답과 신경망의 예측의 차이를 나타내는 함수

▼ 두 결정 경계의 비교



▼ 결정 경계까지의 거리(오차)

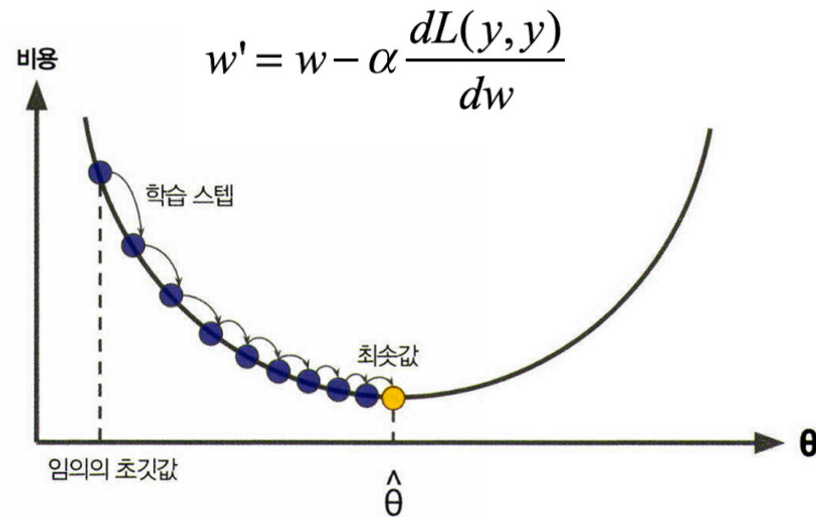


▼ 대표적인 손실 함수

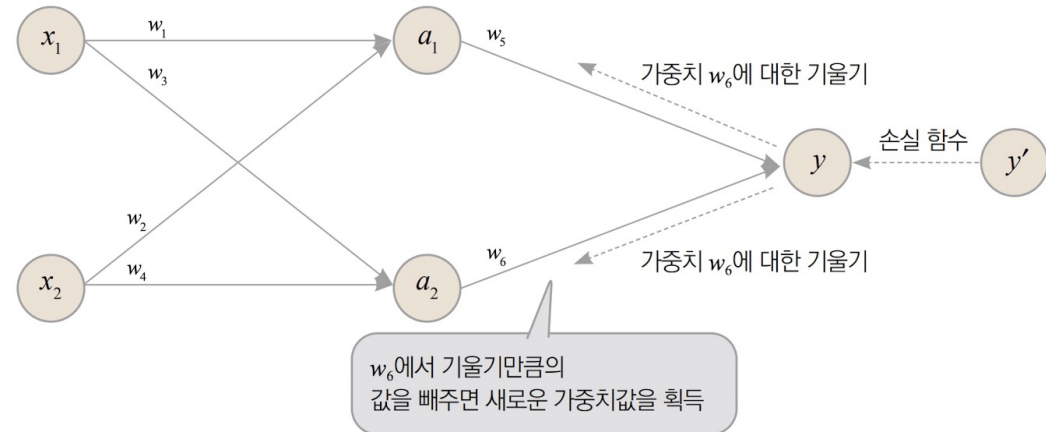
| 함수                                           | 용도            | 설명                                                                                                                             |
|----------------------------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------|
| 평균 제곱 오차<br>(Mean Squared Error, MSE)        | 회귀            | 정답과 예측값의 차의 제곱의 평균값. 1보다 작은 오차는 더 작게, 1보다 큰 오차는 더 크게 키우는 특성을 갖고 있습니다.                                                          |
| 크로스 엔트로피 오차<br>(Cross Entropy Error, CE)     | 이진분류,<br>다중분류 | 두 확률 분포의 차이를 구하는 함수. 분류 문제에서는 인공 신경망의 출력이 확률 분포이므로 확률 분포의 차를 구하는 함수가 필요합니다. 크로스 엔트로피는 정답값의 확률과 모델이 예측한 확률에 로그를 취한 값을 곱해서 구합니다. |
| 평균 절대 오차<br>(Mean Average Error, MAE)        | 회귀            | 정답과 예측값의 차의 절댓값의 평균값. 1보다 작은 오차도 놓치지 않는 꼼꼼함을 갖고 있지만, 오차 크기가 아니라 부호에만 의존하기 때문에 작은 오차라도 기울기가 커질 수 있으므로 학습이 불안정합니다.               |
| 평균 제곱근 오차<br>(Root Mean Squared Error, RMSE) | 회귀            | MSE의 제곱근. 큰 오차에 대한 민감도를 줄여줍니다.                                                                                                 |

# 경사 하강법과 오차 역전파

- 경사하강법 : 손실을 가중치에 대해 미분한 다음, 기울기의 반대 방향으로 학습률 만큼 이동시키는 알고리즘
- 오차 역전파 : 정답과 예측 결과의 오차를 최소화하는 가중치를 찾는 알고리즘, 미분 연쇄 법칙을 이용하여, 출력층에 가까운 가중치부터 수정해 나간다.



▽ 미분의 연쇄 법칙을 이용하여 오차를 역전파!



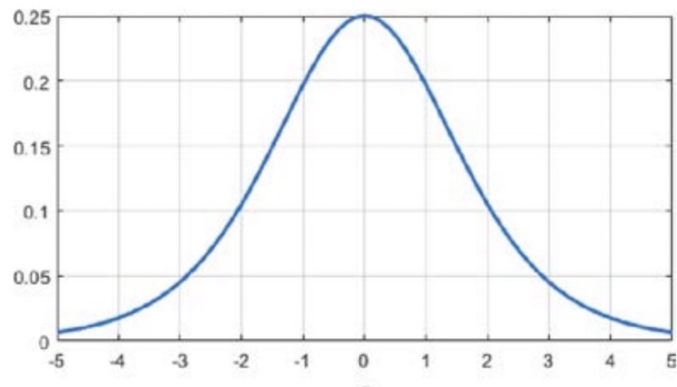
수식으로 표현하면  $W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$  입니다.

새로운 가중치( $W(t+1)$ )는 현재 가중치( $W$ )에서 기울기( $\frac{\partial \text{오차}}{\partial W}$ )를 뺀 값입니다.

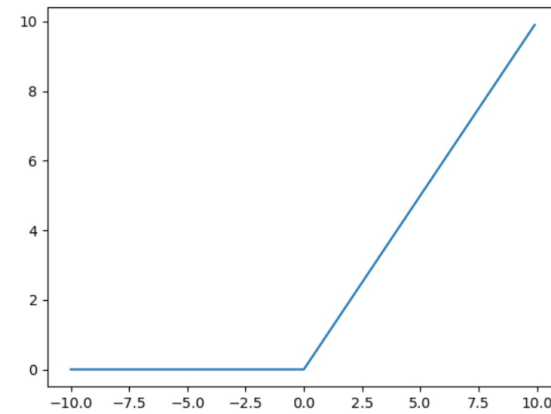
# 기울기 소실 예방

- 기울기 소실은 출력층으로 부터 멀어질수록 역전파되는 오차가 0에 가까워 지는 현상
- 오차가 역전파될 때 층을 한 번 거칠 때 마다, 활성화함수(시그모이드)의 미분(도함수)이 곱해지기 때문
- 시그모이드의 도함수는 크거나 작은 값에서는 0에 수렴
- ReLU 함수는 0보다 작은 값은 0으로, 큰 값은 그 값 그대로 내보냄

▼ 시그모이드 함수의 도함수



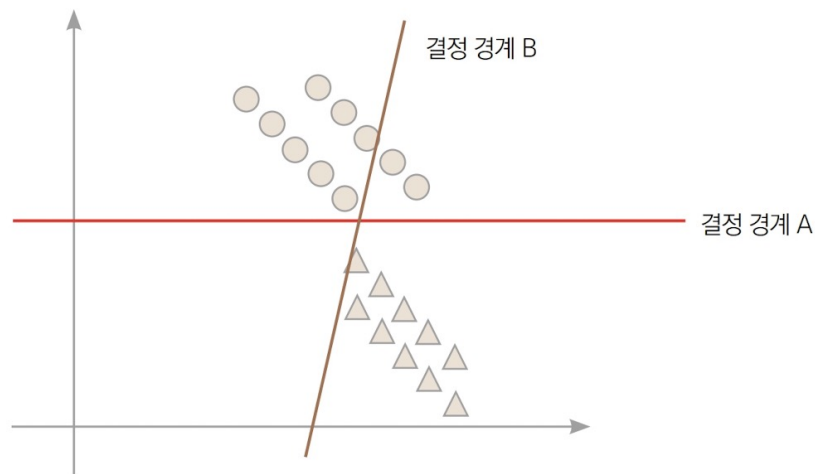
▼ ReLU 함수



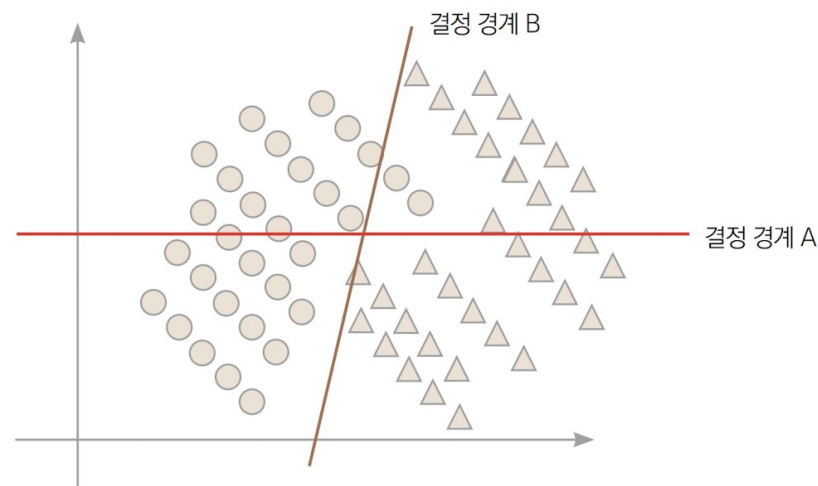
# 신경망 성능 비교

- 오버피팅은 과적합이라고 하며, 학습에 사용한 데이터에 최적화되어 다른 데이터에 대한 예측 성능이 떨어지는 것
- 성능이 너무 좋다면, 데이터가 부족한 게 아닌지 의심해 보자

▼ 같은 데이터를 다르게 분류하는 두 결정 경계



▼ 오버피팅된 결정 경계



# 정리 퀴즈

- 인공지능, 머신러닝, 딥러닝 중 가장 범위가 넓은 개념은?
  - [REDACTED]
- 매출전표를 보고 고객의 취향을 분석하는 머신러닝 알고리즘은 어떤 방법을 사용할까?
  - [REDACTED]
- 경사 하강법이 미분한 결과를, 기존의 값에서 더할 까 뺄까?
  - [REDACTED]
- 학습용 데이터를 완벽하게 분류하는 모델이 가장 좋을까?
  - [REDACTED]
- 이진 분류에서 왜 시그모이드를 사용 할까?
  - [REDACTED]

Q&A