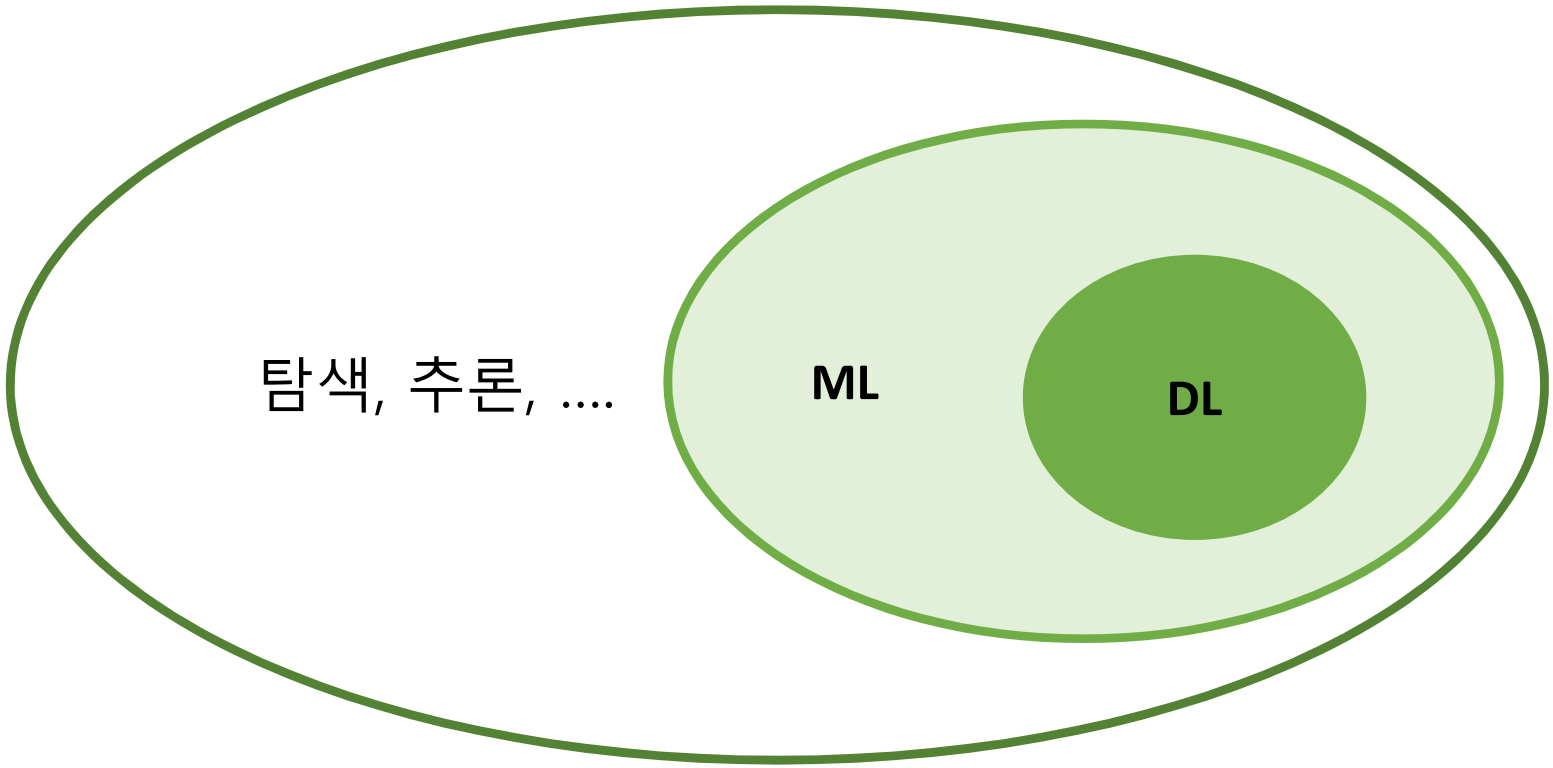
인공신경망
(ARTIFICAL NEURAL NETWORK)

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

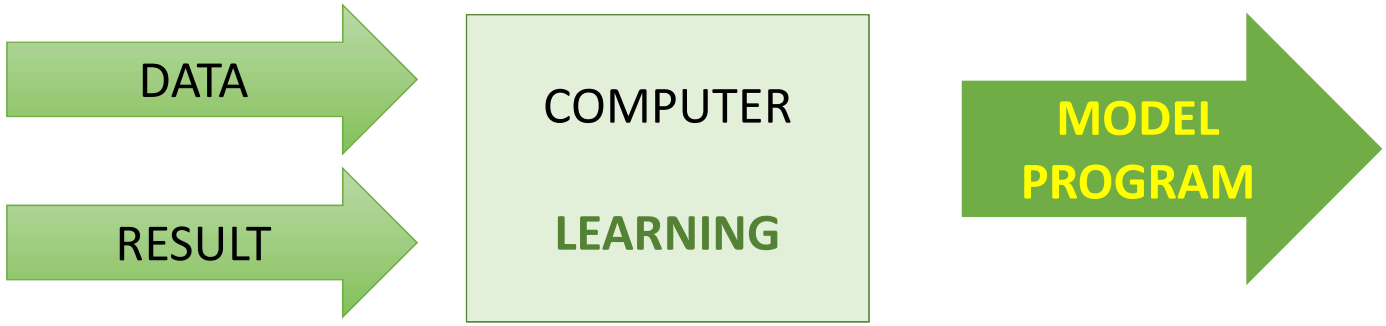


ARTIFICAL NEURAL NETWORK

◆ 인공신경망

- **Sequencial Computer**

- **Machine Learning** ➔ 많은데이터 + 고성능 Computer ➔ DL



ARTIFICAL NEURAL NETWORK

◆ 인공신경망

- **인간의 신경 세포(neuron) 구조**에 많은 영향을 받은 알고리즘
- 우리 몸의 각 신경 세포는 시냅스(synapse)를 통해 결합
- 자극이 들어왔을 때 가지 돌기와 축삭 돌기를 거쳐 시냅스를 통해 서로 전기적인 신호를 사용하여 통신
- 들어오는 자극이 일정하더라도 **각 신경 세포마다 자극에 반응하는 정도가 조금씩 달라, 자극을 강화시켜 전달하기도 하고 축소시켜 전달하기도 함**

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

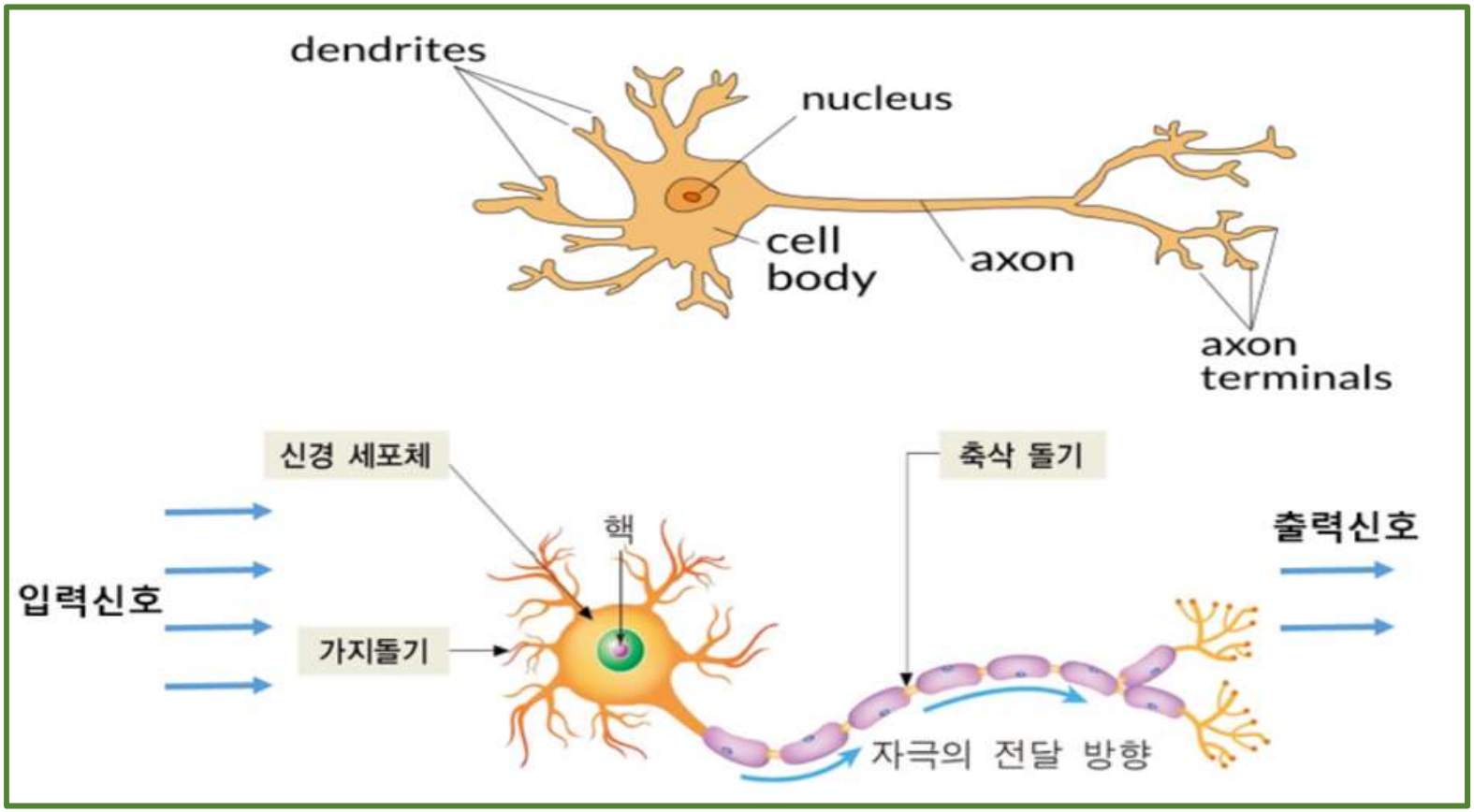
➢ 퍼셉트론(Perceptron)

- 1945년 개념 제안, 1958년 구체적인 공학적 구현 제안
- **생물학적 뉴런**을 **공학적인 구조**로 **변형**한 것
- 1969년 **XOR문제 해결할 수 없는 단순선형분리기** 불과함 증명
  인기하락 -> 논리적인 추론으로 인공지능 트랜드 변화
- 최초 인공신경망 개념을 공학적 구조로 구현한 것으로 큰 의미

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

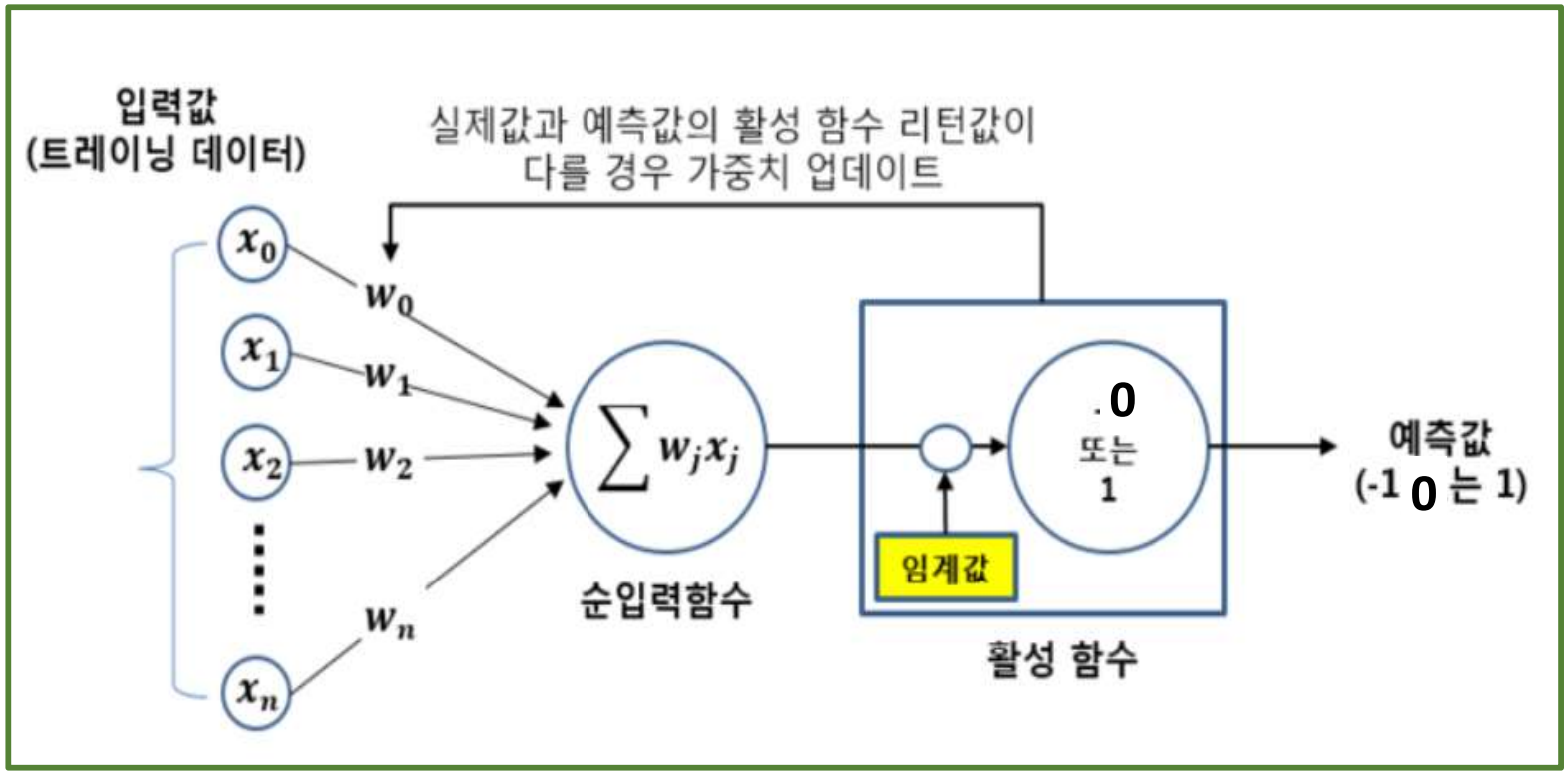
➢ 퍼셉트론(Perceptron)



ARTIFICAL NEURAL NETWORK

◆ 인공신경망

➢ 퍼셉트론(Perceptron)



y=선형이진분류기 (Linear Binary Classifier)

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

➢ 퍼셉트론(Perceptron)

[동작방식]

- Feed-Forward (전방향 연산)
  - 각 입력값에 일정한 가중치(weight) 곱셈
  - 출력층까지 전달 후 에러(손실) 계산
- Back-Propagate (역전파)
  - 실제 목표 출력치와 비교
  - 다음 입력 때 **출력치가 목표치에 근접**하도록 **가중치 조절 과정 반복**
- Feed-Forward & Back-Propagate + 많은 데이터 + 반복 수행

➔ 모든 학습 데이터가 **정확하게 분류될 때까지 반복 진행**

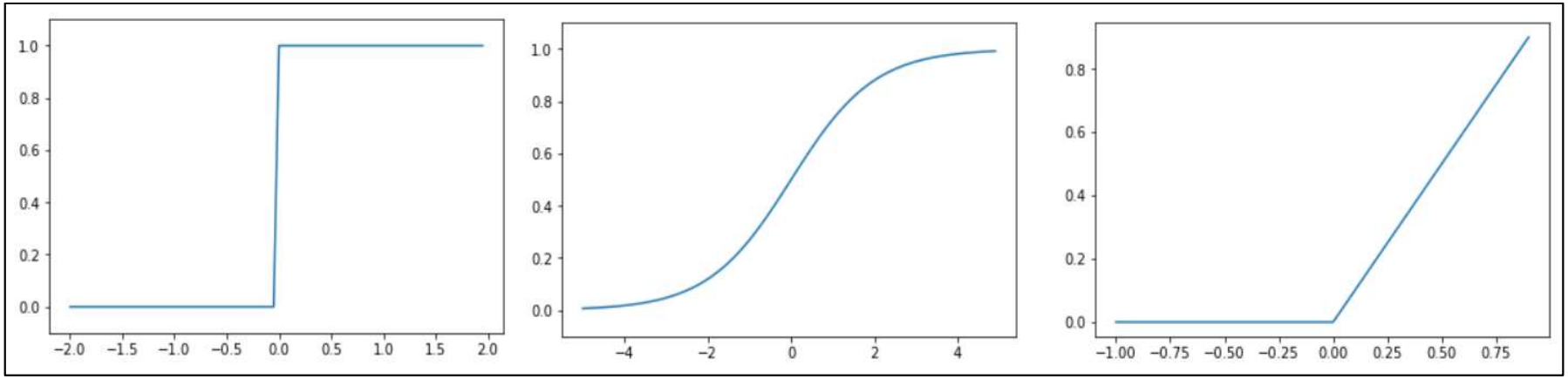
➔ **선형적 분리**에 적합

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

➢ 퍼셉트론(Perceptron)

함수 종류	기능 및 특징
Step Function	- 퍼셉트론에서 사용 즉 이진분류에 사용 - 값이 0보다 작으면 0 출력 / 크면 1출력
Sigmoid Function	- 0 ~ 1 사이의 값 출력 , 평균 0.5 - 분류에서 확률 표현 위해서 사용
ReLU Function (Rectified Linear Unit)	- 입력이 0을 넘으면 그 입력을 그대로 출력 - 입력이 0이하면 0을 출력



Step Function | Sigmoid Function | ReLU Function

ARTIFICAL NEURAL NETWORK

◆ 인공신경망

➢ 퍼셉트론(Perceptron)

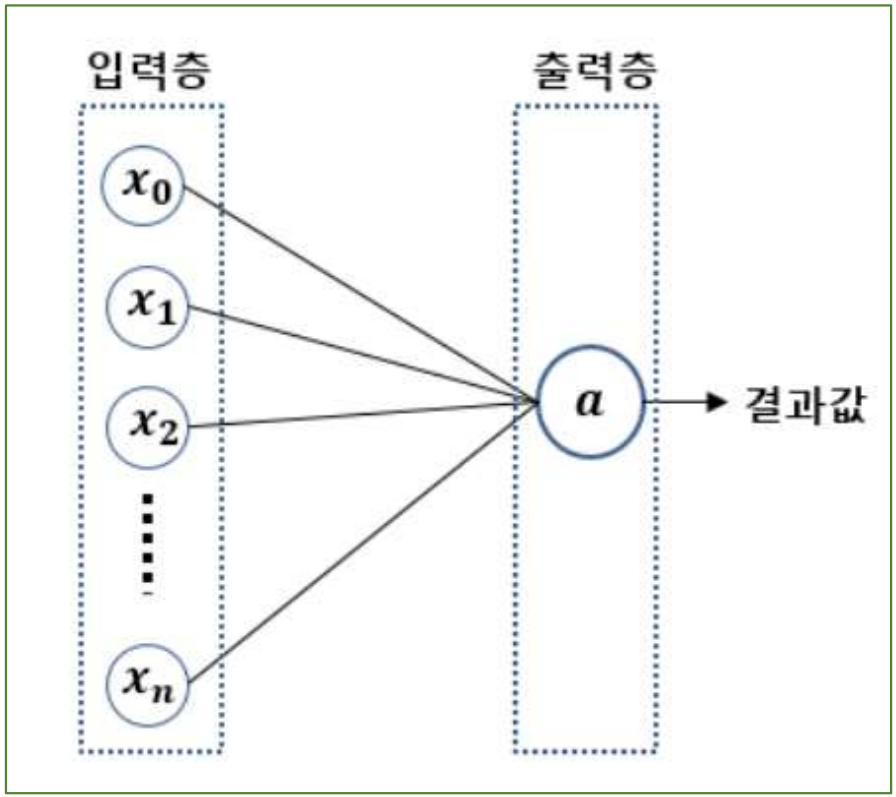
- Vanishing Gradient Problem (기울기 소실 문제)

함수 종류	기능 및 특징
Step Function	- -2이하 +2이상 시 미분값 0 되는 문제
Sigmoid Function	- -6이하 +6이상 시 미분값 0 되는 문제
ReLU Function (Rectified Linear Unit)	- Setp Function, Sigmoid Function 처럼 발생함, 정도가 약함

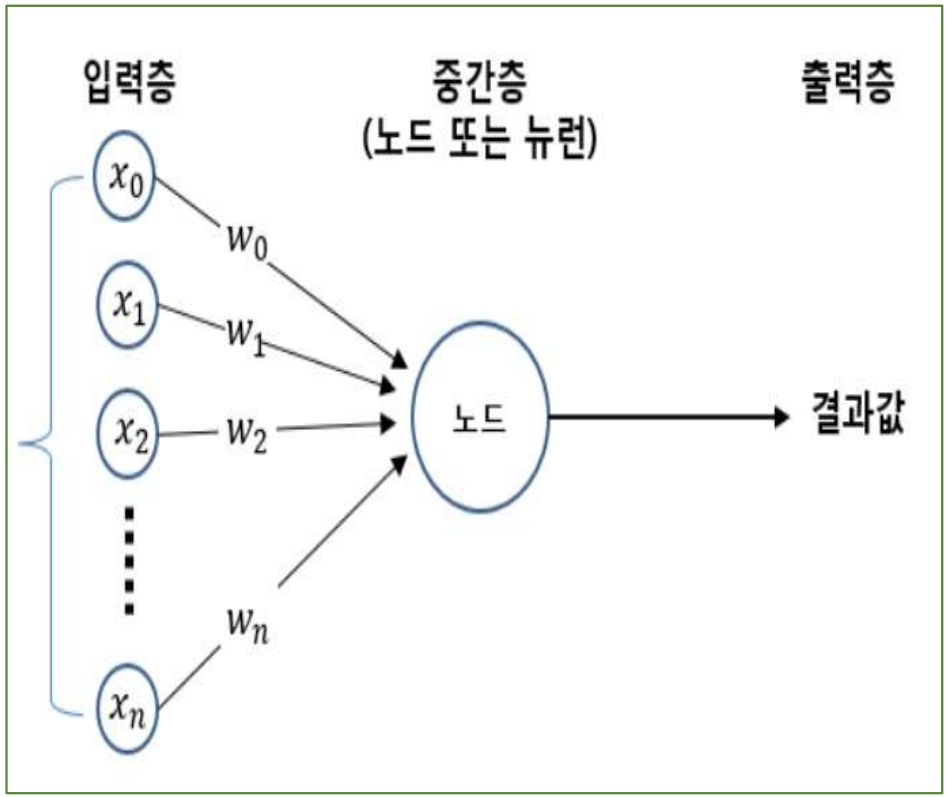
ARTIFICAL NEURAL NETWORK

◆ 인공신경망

➢ 퍼셉트론(Perceptron)



PLA
(Perceptron Learing Algorithm)

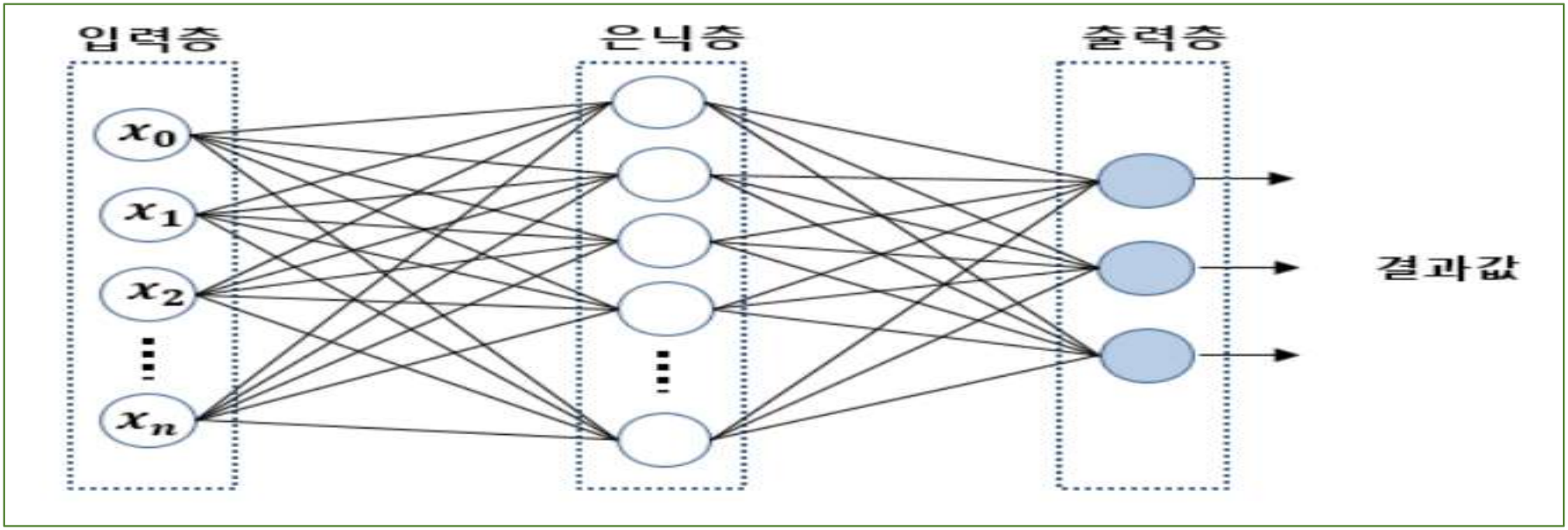


MLP
(Multi-Layer Perceptron Learing Algorithm)

ARTIFICAL NEURAL NETWORK

◆ 다층퍼셉트론

- 퍼셉트론을 **여러층 쌓아 올린 구조**
- 선형분리 불가능한 문제도 해결 가능
- **인공신경망(ANN)을 의미**함
- 경사하강법 손실 함수 사용



ARTIFICAL NEURAL NETWORK

◆ 다층퍼셉트론

➢ 오류역전파(Back Propagate)

- 1986년 제안된 효율적 **최적화 알고리즘**
- 경사하강법 이용 파라미터(w, b) 업데이트하며 학습 진행
- **각 Layer에서 손실함수 미분값 계산에 어려움**을 해결한 알고리즘
- 미분학의 '체인툴 ' 착안 ➔ 연쇄
- **출력Layer에서 입력Layer로 오류를 전달하며 파라미터(w, b) 업데이트**

ARTIFICAL NEURAL NETWORK

◆ 다층퍼셉트론

➢ 최적화

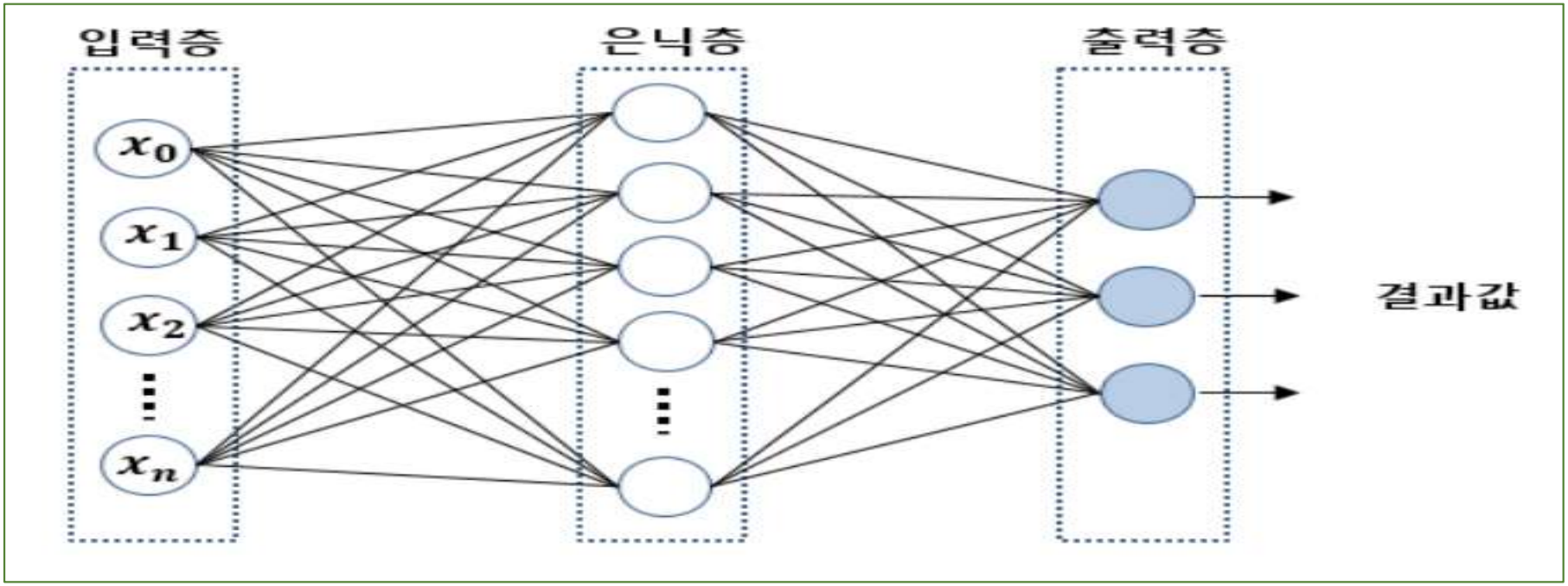
- 손실함수의 값을 최소하하기 위해 기울기를 이용하는 **경사하강법** 사용
- step-size 간격으로 수행, 보통 0.1~0.001 속도가 적당

Batch Gradient Descent	전체 학습 데이터 Loss Function에서 사용 많은 시간 및 계산량 소요
SGD (Stochastic Gradien Descent)	일부 학습 데이터(mini-batch)를 Loss Function에서 사용 BGD에 비해 다소 부정확할 수 있지만 속도가 훨씬 빠름
Momentum	경사하강법에서 업데이트 시 이전 값과 비교하여 같은 방향으로 업데이트 진행 -> 경사하강 + 관성
AdaGrad (Adaptive Gradient)	변수들을 update할 때 각각의 변수마다 step size를 다르게 설정해서 이동하는 방식, 빈도에 따라 설정, 단어구별에 사용
RMSProp / AdaDelta	AdaGrad의 단점을 해결한 방법
Adam (Adaptive Moment Estimation)	과거 미분값 계속 가중평균 내면서 효율적 업데이트 RMSProp + Momentum 방식 결합

DNN(DEEP Neural Network)

◆DNN (심층신경망)

- **Hidden Layer 층**이 2개 이상인 ANN
- 층이 많아 더 강한 학습 · 정확한 답 추론 가능
- **시간이 오래 소요**됨, **과도학 학습으로 문제점 발생**



Neural Network LIB

SCIKIT-LEARN

◆ Perceptron

- 아주 간단한 신경망 분류 모델
- 선형 분류

```
SGDClassifier( loss="perceptron",
               eta0=1,
               learning_rate="constant“
               penalty=None).
```

SCIKIT-LEARN

Perceptron

```
class sklearn.linear_model.Perceptron(
        penalty=None
        alpha=0.0001
        l1_ratio=0.15
        fit_intercept=True
        max_iter=1000
        tol=0.001
        shuffle=True
        verbose=0
        eta0=1.0,
```

SCIKIT-LEARN

Perceptron

```
class sklearn.linear_model.Perceptron(
        n_jobs=None
        random_state=0
        early_stopping=False
        validation_fraction=0.1
        n_iter_no_change=5
        class_weight=None
        warm_start=False)
```

SCIKIT-LEARN

◆ MLP

- 아구 간단한 신경망 분류 모델
- 선형 분류
- SGDClassifier(loss="perceptron", eta0=1, learning_rate="constant", penalty=None).

- SGDClassifier(loss="perceptron",
  eta0=1,
  learning_rate="constant“
  penalty=None).

SCIKIT-LEARN

◆ MLPClassifier

```
class sklearn.neural_network.MLPClassifier(
        hidden_layer_sizes=(100,)  # ( 은닉층수, 뉴런수 )
        activation='relu'          # 은닉층의 활성화 함수
        solver='adam'              # 경사 하강법 알고리즘
        alpha=0.0001               # L2규제 강도설정(클수록 강)
        batch_size='auto'          # 배치 사이즈
        learning_rate= ' constant' # 학습률
        learning_rate_init=0.001   #학습률 초기값
        power_t=0.5                # solver='sgd'인경우 학습률 계산
        max_iter=200               # 학습 횟수
        shuffle=True               # 섞기여부
```

SCIKIT-LEARN

◆ MLPClassifier

```
class sklearn.neural_network.MLPClassifier(
        random_state=None
        tol=0.0001
        verbose=False
        warm_start=False
        momentum=0.9                    # solver='sgd'인경우 적용
        nesterovs_momentum=True
        early_stopping=False            # solver='sgd' 또는 'adam'인 경우
        validation_fraction=0.1
        beta_1=0.9
        beta_2=0.999
```

SCIKIT-LEARN

◆ MLPClassifier

```
class sklearn.neural_network.MLPClassifier(
        epsilon=1e-08
        n_iter_no_change=10      # solver='sgd' 또는 'adam'인 경우
        max_fun=15000)           # 손실 함수 호출의 최대 수
```

SCIKIT-LEARN

MLPRegressor

```
class sklearn.neural_network.MLPRegressor(
        hidden_layer_sizes=(100,)
        activation='relu'
        solver='adam'
        alpha=0.0001
        batch_size='auto'
        learning_rate='constant'
        learning_rate_init=0.001
        power_t=0.5
        max_iter=200
        shuffle=True
```

SCIKIT-LEARN

MLPRegressor

```
class sklearn.neural_network.MLPRegressor(
        random_state=None
        tol=0.0001
        verbose=False
        warm_start=False
        momentum=0.9
        nesterovs_momentum=True
        early_stopping=False
        validation_fraction=0.1
        beta_1=0.9
        beta_2=0.999
```

SCIKIT-LEARN

MLPRegressor

```
class sklearn.neural_network.MLPRegressor(
        epsilon=1e-08
        n_iter_no_change=10
        max_fun=15000)
```

KERAS

◆Keras란

- Theano 또는 TensorFlow를 계산 엔진으로 사용하는 파이썬 패키지
- 신경망을 구성하기 위한 각 구성요소를 클래스로 제공
- 쉽게 신경망 구현

➢ 버전 확인

```
(EV_PY375) C:\Users\anece\PycharmProjects\WEEK03\ML_TENSOR>python
Python 3.7.5 (default, Oct 31 2019, 15:18:51) [MSC v.1916 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>> keras.__version__
'2.2.2'
>>>
```

KERAS

◆Keras기반 신경망 클래스

Sequential 클래스	Layer를 선형으로 쌓는 모델 객체
.add	레이어 추가 - 입력단부터 순차적으로 추가 - 출력 뉴런 개수를 첫번째 인수로 지정 - 최초의 레이어는 input_dim 인수로 입력 크기 설정 - activation인수로 활성화함수 설정
.compile	모델(모형) 완성 - loss 인수 : 비용함수 설정 - optimizer: 최적화 알고리즘 설정 - metrics : 트레이닝 단계에서 기록할 성능 기준 설정
.fit	트레이닝 - nb_epoch : 에포크 횟수 설정 - batch_size : 배치 크기 설정 - verbose : 학습 중 출력 문구 설정
.evaluate	모델 평가 - x_test, y_test 인자 / test_loss, test_acc 리턴
.summary	모델 내부구조 정보 출력

KERAS

◆Keras기반 신경망 클래스

Sequential 클래스	Layer를 선형으로 쌓는 모델 객체
.save(‘model_name.h5’)	인공신경망 hdf5 파일로 저장
load_model(‘model_name.h5’)	인공신경망 모델 hdf5 파일 로딩

KERAS

◆Keras기반 신경망 클래스

Danse 클래스	Layer 객체
Dense()	- 첫번째 인자 : 출력 뉴런 수 설정 - input_dim : 입력 뉴런 수 설정 - init : 가중치 초기화 방법 설정 uniform' : 균일 분포 'normal' : 가우시안 분포 - activation : 활성화 함수 설정 'linear' : 디폴트 값, 입력뉴런과 가중치로 계산된 결과값이 그대로 출력 'relu' : rectifier 함수, 은익층에 주로 사용 'sigmoid' : 시그모이드 함수, 이진 분류에서 출력층 주로 사용 'softmax' : 소프트맥스 함수, 다중 클래스 분류에서 출력층에 주로 사용

Dropout 클래스	기능
Dropout()	- 과적합/과소적합을 해결하기 위한 방법으로 에포크 바다 임의의 은닉계층 뉴런의 일부%(보통 정발)를 최적화에 포함시키지 않음 - 첫번째 인자 : 0 ~ 1사이 비율