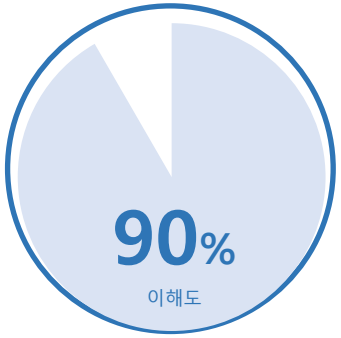
지능형 컴퓨팅과정 포트폴리오 경진대회

학과	학번	이름
컴퓨터정보공학과	20191797	박별이

목차

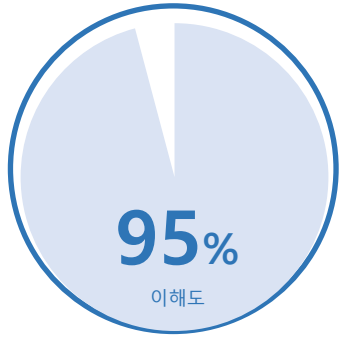
1. 개요 및 데이터 저장과 전처리

- 머신러닝 분류 개요
- 행렬 곱셈
- 브로드캐스팅
- 회귀와 분류
- 정규화 등의 전처리



2. 딥러닝 모델

- 입력, 중간(은닉), 출력 층, 패러미터 수
- 모델 종류(ANN, CNN, RNN 등)
- 옵티마이저, 손실함수
- Early Stopping (Callback 함수)



1강 머신러닝 분류 개요

- **지도학습** -> 정답O
  => 올바른 입력과 출력의 쌍으로 구성된 정답의
  훈련 데이터(labeled data)로부터 입출력 간의 함수를 학습시키는 방법

- **비지도(자율)학습**(unsupervised learning) -> 정답X
  => 정답이 없는 훈련 데이터(unlabeled data)를 사용하여
  데이터 내에 숨어있는 어떤 관계를 찾아내는 방법

- **강화학습**(reinforcement learning) –> 보상, 벌O
  => 잘한 행동에 대해 보상을 주고 잘못한 행동에 대해 벌을
  주는 경험을 통해 지식을 학 습하는 방법

행렬 곱셈

- **행렬 곱셈** -> 2X3 3X2에서 3으로 똑같아야 함
  => 앞에 있는 열과 뒤에 있는 행의 값 개수가 같아야 함

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

```
[4]  # 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())
```

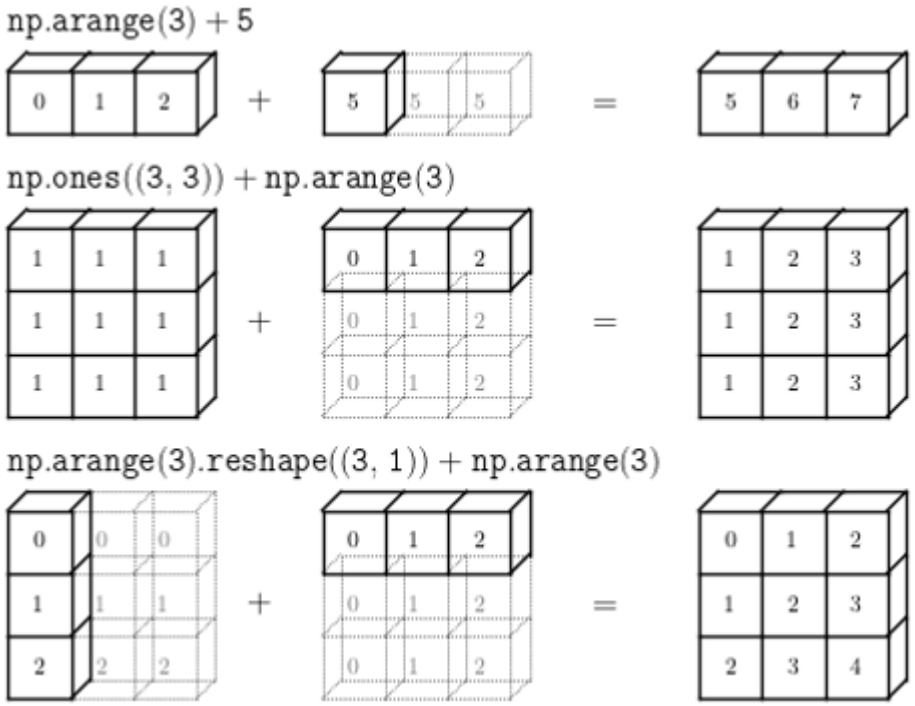
```
[[ 4.  4.]
 [10.  8.]]
```

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

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

브로드캐스팅

- **브로드캐스팅** -> Shape이 다르더라도 연산이 가능하도록
  => 가지고 있는 값을 이용하여 Shape을 맞춤



[0 1 2]

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

     x = tf.constant((np.ones((3, 3))))
     y = tf.constant(np.arange(3), dtype=tf.double)
     print((x+y).numpy())

     x = tf.constant(np.arange(3).reshape(3, 1))
     y = tf.constant(np.arange(3))
     print((x+y).numpy())
```

```
[5 6 7]
[[1. 2. 3.]
 [1. 2. 3.]
 [1. 2. 3.]]
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

3행, 3열로 값이 모두 1.인 배열 생성

회귀와 분류

- **회귀 모델 : 연속적인 값을 예측**

  1. 단순 선형 회귀 분석 -> 입력 : 특징 1개 | 출력 : 하나의 값

  2. 다중 선형 회귀 분석 -> 입력 : 특징 n개 | 출력 : 하나의 값
  ⇒선형 회귀 : 데이터의 경향성을 가장 잘 설명하는 하나의 직선을 예측하는 방법
  ex) 키와 몸무게, 치킨과 맥주의 판매량
  ⇒딥러닝 분야에서 선형회귀는 Y = wX + b 즉, **가중치 w와 편향인 b를 구하는 것**

  3. 로지스틱 회귀(이진 분류) -> 입력 : 1 or n개 | 출력 : 0 or 1

- **분류 모델 : 불연속적인 값(종류)을 예측**

정규화 등의 전처리

- 전처리 : 기존 데이터를 머신러닝 알고리즘에 알맞은 데이터로 바꾸는 과정

⇒ 전처리 과정을 통해 모델 학습의 성능을 높일 수 O

- **정규화**

1. Normalization : 값의 범위(scale)를 0~1 or 0~n사이의 실수로 값을 구성
   -> 값의 범위 줄임, 학습 속도 향상
   ex) 샘플 값을 정수(0~255)에서 부동소수(0~1)로 변환
   x_train, x_test = x_train / 255.0, x_test / 255.0

2. Standardization : 값의 범위(scale)를 평균 0, 분산 1이 되도록 변환
   -> 값의 범위 줄임, 학습 속도 향상

3. Regularization : weight를 조정하는데 규제(제약)를 거는 기법
   -> Overfitting을 막기위해 사용

2강 퍼셉트론, 패러미터 수

- Flatten 평평하게 1줄로 만듦
- Dense 히든층 만듦 -> 뉴런 수(임의 값) | 출력 때 -> 클래스 수
  사이에 가중치랑 편향은 컴퓨터가 알아내야하는 최종 값
  => 패러미터(가중치+편향)
  패러미터 수 구하는 법 = (입력측 뉴런 수 + 1) * (출력측 뉴런 수)

① 훈련과 정답 데이터 지정 ① - 1 데이터 전처리(옵션)

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist
(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
```

2강 퍼셉트론, 패러미터 수

② 모델 구성

```
# 층을 차례대로 쌓아 tf.keras.models.Sequential 모델을 생성
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
-> 60000 개의 (28, 28) 크기를 가진 배열

    tf.keras.layers.Dense(128, activation='relu'),
-> Dense() 완전 연결층, activation 활성화 함수

    tf.keras.layers.Dropout(0.2),
-> 훈련 중에 20%를 중간에 끊음, 예측때는 모두 사용

    tf.keras.layers.Dense(10, activation='softmax')
-> 출력이 10개, 확률값이 가장 큰 것이 결과 ])
```

2강 퍼셉트론, 패러미터 수

③ 학습에 필요한 최적화 방법과 손실 함수 등 설정
③ - 1 구성된 모델 요약(옵션)
# 모델 요약 표시
model.summary()
-> Total params: 101,770 -> 모델이 구해야 할 수의 개수

훈련에 사용할 옵티마이저(optimizer)와 손실 함수, 출력정보를 모델에 설정
model.compile(optimizer='adam',
-> 옵티마이저 (입력된 데이터와 손실 함수를 기반으로
모델(w와 b)을 업데이트하는 메커니즘)
loss='sparse_categorical_crossentropy',
-> 손실 함수 (• 훈련 데이터에서 신경망의 성능을 측정하는
방법 • 모델이 옳은 방향으로 학습될 수 있도록 도와 주는 기준 값)
metrics=['accuracy'])
-> 훈련과 테스트 과정을 모니터링할 지표 (여기에서는
정확도(정확히 분류된 이미지의 비율)만 고려)
metrics=['accuracy', 'mse'])

2강 퍼셉트론, 패러미터 수

④ 생성된 모델로 훈련 데이터 학습

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

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

2강 퍼셉트론, 패러미터 수

시그모이드 함수 (S자 곡선) -> 0, 1 사이의 값

```
def sigm_func(x):
    return 1 / (1 + np.exp(-x))
```

ReLU 함수

0, 음수면 0이고 양수면 x값

```
def relu_func(x):
    return np.maximum(0, x)
```

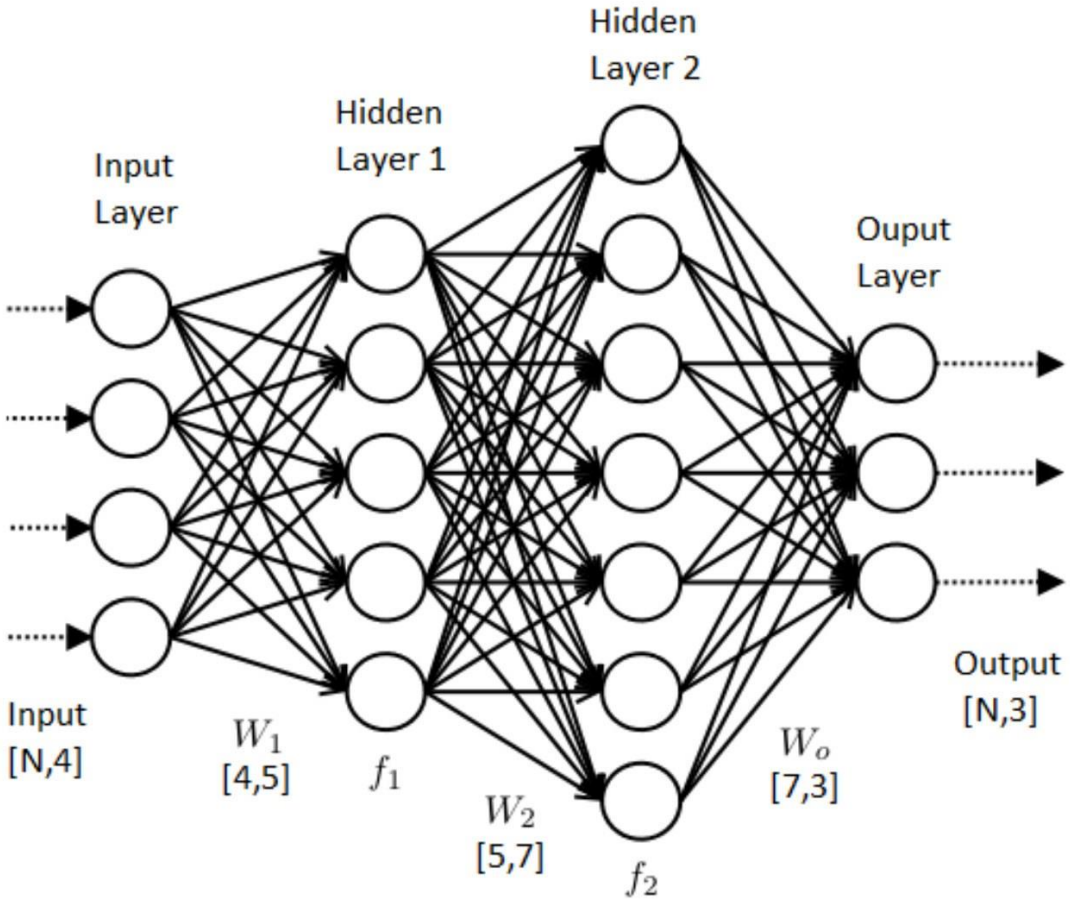
OR 게이트 구현 -> 1이 있으면 1

XOR 게이트 구현 -> XOR은 같으면 0, 다르면 1

모델 종류(ANN, CNN, RNN)

- **ANN (Artificial Neural Network)**

⇒ 사람의 신경망 원리와 구조를 모방하여 만든 기계학습 알고리즘



다수의 입력 데이터를 받는 **입력층**(Input)
데이터의 출력을 담당하는 **출력층**(Output)
입력층과 출력층 사이에 존재하는 레이어들
(**중간층, 은닉층**)

⇒ 은닉층의 개수와 노드의 개수를 구성해 원하는 Output값을 예측을 해야 함

⇒ 은닉층에서 **활성화 함수**를 사용해 최적의 **Weight와 Bias**를 찾아냄

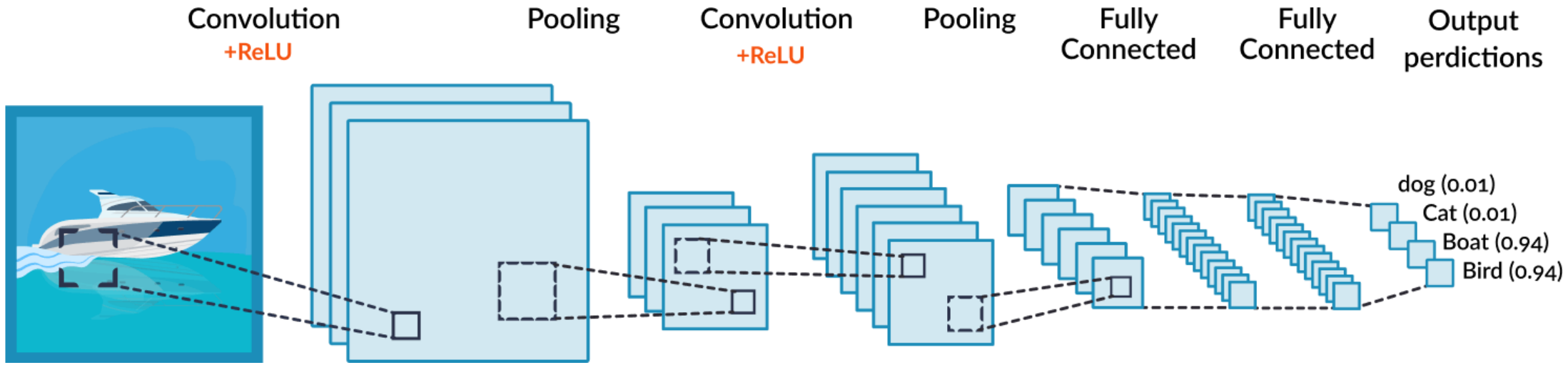
모델 종류(ANN, CNN, RNN)

- **CNN** (합성곱신경망 : Convolution Neural Network)

⇒데이터의 특징을 추출하여 특징들의 패턴을 파악하는 구조

1. Convolution : 데이터의 특징을 추출하는 과정 (조사->특징 파악->압축)
2. Pooling : 위 과정 후 레이어의 사이즈 줄임, 노이즈 상쇄, 특징 제공

ex) 정보추출, 문장분류, 얼굴인식 등



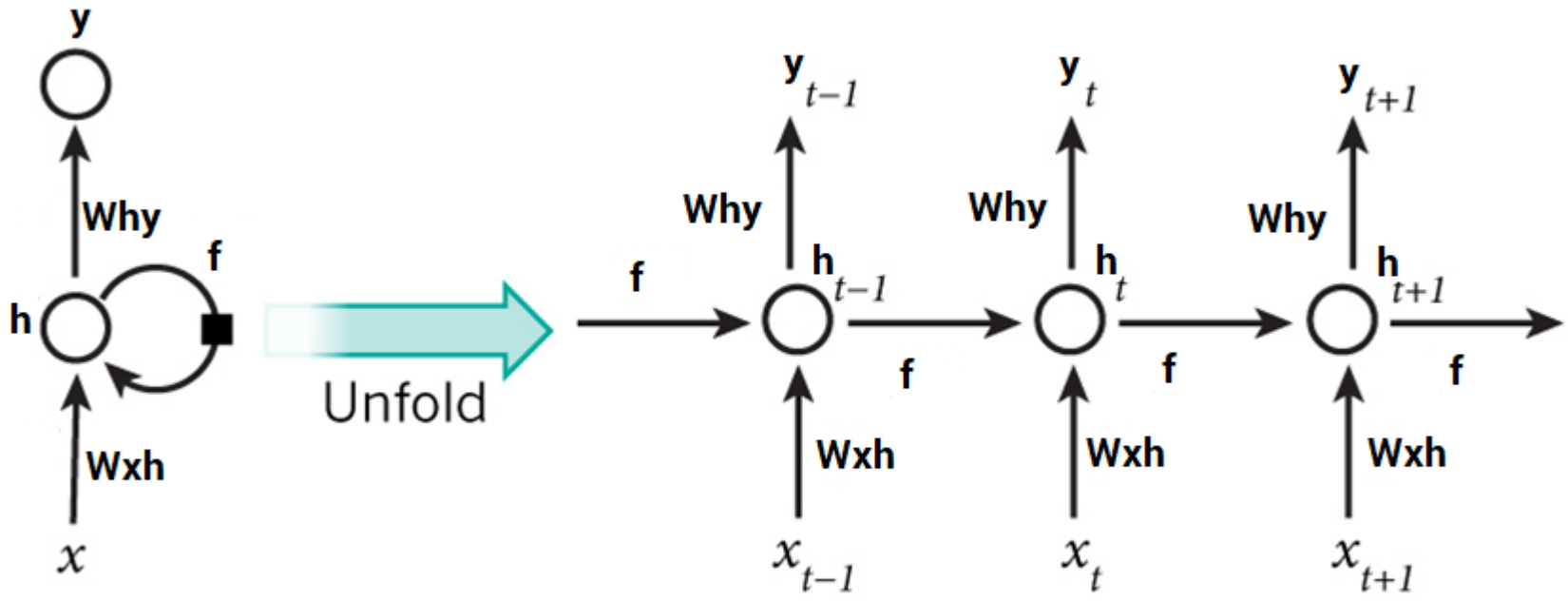
모델 종류(ANN, CNN, RNN)

- **RNN** (순환신경망 : Recurrent Neural Network)

⇒ 반복적이고 순차적인 데이터 학습에 특화된 인공신경망의 한 종류

순환구조를 이용하여 과거의 학습을 가중치를 통해 현재 학습에 반영

ex) 음성 웨이브폼 파악, 텍스트의 앞 뒤 성분 파악



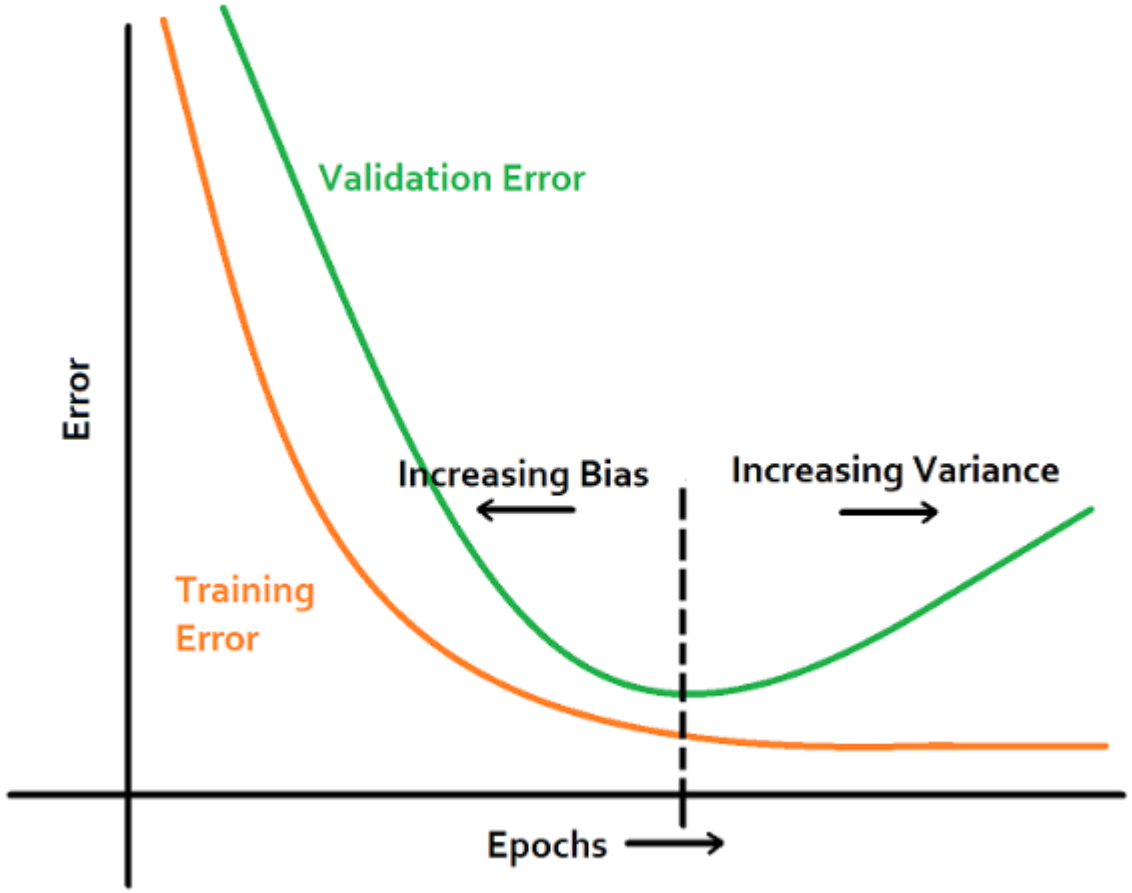
옵티마이저, 손실함수

- **옵티마이저**(Optimizer) : 최적화 과정 -> 적절한 W와 b를 찾아내는 과정
  => Gradient Descent(**경사 하강법**)
  1. 시작 값(시작점)을 선택 (중요X)
  2. 시작점에서 손실 곡선의 기울기를 계산
  3. 기울기가 0인 지점을 찾기위해 기울기 반대 방향으로 이동
  4. 기울기에 학습률을 곱하여 다음 지점을 결정
  5. 학습률의 값이 너무 작으면 학습시간 길고 너무 크면 최저점 이탈
  => 적절한 학습률 설정해야 함 (기울기가 작으면 학습률을 크게)
  보통 0.001에서 0.1 사용

- **손실함수**(Loss Function) : 예측 값의 오차를 줄이는 일에 최적화 된 식
  =>MSE(Mean Square Error 평균제곱오차)

Early Stopping

- **Early stopping : 무조건 Epoch 을 많이 돌린 후, 특정 시점에서 멈추는 것**



인공지능응용프로그래밍

지능형 컴퓨팅과정 포트폴리오 경진대회

끝까지 읽어주셔서 감사합니다!