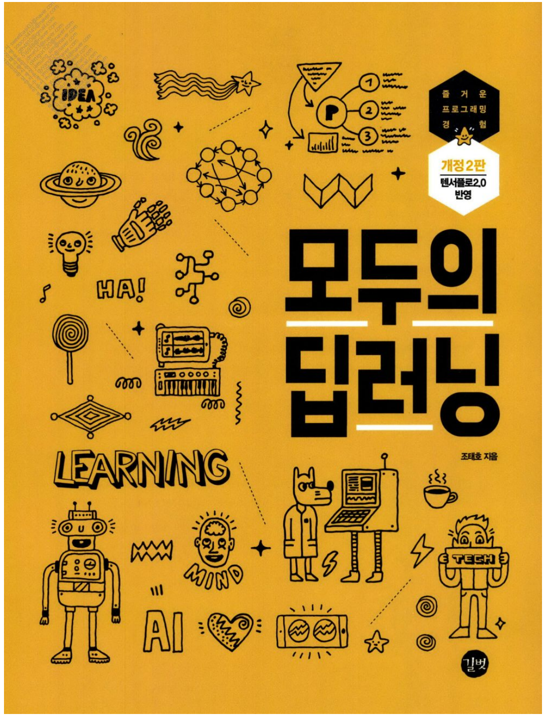
딥러닝의 동작 원리 _ 3장 선형 회귀

2023 인공지능 기초 스터디 1팀

CONTENT

책 소개 : 모두의 딥러닝

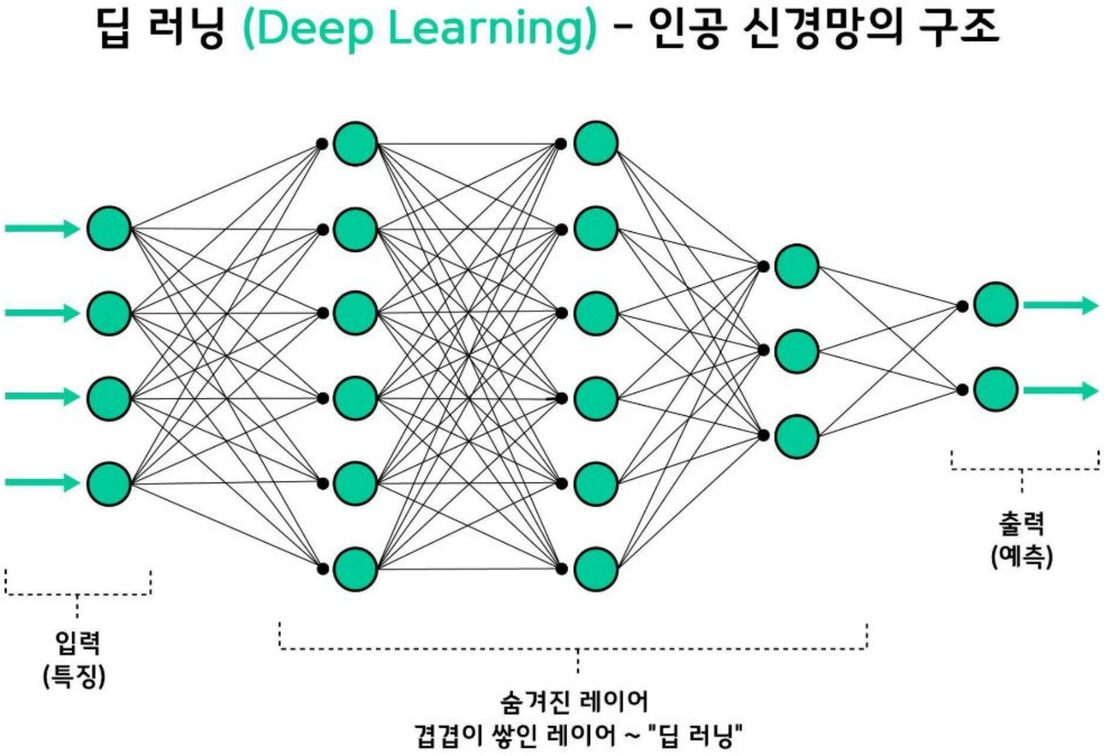


1. 2022 winter AI study 책
2. 파이썬 인공지능 관련 라이브러리 다뤄보기 (데이터 분석 스터디에서 학습한 내용 이용하기)
3. 선형회귀, 분류, 퍼셉트론, CNN, RNN, 자연어처리, GAN에 대한 기초 공부해보기

교재의 추가적인 복제 및 재배포를 막기 위해 pdf에 워터마크가 새겨져 있습니다.

발표 자료 제작 시 교재 캡쳐는 삼가주세요!

딥러닝에 대해 알아보기



1장 맛보기 : 머신러닝과 딥러닝

1장 맛보기 : 머신러닝과 딥러닝

Q1. 방대한 데이터를 수집할 수 있게 되면서 빅데이터를 분석해 스스로 학습하는 (B)의 형태로 진화하였다. 다만 데이터가 포함한 내용의 특징을 파악하는 데 한계를 보였는데, 이를 뛰어넘는 (C)이 고안되면서 문제가 해결되었다. *
(A)의 큰 범주 안에 (B)이 속하고, (B)의 일부분이 (C)인 것이다.

- ○ (A) 인공지능 , (B) 딥러닝 , (C) 머신러닝
- ◉ (A) 인공지능 , (B) 머신러닝 , (C) 딥러닝
- ○ (A) 머신러닝 , (B) 딥러닝 , (C) 인공지능
- ○ (A) 머신러닝 , (B) 인공지능 , (C) 딥러닝

1장 맛보기 : 머신러닝과 딥러닝

1장 맛보기 : 딥러닝 실행을 위한 준비사항

Data

-지도학습
-비지도학습

Computer

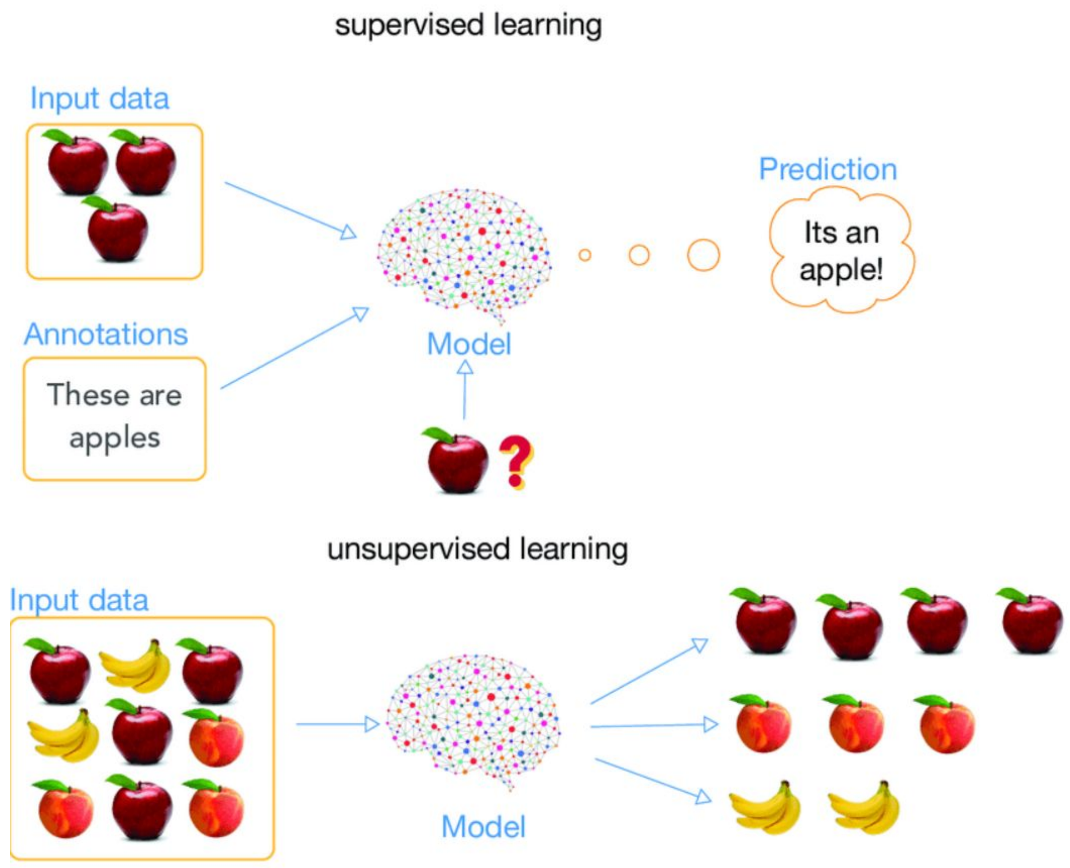
- 일반 CPU 컴퓨터
- GPU

Program

-딥러닝 구동

1장 맛보기 : 딥러닝 실행을 위한 준비사항(데이터)

1장 맛보기 : 딥러닝 실행을 위한 준비사항(데이터)

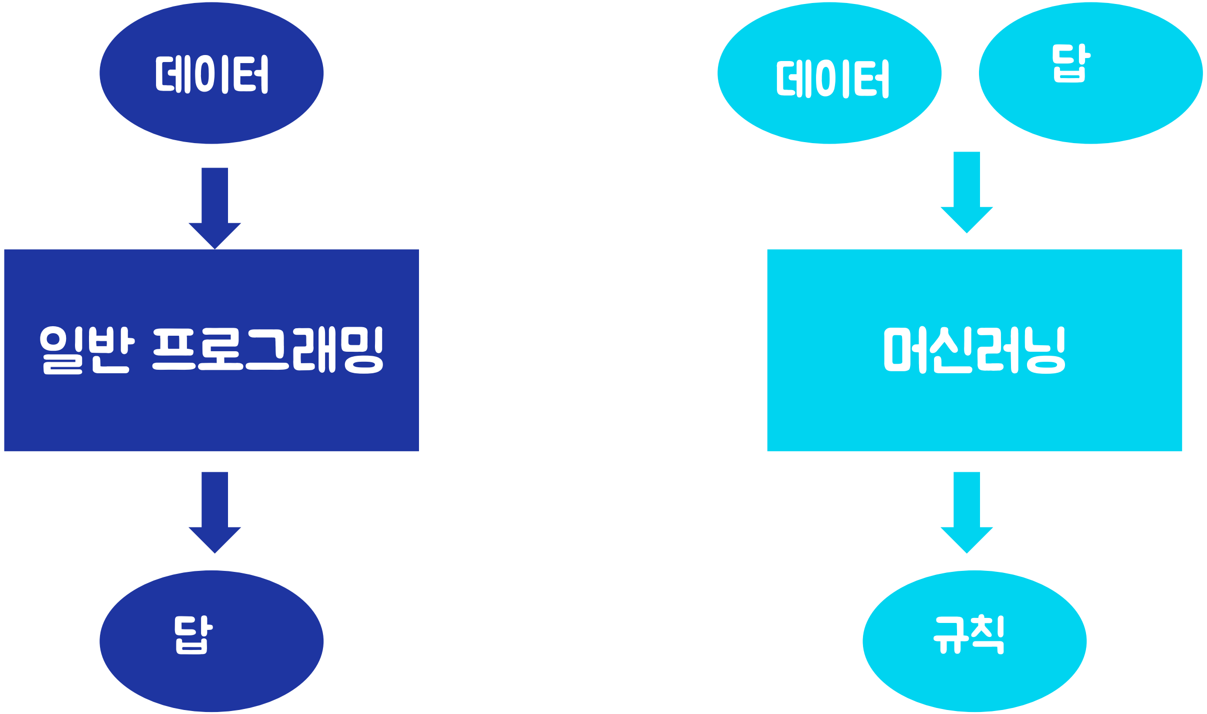


깜짝 Quiz!

이름표가 없는 머핀과 치와와 사진이 섞여 있습니다. 우리는 이 사진들을 두 개의 그룹으로 분류하고 싶은데요.
이때 우리의 선택은?!
(댓글로 답을 보내주세요~)

1. 지도 학습
2. 비지도 학습

1장 맛보기 : 미지의 일을 예측하는 일



Colab(Colaboratory, 코랩)

: 구글 서버에서 제공하는 주피터 노트북 개발환경

장점	단점
(1) 작성 용이 (2) 별도의 설치나 컴퓨터의 성능 불필요	(1) 새로 설치한 패키지 저장 X (2) 파일이나 가중치 값 유실 가능

NumPy

(Numerical Python)

: 배열 형태의 데이터를 처리, 연산하기 위한 파이썬 라이브러리

1 . 고성능의 수치 계산

2. 고성능의 수치 해석 가능

3. 통계, 선형대수, 행렬 계산

4. 벡터, 다차원 배열 연산

NumPy 문법
(Numerical Python)

생성	연산
난수 생성	저장 / 불러오기

https://colab.research.google.com/drive/1mmgdLhPKWE5IwV8H6-VHViTfwPT6FkKj#scrollTo=Lti5FVsCqjEc

[NumPy 문법] 생성

np.array()

```
# (1) 생성

x = np.array([1.0, 2.0, 3.0])
y = np.array([4.0, 5.0, 6.0])

print(x)
print(y)
```

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

[NumPy 문법] 연산

# 사칙연산 	
n1=np.add(x,y) # x+y
n2=np.subtract(x,y) # x-y
n3=np.multiply(x,y) # x*y
n4=np.divide(x,y) # x/y
	[5. 7. 9.]
	[-3. -3. -3.]
	[4. 10. 18.]
	[0.25 0.4 0.5]

# 행렬곱 	
n5 = np.dot(x,y)
	32.0

# 최대, 최소 	
max= np.max(x)
min=np.min(x)
	3.0
	1.0

[NumPy 문법] 연산

Mean : 산술평균
Average : 가중치를 둔 평균

```
[30] # (2) 연산 _ 평균
     n6=np.mean(x)
     n7=np.average(x)

     print(n6)
     print(n7)
```

```
2.0
2.0
```

차이

```
# (2) 연산 _ mean VS average
n6=np.mean(x)
n7=np.average(x,weights=(2,2,1))

print(n6)
print(n7) # (1*2+2*2+3*1)/3
```

```
2.0
1.8
```

[NumPy 문법] 난수 생성

np.random.rand : N개의 난수를 생성

```
a = np.random.rand(5)
print(a)
```

```
[0.1102777  0.75824413 0.02564787 0.85949244 0.26065274]
```

np.random.randint : A부터 B까지 중에서 정수인 난수를 생성

```
c = np.random.randint(1, 5, size=(2, 3))
print(c)
```

```
[[3 4 2]
 [2 3 2]]
```

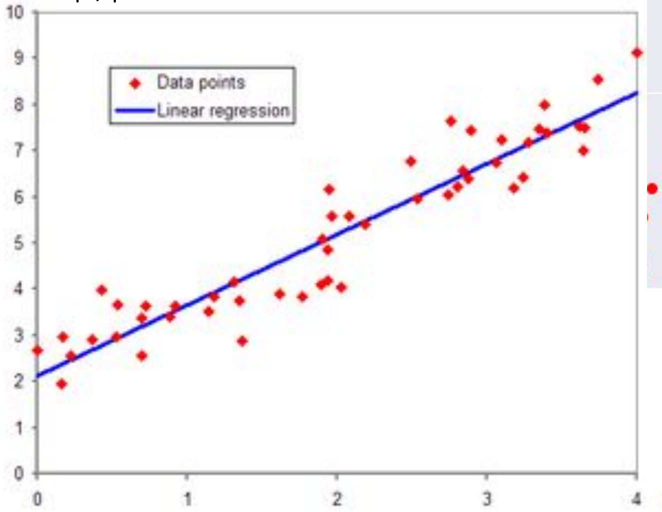
np.random.seed : 알고리즘을 거쳐 난수처럼 보이게 만드는 함수

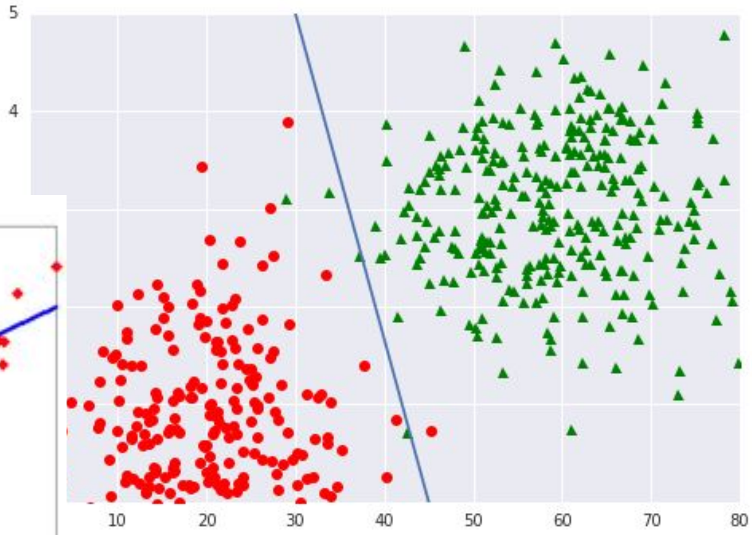
[NumPy 문법] 저장/불러오기

저장		불러오기
[1개의 배열]	np.save()	np.load ()
[2개 이상 배열]	np.savez()	

회귀_분류

<회귀>





<분류>

이진 분류

다중 클래스 분류

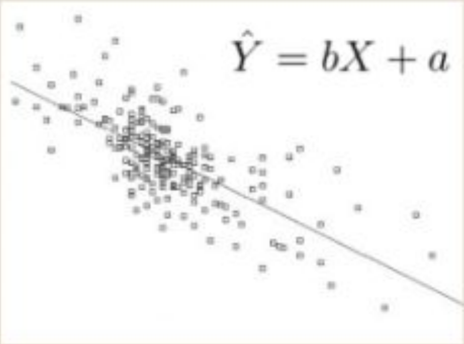
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

선형회귀

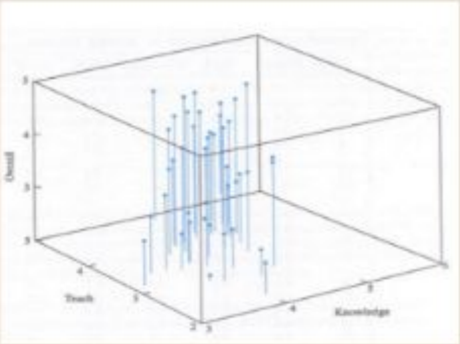
Linear Regression

선형회귀란?

입력
(정보) → 처리
(연산) → 출력



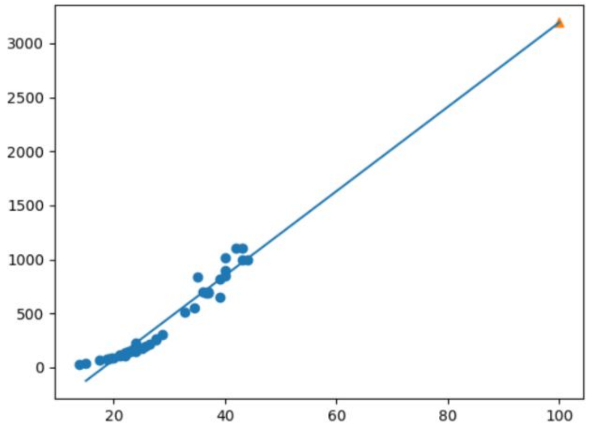
단순선형 회귀



다중선형 회귀

선형회귀

농어의 무게



농어의 길이

단순 선형 회귀

독립변수

$$H(x) = \underbrace{W}_{\text{가중치 (WEIGHT)}}x + \underbrace{b}_{\text{편향 (BIAS)}}$$

다중 선형 회귀

$$y = W_1\underbrace{x_1 + W_2x_2 + \dots + W_nx_n}_{\text{독립변수}} + b$$

다항 회귀

$$y = 1.014x^2 - 21.5579x + 116.05$$

최소제곱법

최소제곱법

더 정확한 기울기와 Y절편 구하기

$f(x) = ax + b$ 기울기와 상수항의 값을 임의로 지정했을때,

|실제값 - 예상값| $e_i = |y_i - f(x_i)|$ 들의 합

$$E = \sum_{i=1}^{n}[y_i - ax_i - b]^2$$: 최소제곱합

E가 최소일때 => $\frac{\partial E}{\partial a} = 0$ and $\frac{\partial E}{\partial b} = 0$

최소제곱법 _ 행렬계산

최소제곱법 _행렬계산

예시

P = { (1,1),(2,3), (3,4), (4,6), (5,5) }

$$Y=\begin{pmatrix}1\\3\\4\\6\\5\end{pmatrix} \quad A=\begin{pmatrix}1&1\\2&1\\3&1\\4&1\\5&1\end{pmatrix} \quad X=\begin{pmatrix}a\\b\end{pmatrix}$$

$$X=(A^TA)^{-1}A^TY$$

$$A^TA=\begin{pmatrix}1&2&3&4&5\\1&1&1&1&1\end{pmatrix}\begin{pmatrix}1&1\\2&1\\3&1\\4&1\\5&1\end{pmatrix}=\begin{pmatrix}55&15\\15&1\end{pmatrix}$$

$$X=\begin{pmatrix}55&15\\15&1\end{pmatrix}^{-1}\begin{pmatrix}1&2&3&4&5\\1&1&1&1&1\end{pmatrix}\begin{pmatrix}1\\3\\4\\6\\5\end{pmatrix}=\begin{pmatrix}1.1\\0.5\end{pmatrix}$$

단순선형회귀 실습 01

단순선형회귀 실습 01

```python
import numpy as np

# x값과 y값
x = [2,4,6,8]
y = [81,93,91,97]

# x,y의 평균값
mx = np.mean(x)
my = np.mean(y)

#기울기의 분모
divisor = sum([(mx - i)**2 for i in x])

# 기울기의 분자
def top(x,mx,y,my):
  d=0
  for i in range(len(x)):
    d += (x[i] - mx) * (y[i] - my)
  retrun d
dividend = top(x,mx,y,my)

a = dividend / divisor
b = my - (mx*a)
```

실행결과

x의 평균값: 5.0 y의 평균값: 90
기울기 a = 2.3 y 절편 b = 79.0

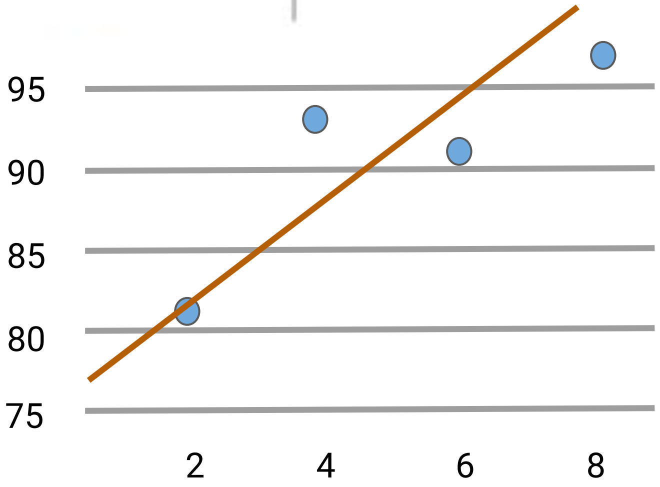
$$H(x) = Wx + b$$

기울기

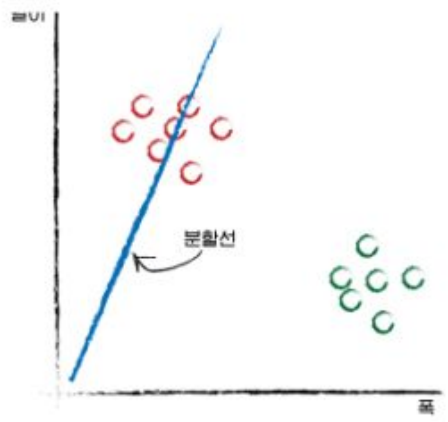
$$W = \frac{(x - x\text{ 평균})(y - y\text{ 평균})\text{의 합}}{(x - x\text{ 평균})^2\text{의 합}}$$

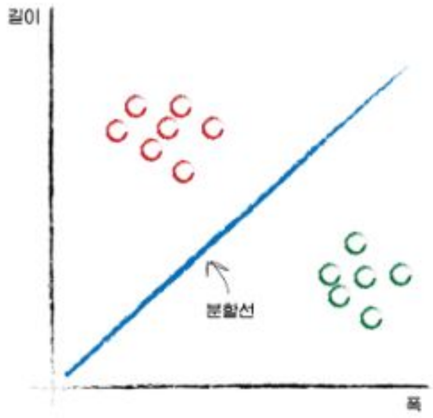
Y 절편

$$b = y\text{의 평균} - (x\text{의 평균} \times \text{기울기})$$



예측자와 분류자





[신경망 첫걸음: 수포자도 이해하는 신경망 동작 원리와 딥러닝 기초 - 타리크 라시드 - Google 도서]

오차 수정

선형회귀를 이용하여 데이터를 분석할 경우, 실제 값과 예측값 사이의 오차가 발생한다.
이러한 오차가 얼마나 큰지 계산하는 것으로 더욱 정확한 예측이 가능하다.

$$오차 = 예측값 - 실제값$$

오차 수정

$$\text{오차의합} = \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

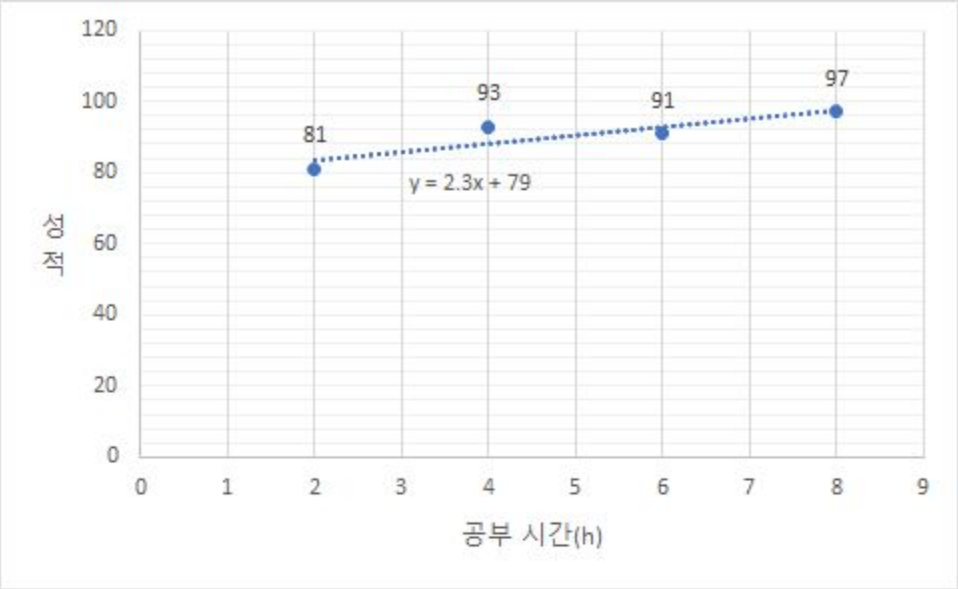
y_i = 실제 값
\hat{y}_i = 예측 값
n = x 원소의 총 개수
i = 원소의 순서

오차 수정

$$평균제곱오차(MSE) = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$$

y_i = 실제 값
\hat{y}_i = 예측 값
n = x 원소의 총 개수
i = 원소의 순서

오차 수정



x	y	예측값	오차	오차제곱
2	81	83.6	2.6	6.76
4	93	88.2	-4.8	23.04
6	91	92.8	1.8	3.24
8	97	97.4	0.4	0.16
오차 제곱의 평균		8.3		

y = 2.3x + 79

a(기울기) = 2.3
b(y절편) = 79

MSE = 8.3

오차 수정

엥? 이거 완전 '분산' 아니냐?

오차 수정

$$\text{평균제곱오차}(MSE) = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2 = Var(\hat{y}) + Bias(\hat{y})^2 + Var(\epsilon)$$

Var(\hat{y}) = 분산
Bias(\hat{y})^2 = 편향의 제곱
Var(\epsilon) = 노이즈(irreducible value)

오차 수정

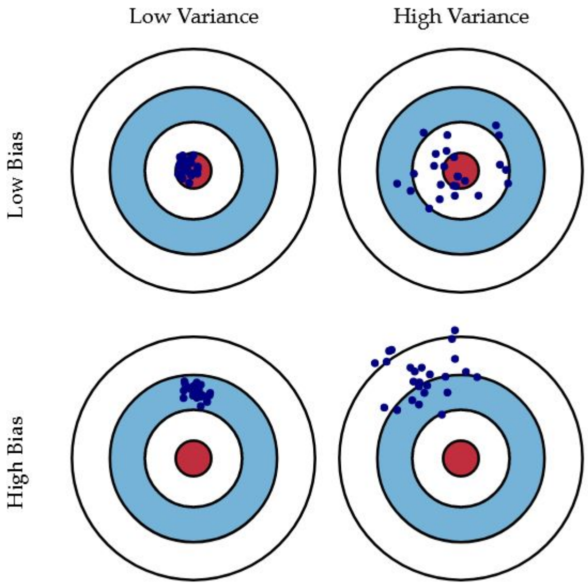


Fig. 1 Graphical illustration of bias and variance.

https://scott.fortmann-roe.com/docs/BiasVariance.html

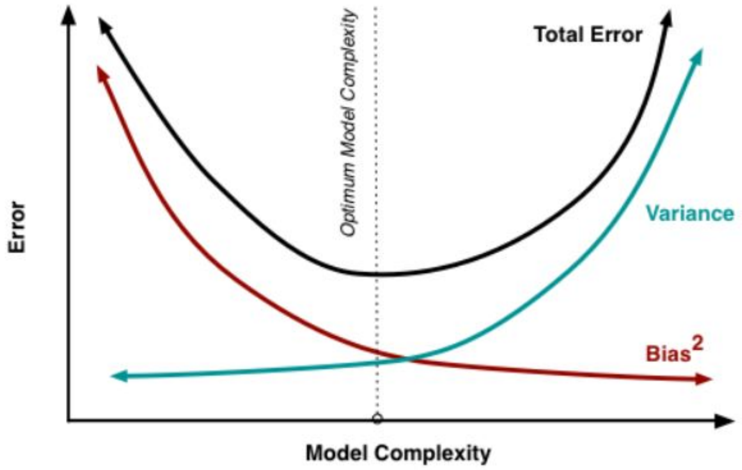


Fig. 6 Bias and variance contributing to total error.

오차 수정

1. 데이터 설정

- $a = 3, b = 76$으로, `fake_a_b` 라는 리스트를 선언하여 사용해보자.

```
fake_a_b = [3, 76]
```

- `data` 리스트를 만들어, (x, y) 값을 저장하고, `x` 와 `y` 변수에 for문을 사용하여 저장한다.

```
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]    ##data리스트의 순서쌍 중 첫 번째 값을 저장
y = [i[1] for i in data]    ##data리스트의 순서쌍 중 두 번째 값을 저장
```

오차 수정

2. 내부 함수 구성

> 일차방정식 $y = ax + b$ 구현

```
def predict(x):
  return fake_a_b[0] * x + fake_a_b[1]
  ##fake_a_b리스트의 첫 번째 원소를 a, 두 번째 원소를 b로 삼음
```

> 평균 제곱 오차 공식 함수 구현

```
def mse(y, y_hat):
  return ((y-y_hat) ** 2).mean())   ##.mean() : numpy 라이브러리에서 평균을 구해주는 함수
```

> `mse()` 함수에 값을 대입해주는 함수 구현

```
def mse_val(y, predict_result):
  return mse(np.array(y), np.array(predict_result)) ##y는 실제값, predict_result는 예측값
```

오차 수정

3. main 함수 작성

```
predict_result = []   ##예측값을 저장할 빈 리스트 선언

for i in range(len(x)):   ##독립변수 x의 수 만큼 반복
  predict_result.append(predict(x[i]))    ##predict_result 리스트에 predict() 함수에 x값 대입한 결과값 추가
  print("공부시간 = %.f, 실제점수 = %.f, 예측점수 = %.f" %(x[i], y[i], predict(x[i])))    ##결과 출력
```

4. MSE값 출력

```
print("mse 최종값: " + str(mse_val(predict_result, y)))
##print문에서 문자열과 결합하여 출력하므로 str()함수 사용
```

감사합니다!