AI 프로그래밍

- 2주차

인하 공전 컴퓨터 정보과

민 정혜

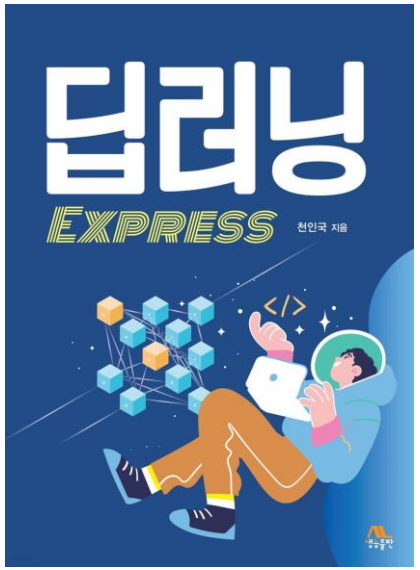
오늘 수업 순서

- 지난 주 내용 복습
- 머신 러닝 (machine learning)
- 회귀 (regression)
- 선형 회귀 실습 (linear regression)

교재

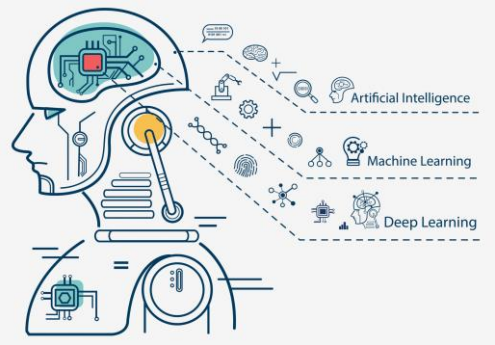
- **주교재**
  - 모두의 딥러닝,길벗, 2020

- **부교재**
  - 딥러닝 express, 생능 출판사,2021
  - 부분적으로 사용할 계획

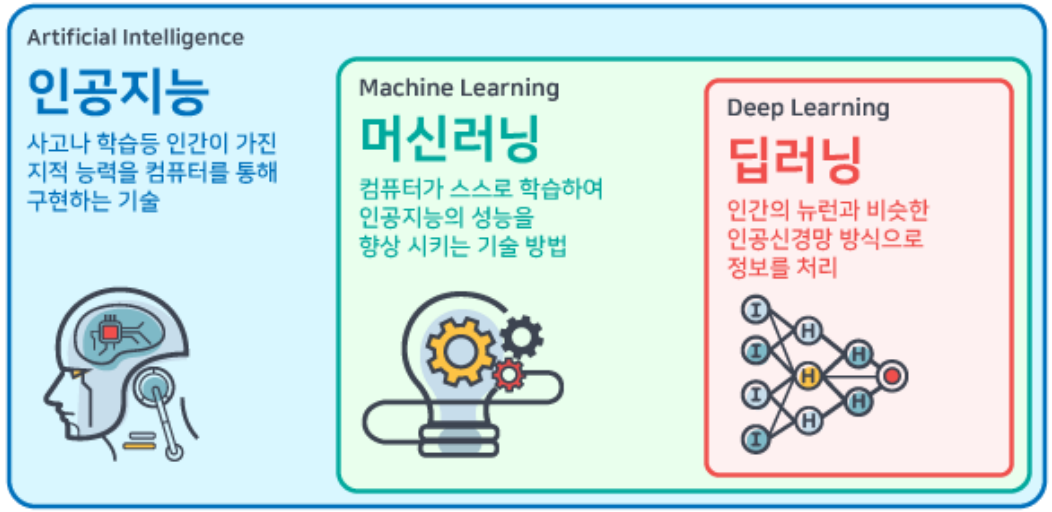


인공 지능 이란

- **인공 지능** (Artificial Intelligence)
  - **인간**이 가진 **지적 능력**을 **컴퓨터**를 통해 구현 하는 기술
- **머신 러닝** (Machine Learning)
  - 데이터와 **알고리즘**을 통해 컴퓨터를 **학습** 시켜 인공 지능의 성능을 향상 시키는 기술
- **딥 러닝** (Deep Learning)
  - **신경망 (Neural Network)** 을 이용한 머신 러닝 방법.
  - 여러 층으로 이루어진 신경망을 사용.



Machine Learning: A Primer to Laboratory Applications (thermofisher.com)



https://m.blog.naver.com/pwj6971/221614497987

인공 지능 적용 분야

자율주행 자동차

테슬라,현대 자동차

AI 비서

AI 스피커, 가상 비서

챗 봇

AS 상담 챗봇, 호텔 예약 챗봇

로봇

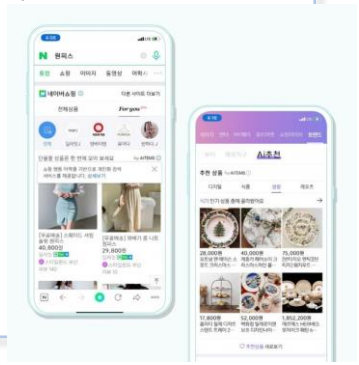
청소로봇, 교육용 로봇

영상 인식

얼굴 인식, 번호판 인식,

개인화 추천

영화 추천, 광고 추천,
상품 추천, 게임 운영



기계 번역

구글 번역, 네이버, 파파고

papago

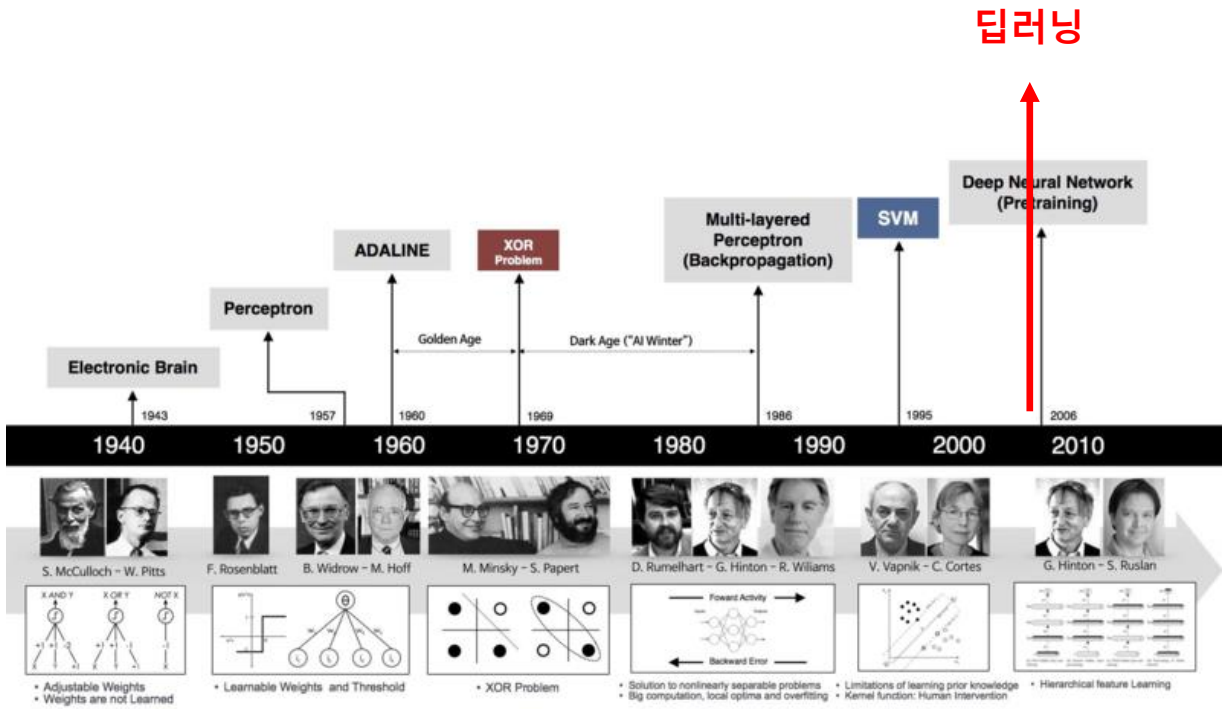
Google Translate

의료

질병 진단

인공지능의 역사

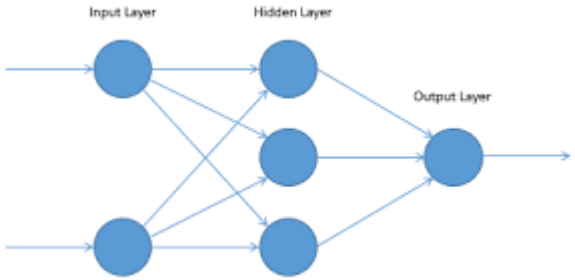
- 1950년대에 인공 지능의 개념과 신경망의 기본 개념인 퍼셉트론 개념이 성립됨.
- 2000년까지 신경망을 사용하지 않은 머신 러닝 사용됨.
- 2006 년부터 신경망을 이용한 딥 러닝 (Deep Learning)이 급속히 발전함.



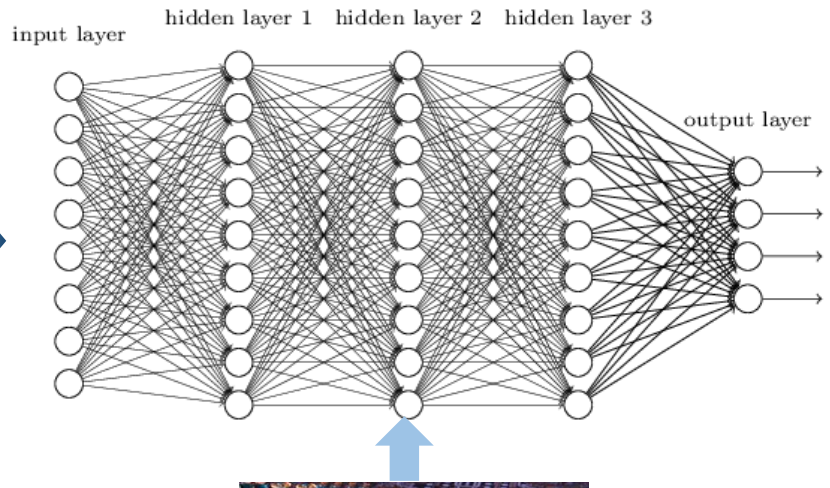
딥 러닝 (Deep Learning) 발전 이유

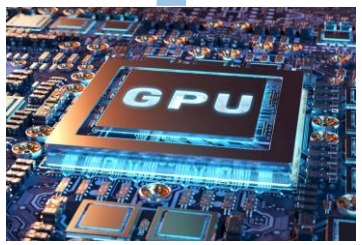
- **하드웨어의 발전**. 강력한 **GPU**는 딥러닝에서 복잡한 연산에 소요되는 시간을 크게 단축
- **빅 데이터**. 대량으로 쏟아져 나오는 데이터들을 학습에 활용 가능
- 신경망 이론의 단점 해결

small neural network

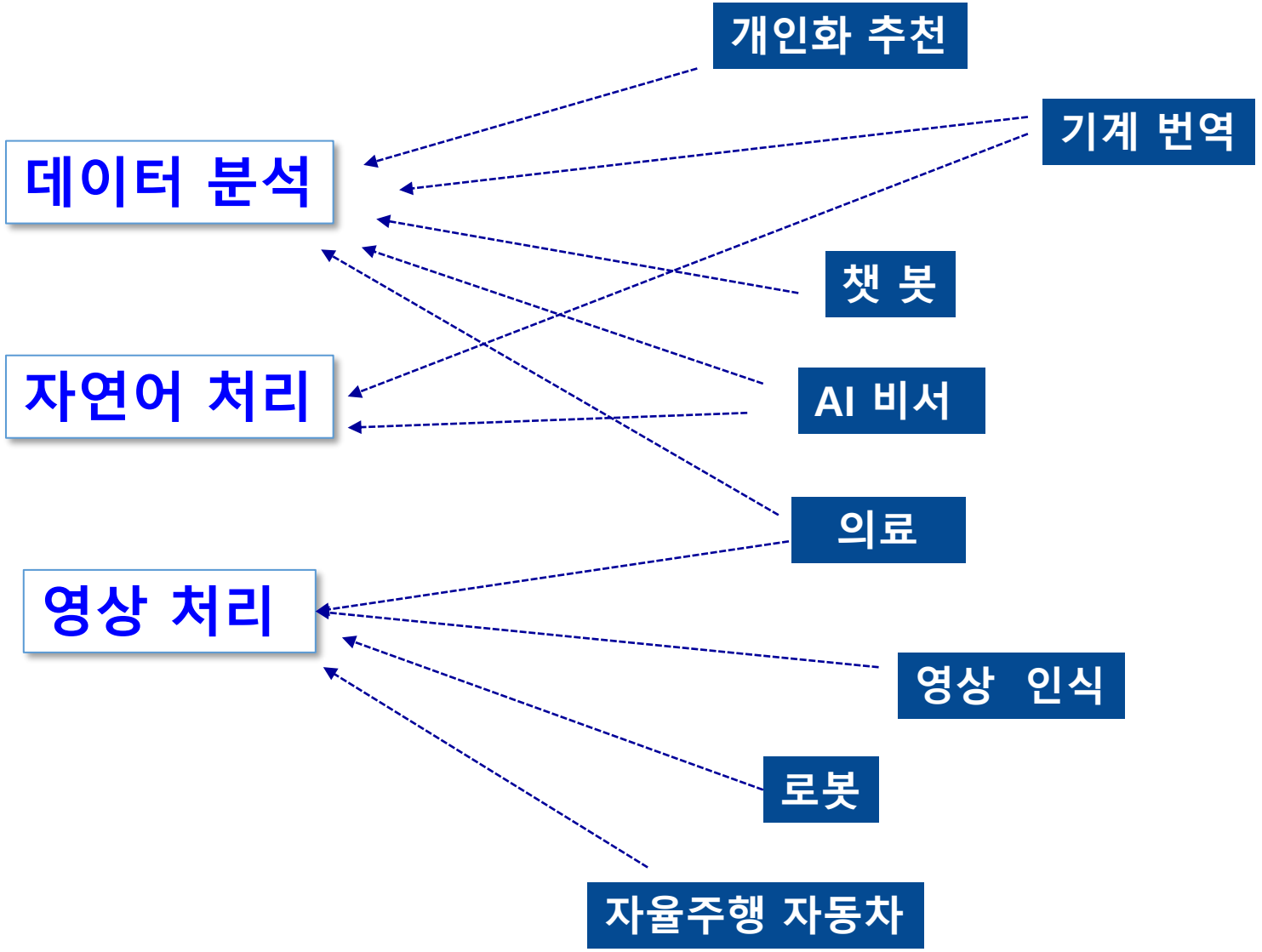


Deep Learning





인공 지능 기술 분야



인공 지능(머신 러닝) 분야 프로그래밍 환경

2010 년 이전

- 기본 수식 구현 부터 대부분의 기능을 c로 구현
- 프로그래밍 환경 설정이 쉬움.

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

```r
main <- function() {
    header <- c("sepal_length", "sepal_width", "petal_length", "petal_width", "class")
    iris <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
    names(iris) <- header
    sample <- sample.int(n = nrow(iris), size = floor(.75*nrow(iris)), replace = F)
    train <- iris[sample, ]
    test  <- iris[-sample, ]

    cl <- makePSOCKcluster(3)
    registerDoParallel(cl)
    lr <- multinom(class ~ ., data = train)
    lda_model <- lda(class ~ ., data = train)

    man_grid <- expand.grid(k = c(1:10))
    ctrl <- trainControl(method="cv", number = 5)
    knn <- train(class ~ ., data = train, method = "knn", trControl = ctrl, tuneGrid = man_grid)

    man_grid <- expand.grid(cost = c(1:10))
    ctrl <- trainControl(method="cv", number = 5)
    svm <- train(class ~ ., data = train, method = "svmLinear2", trControl = ctrl, tuneGrid = man_grid)

    stopCluster(cl)

    # Evaluate
    lr_accuracy <- sum(predict(lr, test) == test$class) / nrow(test)
    lda_accuracy <- sum(predict(lda_model, test)$class == test$class) / nrow(test)
    knn_accuracy <- sum(predict(knn, test) == test$class) / nrow(test)
    svm_accuracy <- sum(predict(svm, test) == test$class) / nrow(test)
    print(paste(lr_accuracy, ",", lda_accuracy, ",", knn_accuracy, ",", svm_accuracy))
}
```

2010 년 이후

- 파이썬 기반으로 다양한 라이브러리 사용 가능
- 대부분의 기능이 라이브러리에 함수로 구현되어 있음.
- 목적에 따라서 라이브러리를 선택 하여 구현
- 프로그래밍 환경 설정이 복잡함

python™

Keras

Caffe

TensorFlow

PyTorch

머신 러닝 라이브러리

머신러닝

TensorFlow

Keras

PYTORCH
pytorch

scikit learn
Scikit learn

데이터 분석

pandas

seaborn

Matplotlib

자연어 처리

spaCy

Transformers
Hugging face

영상 처리

OpenCV

pillow

머신 러닝

- **데이터**와 **알고리즘**을 통해 컴퓨터를 **학습** 시켜 인공 지능의 성능을 향상 시키는 기술

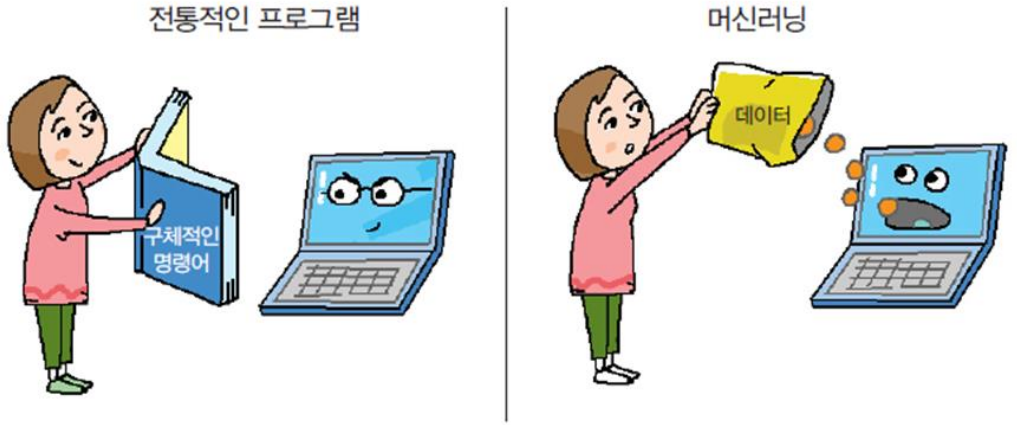
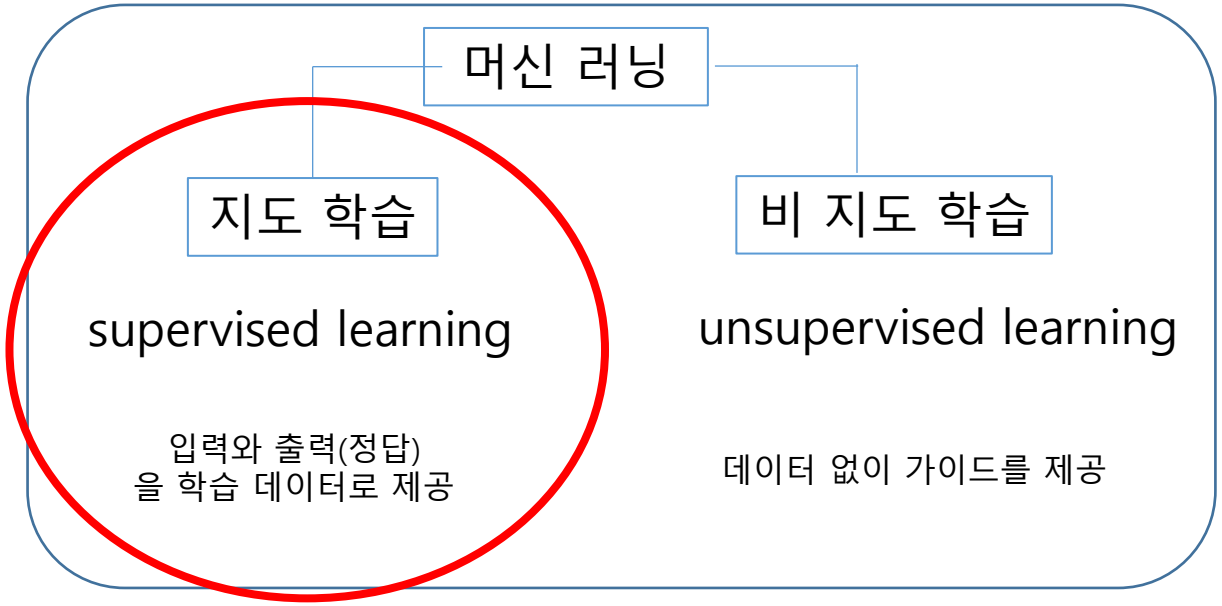


그림 3-2 전통적인 프로그래밍과 머신러닝의 차이점

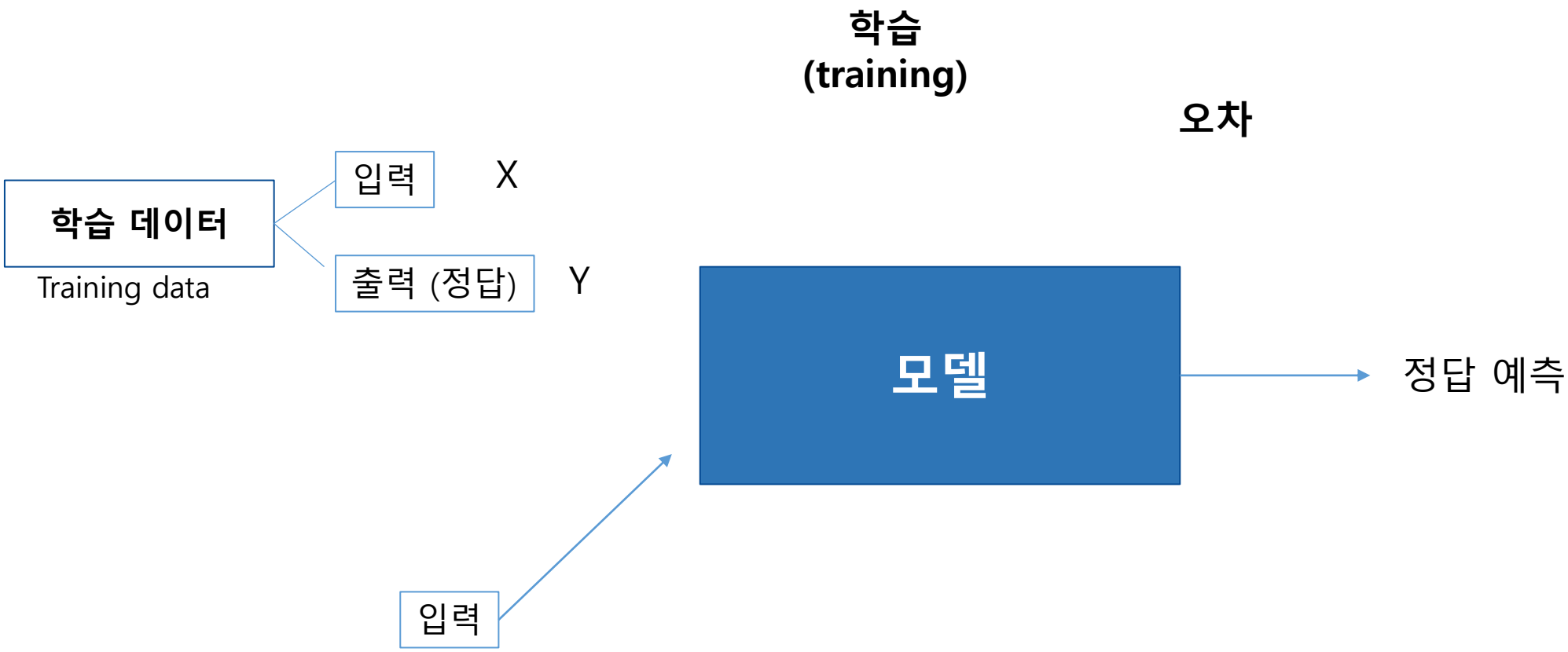
참고자료 : 딥러닝 express

머신 러닝

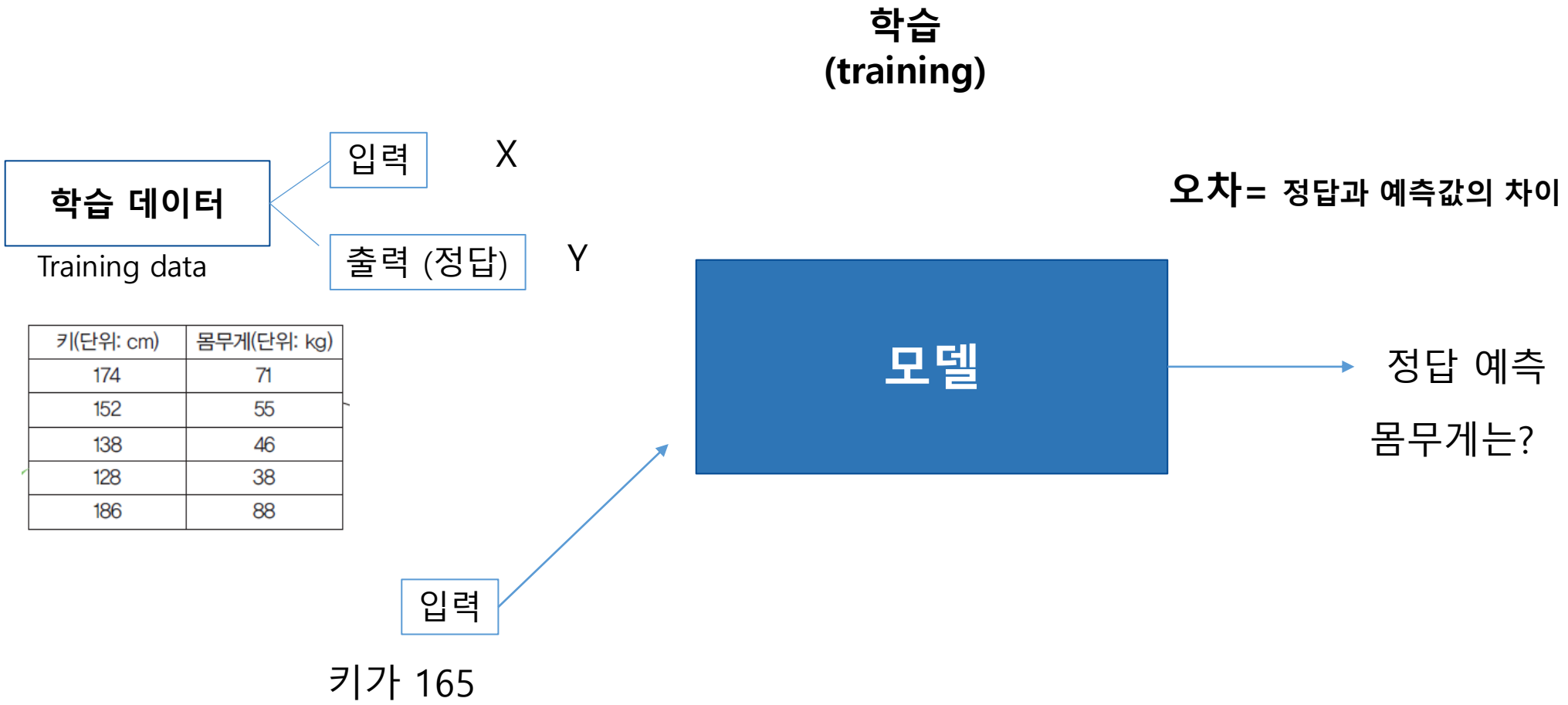

머신 러닝
지도 학습
비 지도 학습
supervised learning
unsupervised learning
입력와 출력(정답)
을 학습 데이터로 제공
데이터 없이 가이드를 제공


머신러닝
지도 학습
비지도 학습
회귀
• Linear
• polynomial
클러스터링
• SVD
• PAC
• K-means
판단트리
랜덤 포리스트
분류
• KNN
• Trees
• logistic Regression
• Naive-Bayes
• SVM

머신 러닝



머신 러닝 - 예제


학습
(training)
학습 데이터
Training data
입력
X
출력 (정답)
Y
키(단위: cm)
몸무게(단위: kg)
174
71
152
55
138
46
128
38
186
88
입력
키가 165
모델
오차= 정답과 예측값의 차이
정답 예측
몸무게는?

회귀 (regression)

- **데이터를 이용하여 결과를 예측 하는 함수를 도출 y=f(x)**
- **선형 회귀**
  - 입력 값 x가 1차원일때
  - 직선의 방정식
  - Y=ax+b

⇒Y =wx+b, w: weight, b: bias

키(단위: cm)	몸무게(단위: kg)
174	71
152	55
138	46
128	38
186	88

그림 4-2 회귀의 종류

참고자료 : 딥러닝 express

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4

선형 회귀 예제

- 공부한 시간(x)과 성적 (y)
- Y=wx+b 로 예측
- 오차를 최소화 하는 w와 b 찾기

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4

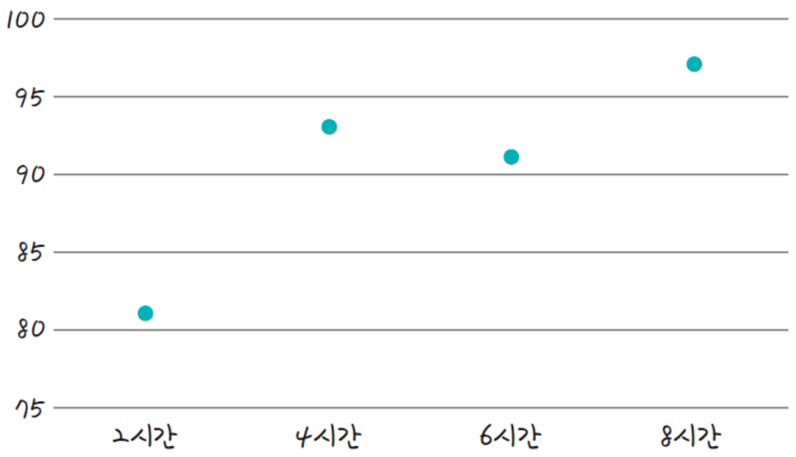


그림 3-4 공부한 시간과 성적의 관계도

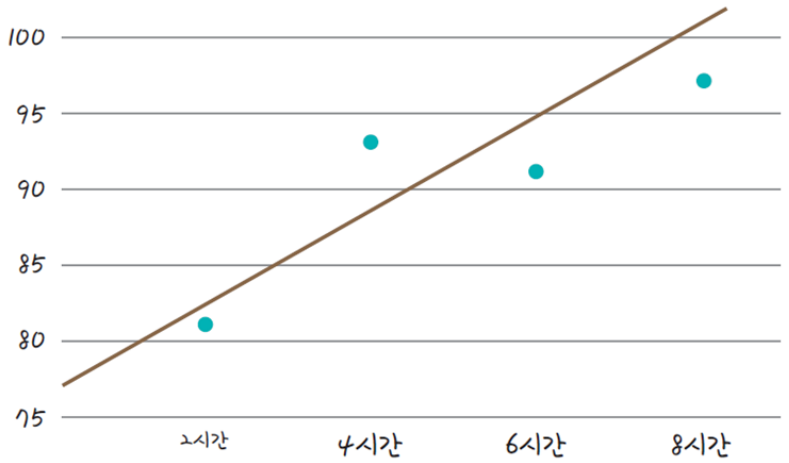


그림 3-5 임의의 직선 그려보기

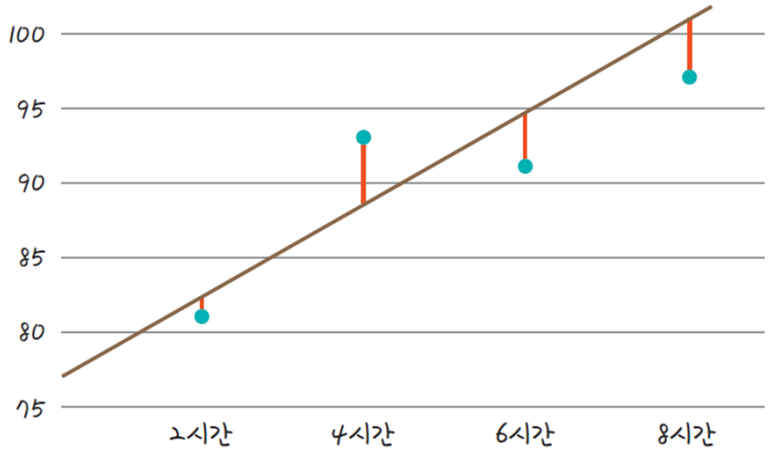


그림 3-6 임의의 직선과 실제 값 사이의 거리

오차

선형 회귀 예제

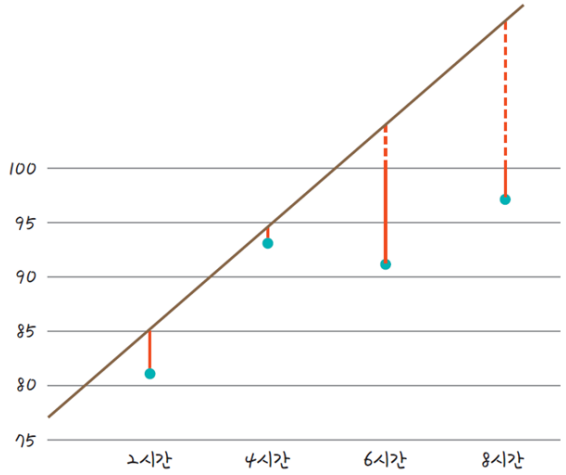


그림 3-7 기울기를 너무 크게 잡았을 때의 오차

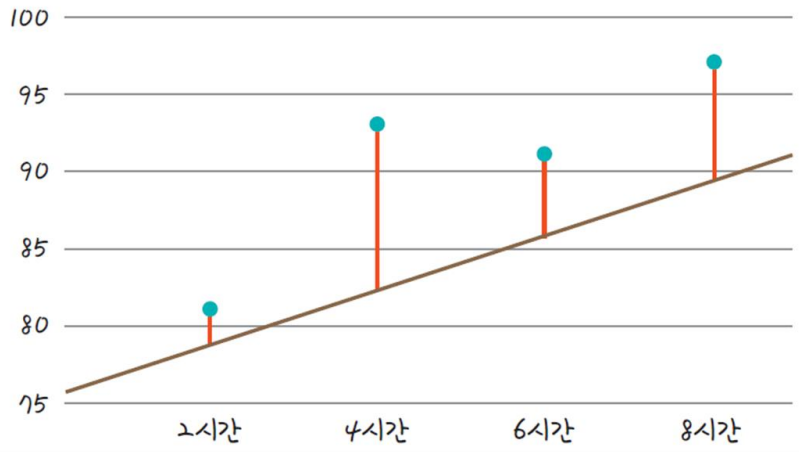


그림 3-8 기울기를 너무 작게 잡았을 때의 오차

선형 회귀 예제 실습

x	Y
0	3
1	3.5
2	5.5

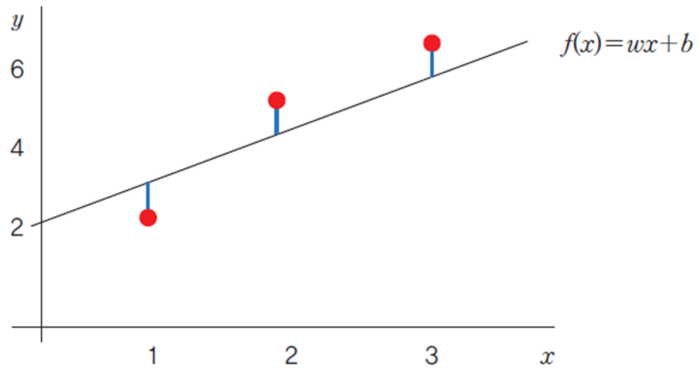
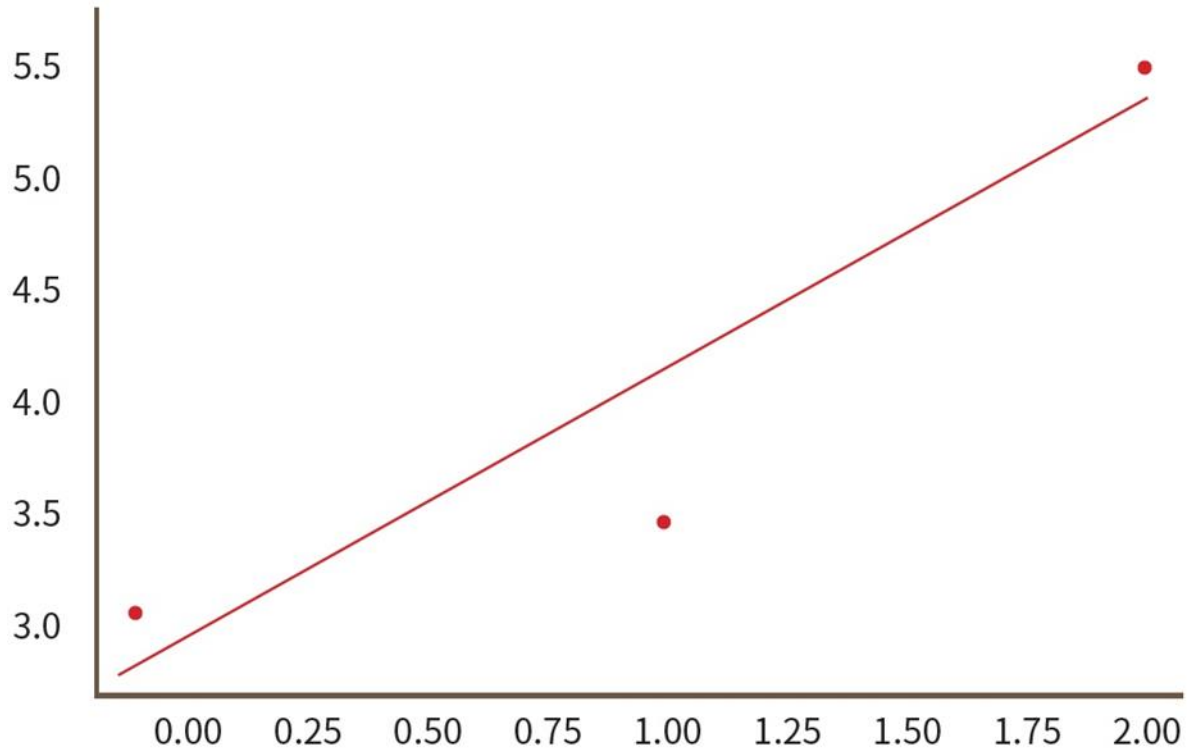


그림 4–5 데이터와 직선 간의 거리

```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([0.0, 1.0, 2.0])
y = np.array([3.0, 3.5, 5.5])

w = 0       # 기울기
b = 0       # 절편

lrate = 0.01  # 학습률
epochs = 1000  # 반복 횟수

n = float(len(X)) # 입력 데이터의 개수

# 경사 하강법
for i in range(epochs):
    y_pred = w*X + b                                  # 선형 회귀 예측값
    dw = (2/n) * sum(X * (y_pred-y))                  # 넘파이 배열간의 산술 계산은 요소별로 적용
    db = (2/n) * sum(y_pred-y)                        # sum()은 모든 요소들의 합을 계산하는 내장 함수
    w = w - lrate * dw                                # 기울기 수정
    b = b - lrate * db                                # 절편 수정

# 기울기와 절편을 출력한다.
print (w, b)

# 예측값을 만든다.
y_pred = w*X + b

# 입력 데이터를 그래프 상에 찍는다.
plt.scatter(X, y)

# 예측값은 선그래프로 그린다.
plt.plot([min(X), max(X)], [min(y_pred), max(y_pred)], color='red')
plt.show()
```

참고자료 : 딥러닝 express

선형 회귀 예제



```
import numpy as np
import matplotlib.pyplot as plt


def mse(y,y_hat):
    return ((y-y_hat)**2).mean()

def mse_val(y,predict_result):
  return mse(np.array(y),np.array(predict_result))



X = np.array([0.0, 1.0, 2.0])
y = np.array([3.0, 3.5, 5.5])

w = 0       # 기울기
b = 0       # 절편

lrate = 0.01  # 학습률
epochs = 500  # 반복 횟수

n = float(len(X)) # 입력 데이터의 개수
```

```
# 경사 하강법
for i in range(epochs):
    y_pred = w*X + b         # 선형 회귀 예측값
    dw = (2/n) * sum(X * (y_pred-y))  # 넘파이 배열간의 산술 계산은 요소별로 적용
    db = (2/n) * sum(y_pred-y)    # sum()은 모든 요소들의 합을 계산하는 내장 함수
    w = w - lrate * dw       # 기울기 수정
    b = b - lrate * db       # 절편 수정
    if (i%50==0):
      print('iteration %3d: loss  %4.2f w %3.2f b %3.2f '%(i,mse(y,y_pred),w,b))


# 기울기와 절편을 출력한다.
print ('########## final w,b',w, b)

# 예측값을 만든다.
y_pred = w*X + b


# 입력 데이터를 그래프 상에 찍는다.
plt.scatter(X, y)

# 예측값은 선그래프로 그린다.
plt.plot([min(X), max(X)], [min(y_pred), max(y_pred)], color='red')
plt.show()
```

첫번째 loss 값과 w,b 값을 저에게 채팅으로 보내 주세요.

선형 회귀 예제 실습

x	Y
0	3
1	3.5
2	5.5

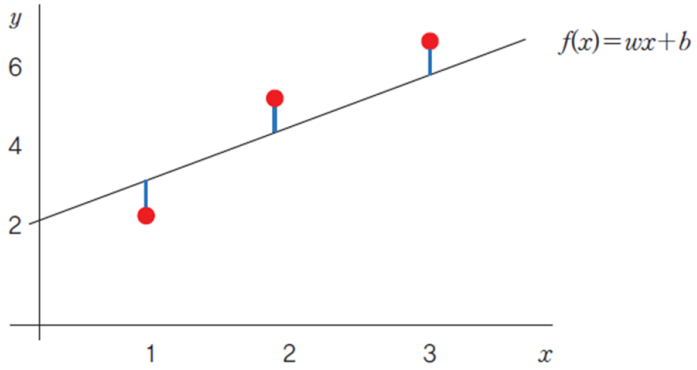
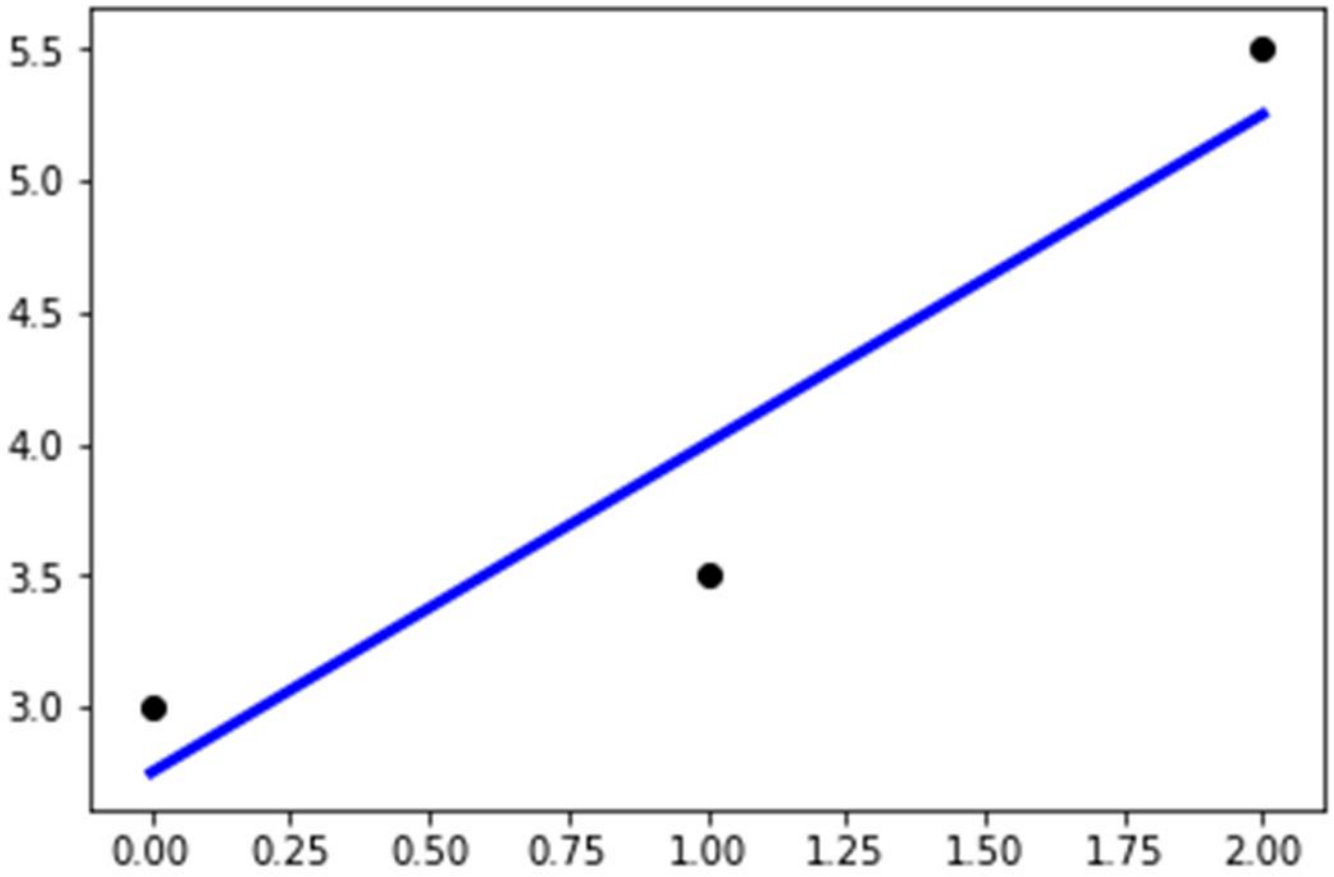


그림 4-5 데이터와 직선 간의 거리

```
import matplotlib.pylab as plt
from sklearn import linear_model

# 선형 회귀 모델을 생성한다.
reg = linear_model.LinearRegression()

# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
X = [[0], [1], [2]]                    # 반드시 2차원으로 만들어야 함
y = [3, 3.5, 5.5]                      # y = x + 3

# 학습을 시킨다.
reg.fit(X, y)

print(reg.coef_)                       # 직선의 기울기
print(reg.intercept_)          # 직선의 y-절편
print(reg.score(X, y))

print(reg.predict([[5]]))

# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X, y, color='black')

# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X)

# 학습 데이터와 예측값으로 선그래프로 그린다.
# 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
plt.plot(X, y_pred, color='blue', linewidth=3)
plt.show()
```

참고자료 : 딥러닝 express

선형 회귀 예제



선형 회귀 예제 실습

키(단위: cm)	몸무게(단위: kg)
174	71
152	55
138	46
128	38
186	88

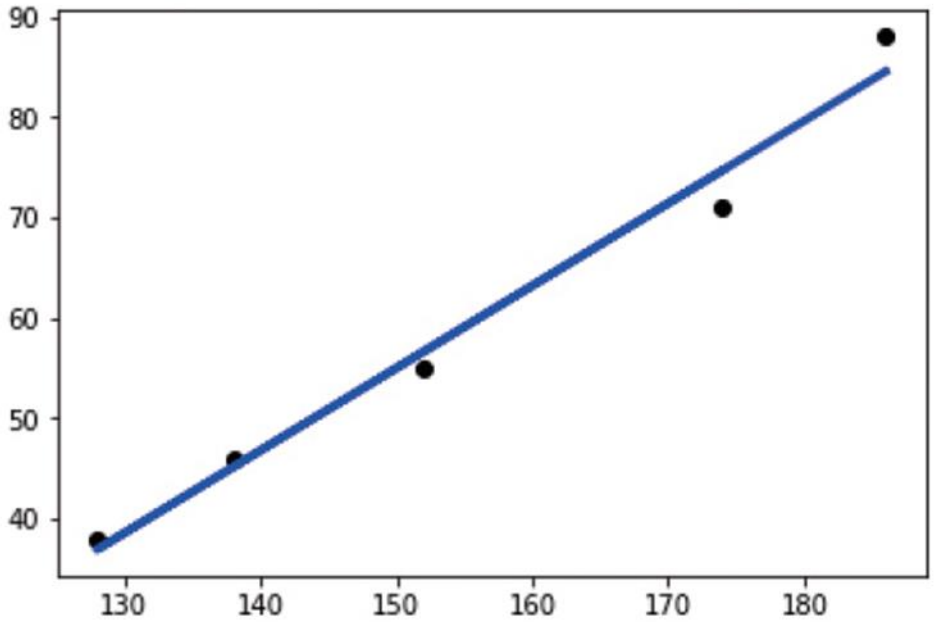
키가 160일때의 몸무게 예측 값을 보내 주세요

```
import matplotlib.pylab as plt
from sklearn import linear_model

reg = linear_model.LinearRegression()

X = [[174], [152], [138], [128], [186]]
y = [71, 55, 46, 38, 88]

reg.fit(X, y)                                    # 학습

print(reg.predict([[165]]))

# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X, y, color='black')

# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X)

# 학습 데이터와 예측값으로 선그래프로 그린다.
# 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
plt.plot(X, y_pred, color='blue', linewidth=3)
plt.show()
```

참고자료 : 딥러닝 express

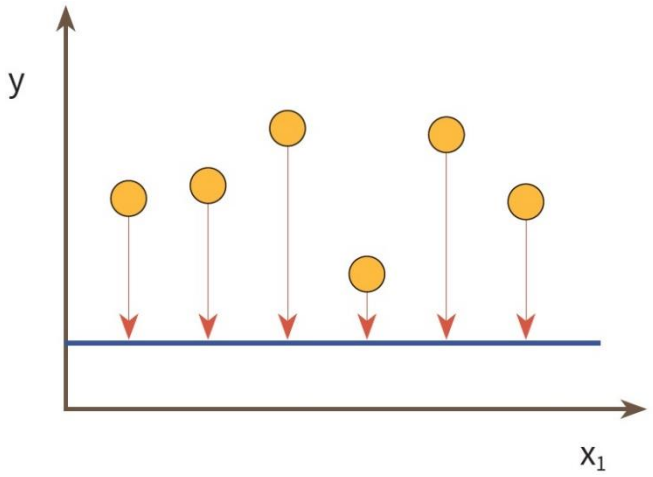
선형 회귀 예제

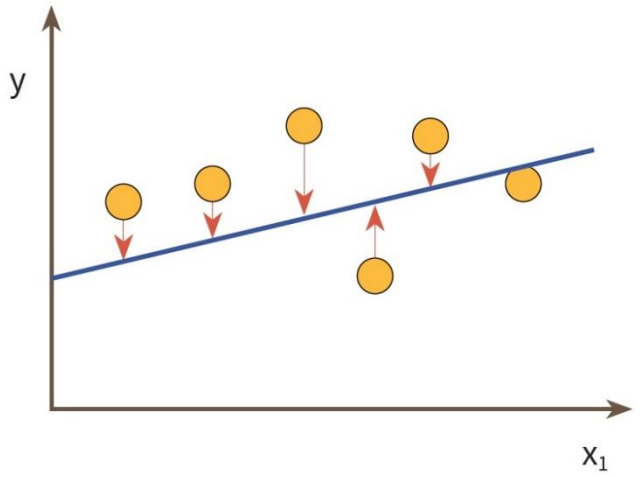

```
[67.30998637]
```



선형 회귀 예제

- 부모의 키와 자녀의 키의 관계 조사
- 면적에 따른 주택의 가격
- 연령에 따른 실업율 예측
- 공부 시간과 학점 과의 관계
- CPU 속도와 프로그램 실행 시간 예측





참고자료 : 딥러닝 express

Numpy 배열 slicing 연습

- **import numpy as np**

```
a = np.arange(10)
print(a)
# [0 1 2 3 4 5 6 7 8 9]

print(a[4:8])
# [4 5 6 7]

print(a[-5:-2])
# [5 6 7]

print(a[::-1])
# [9 8 7 6 5 4 3 2 1 0]
```

```
a[3:6] = 100
print(a)
# [  0   1   2 100 100 100   6   7   8   9]

a[3:6] = [100, 200, 300]
print(a)
# [  0   1   2 100 200 300   6   7   8   9]
```

```
# a[3:6] = [100, 200, 300, 400]
# ValueError: cannot copy sequence with size 4 to array axis with
dimension 3
```

https://engineer-mole.tistory.com/74

Numpy slicing


```
a = np.arange(10)
print(a)
# [0 1 2 3 4 5 6 7 8 9]

print(a[2:8:2])
# [2 4 6]

a[2:8:2] = 100
print(a)
# [  0   1 100   3 100   5 100   7   8   9]

a[2:8:2] = [100, 200, 300]
print(a)
# [  0   1 100   3 200   5 300   7   8   9]
```

Numpy slicing


```
a = np.arange(12).reshape((3, 4))
print(a)
# [[ 0  1  2  3]
#  [ 4  5  6  7]
#  [ 8  9 10 11]]
```

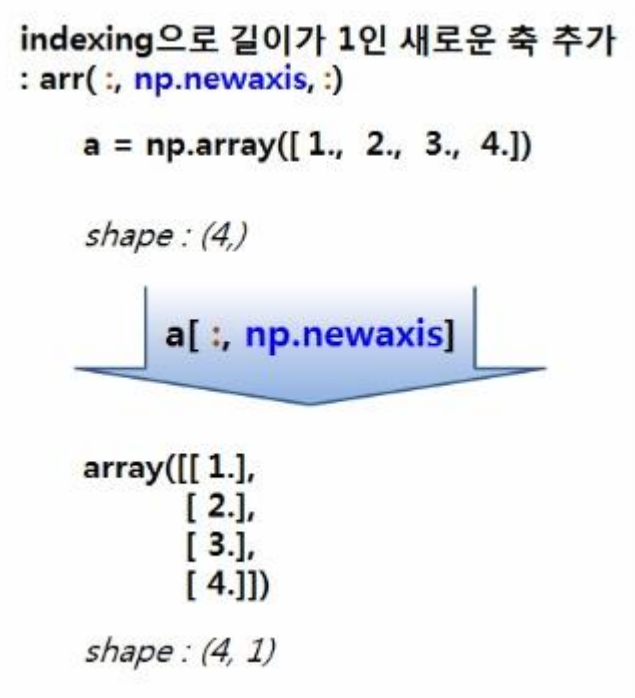
```
print(a[1:, 1:3])
# [[ 5  6]
#  [ 9 10]]
```

```
print(a[1:, :])
# [[ 4  5  6  7]
#  [ 8  9 10 11]]

print(a[1:])
# [[ 4  5  6  7]
#  [ 8  9 10 11]]
```

Numpy slicing


```
a = np.arange(12).reshape((3, 4))
print(a)
# [[ 0  1  2  3]
#  [ 4  5  6  7]
#  [ 8  9 10 11]]

print(a[1:, 1:3])
# [[ 5  6]
#  [ 9 10]]

a[1:, 1:3] = 100
print(a)
# [[  0   1   2   3]
#  [  4 100 100   7]
#  [  8 100 100  11]]

a[1:, 1:3] = [100, 200]
print(a)
# [[  0   1   2   3]
#  [  4 100 200   7]
#  [  8 100 200  11]]

a[1:, 1:3] = [[100, 200], [300, 400]]
print(a)
# [[  0   1   2   3]
#  [  4 100 200   7]
#  [  8 300 400  11]]
```

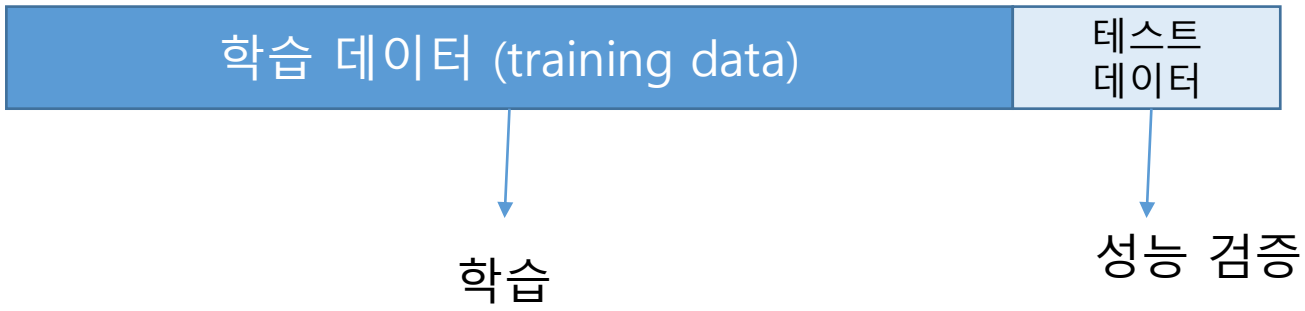
Numpy




```python
import numpy as np

arr = np.array([1, 2, 3, 4])
arr1=arr
arr2=arr[np.newaxis]
arr3=arr[:, np.newaxis]
print(arr1,arr1.shape)
print(arr2,arr2.shape)
print(arr3,arr3.shape)
```

```
arr1 [1 2 3 4] (4,)
arr2 [[1 2 3 4]] (1, 4)
arr3 [[1] [2] [3] [4]] (4, 1)
```

선형 회귀 예제 실습



선형 회귀 예제 실습 – 당뇨병 예제

Bmi와 혈당간의 관계 예측

```
import matplotlib.pylab as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model

# 당뇨병 데이터 세트를 적재한다.
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

print(diabetes_X.data.shape )
# 하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱스가 2이다.
diabetes_X_new = diabetes_X[:, np.newaxis, 2]
print(diabetes_X_new.data.shape )
```

선형 회귀 예제 실습 – 당뇨병 예제


```
import matplotlib.pylab as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model

# 당뇨병 데이터 세트를 적재한다.
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

print(diabetes_X.data.shape )
# 하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱스가 2이다.
diabetes_X_new = diabetes_X[:, np.newaxis, 2]
print(diabetes_X_new.data.shape )

# 학습 데이터와 테스트 데이터를 분리한다.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes_X_new, diabetes_y,
test_size=0.1, random_state=0)

regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

# 테스트 데이터로 예측해보자.
y_pred = regr.predict(X_test)

# 실제 데이터와 예측 데이터를 비교해보자.
# plt.plot(y_test, y_pred, '.')

plt.scatter(X_test, y_test,  color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.show()
```

수고 하셨습니다

jhmin@inhatc.ac.kr