

AI 프로그래밍

- 2024

3주차

■ 주 차 별 수업 내용

- 1주차 : 인공지능, 딥러닝, 머신러닝의 정의와 적용 분야
- 2주차 : 머신 러닝 개념, 회귀 (regression), 선형 회귀
- 3 주차 : 선형 회귀, 넘파이(Numpy), 맷플롯립(Matplotlib) 라이브러리 실습
- 4주차 : 경사 하강법 , 다중 선형 회귀
- 5주차 : 분류: 회귀 군집, 로지스틱 회귀
- 6주차 : 퍼셉트론, 활성화 함수, 다중 퍼셉트론
- 7주차: 다중 퍼셉트론, 딥러닝 개념
- 8주차 : 중간 평가

■ 주 차 별 수업 내용

- 9 주차 : 딥러닝 개념, 손실 함수
- 10-11주차 : 딥러닝 구현을 위한 개념
- 12 -13주차 : 컨볼루션 신경망
- 14주차: 순환 신경망
- 15주차 : 기말 평가

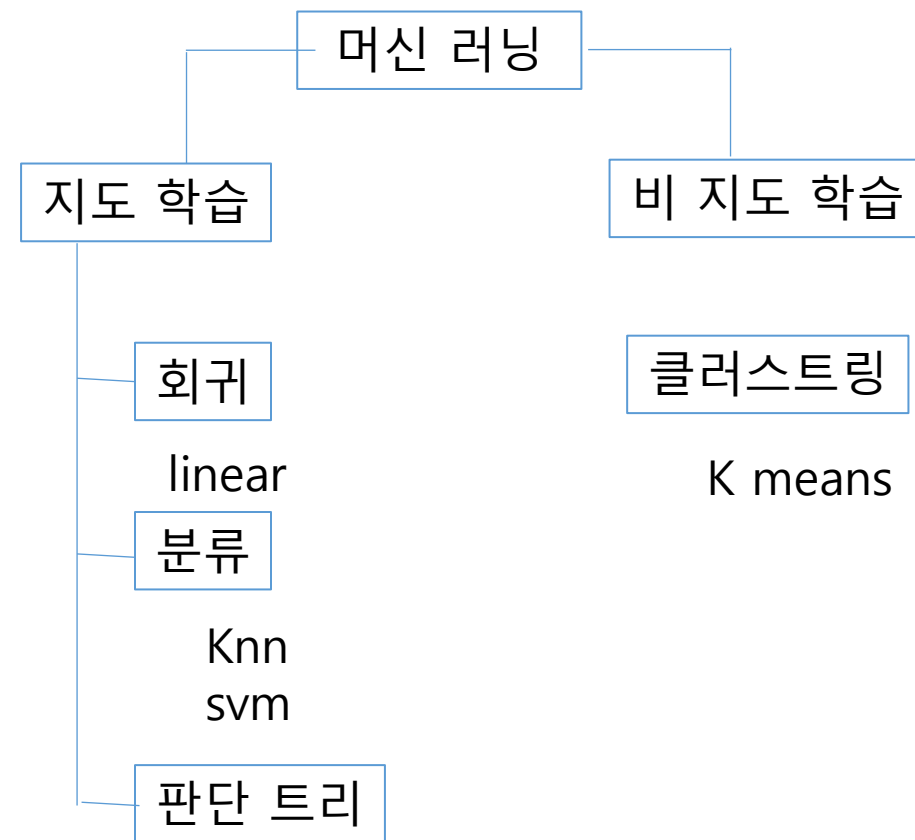
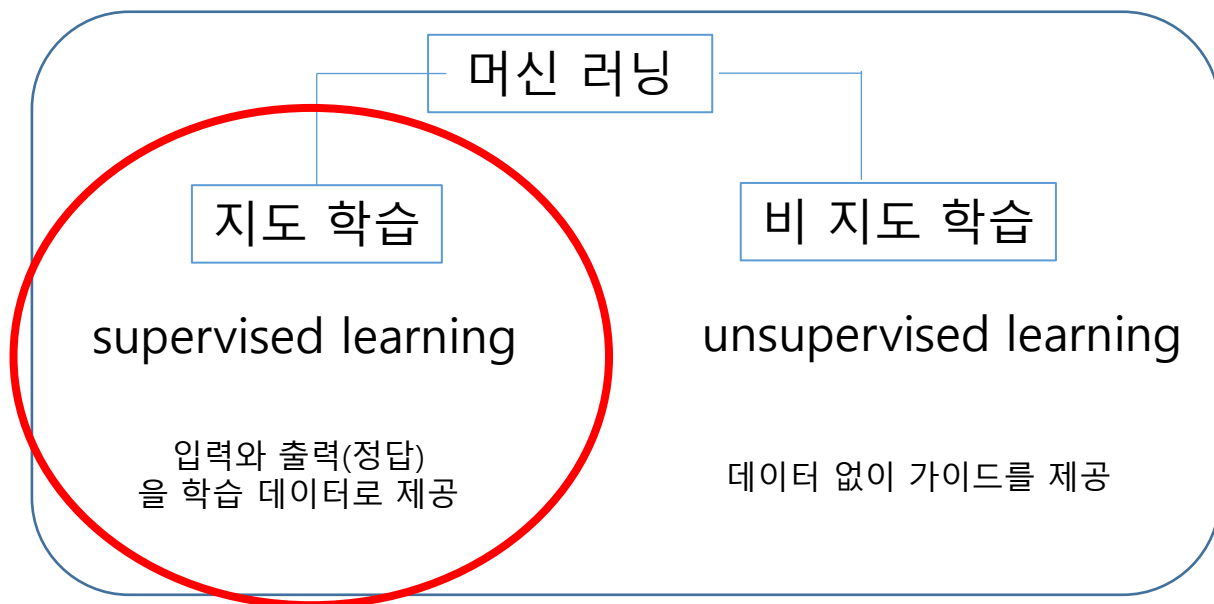
오늘 수업 순서

인하공전 컴퓨터 정보과

- 머신 러닝 개념 복습
- 선형 회귀 실습
- Numpy library

머신 러닝

인하공전 컴퓨터 정보과



지도학습 (Supervised Learning)

: 정답(입력 과 원하는 출력)을 제공 받음

비지도 학습(Unsupervised learning)

: 정답이 주어지지 않음. 입력 데이터에서
패턴을 찾는 학습

회귀 (regression)

: 입력과 출력이 실수

분류(classification)

: 입력을 레이블로 분할. 출력이 이산적(discrete)

1. 데이터 확보
2. 데이터 정제
3. 모델 학습
4. 모델 평가
5. 예측에 사용

선형 회귀 예제 실습

인하공전 컴퓨터 정보과

X	Y
0	3
1	3.5
2	5.5

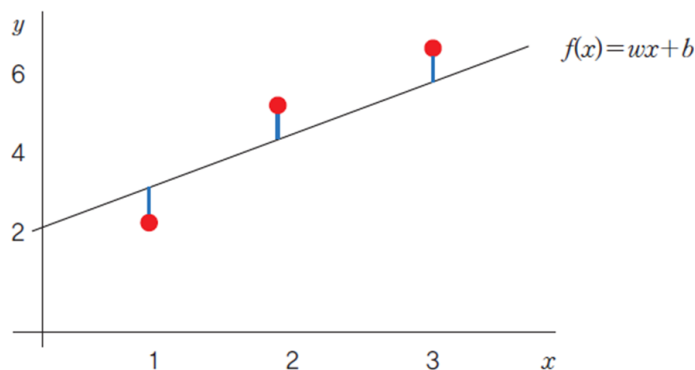


그림 4-5 데이터와 직선 간의 거리

```
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
```

```
lr_rate = 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
    dw = (2/n) * sum(X * (y_pred-y))
    db = (2/n) * sum(y_pred-y)
    w = w - lr_rate * dw
    b = b - lr_rate * db
```

```
# 기울기와 절편을 출력한다.
print (w, b)
```

```
# 예측값을 만든다.
y_pred = w*X + b
```

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

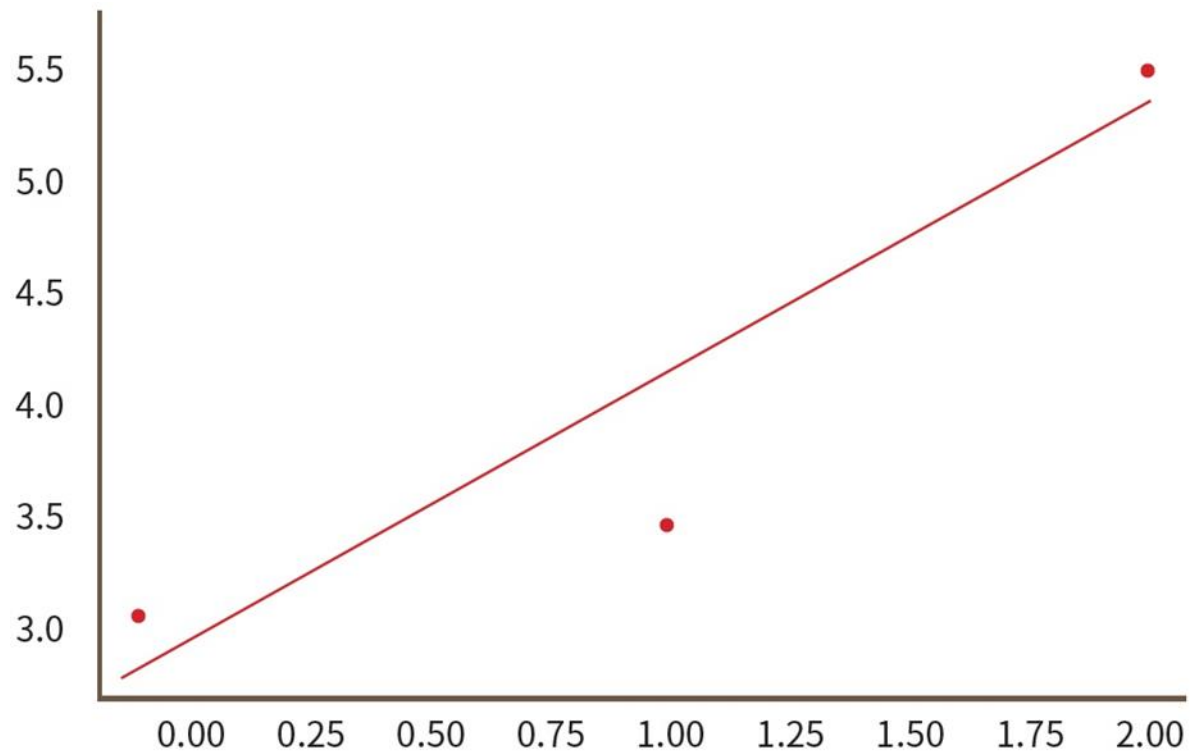
```
# 예측값은 선그래프로 그린다.
plt.plot([min(X), max(X)], [min(y_pred), max(y_pred)], color='red')
plt.show()
```

```
# 선형 회귀 예측값
# 넘파이 배열간의 산술 계산은 요소별로 적용
# sum()은 모든 요소들의 합을 계산하는 내장 함수
# 기울기 수정
# 절편 수정
```

선형 회귀 예제

인하공전 컴퓨터 정보과

x	Y
0	3
1	3.5
2	5.5



$$Y=1.252*x+2.745$$

X=3, 6.5

regression1_loss_noprint .ipynb

```
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
```

```
lr = 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 - lr * dw      # 기울기 수정
    b = b - lr * 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()
```

```

for i in range(epochs):
    y_pred = w*X + b      # 선형 회귀 예측값
    dw = (2/n) * sum(X * (y_pred-y)) # 넘파이 배열간
    의 산술 계산은 요소별로 적용
    db = (2/n) * sum(y_pred-y)  # sum()은 모든 요소
    들의 합을 계산하는 내장 함수
    w = w - lr * dw      # 기울기 수정
    b = b - lr * 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))

```

```

iteration 0: loss 17.17 w 0.10 b 0.08
iteration 50: loss 0.61 w 1.73 b 1.71
iteration 100: loss 0.33 w 1.73 b 2.05
iteration 150: loss 0.24 w 1.63 b 2.23
iteration 200: loss 0.19 w 1.53 b 2.36
iteration 250: loss 0.16 w 1.47 b 2.45
iteration 300: loss 0.15 w 1.41 b 2.52
iteration 350: loss 0.14 w 1.37 b 2.58
iteration 400: loss 0.13 w 1.34 b 2.62
iteration 450: loss 0.13 w 1.32 b 2.65
##### final w,b 1.3033228991130752
2.6760184293088694

```

1.25 2.74

선형 회귀(Linear Regression)

인하공전 컴퓨터 정보과

- $Y = w * X + b$

w : 직선의 기울기

b : y 절편

- 예측 값 = $w * X + b$

- `from sklearn import linear_model`

`reg = linear_model.LinearRegression()`

`reg.coef_` # 직선의 기울기, w
`reg.intercept_` # 직선의 y-절편, b

`reg.predict([[x]]))` # 예측값

x	Y
0	3
1	3.5
2	5.5

```
import matplotlib.pyplot 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, W
```

```
print(reg.intercept_)
```

```
# 직선의 y-절편, B
```

```
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()
```

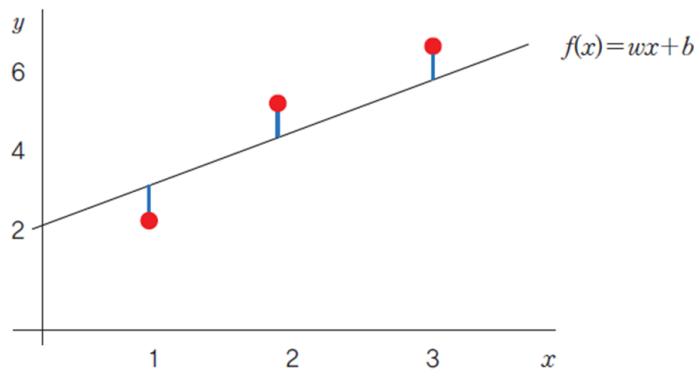
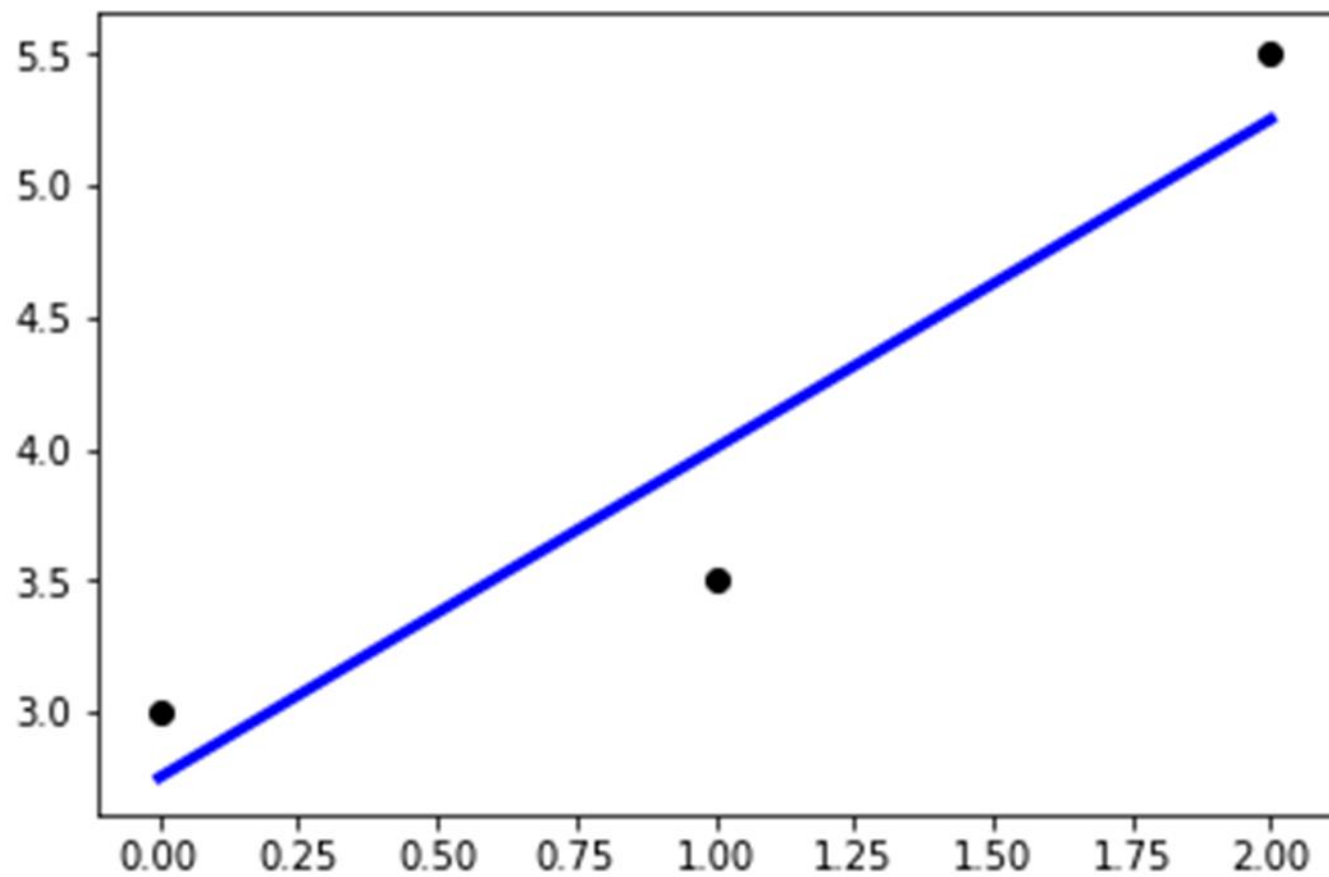


그림 4-5 데이터와 직선 간의 거리



선형 회귀 예제

인하공전 컴퓨터 정보과



선형 회귀 예제 실습

인하공전 컴퓨터 정보과

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

```
import matplotlib.pyplot 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')
```

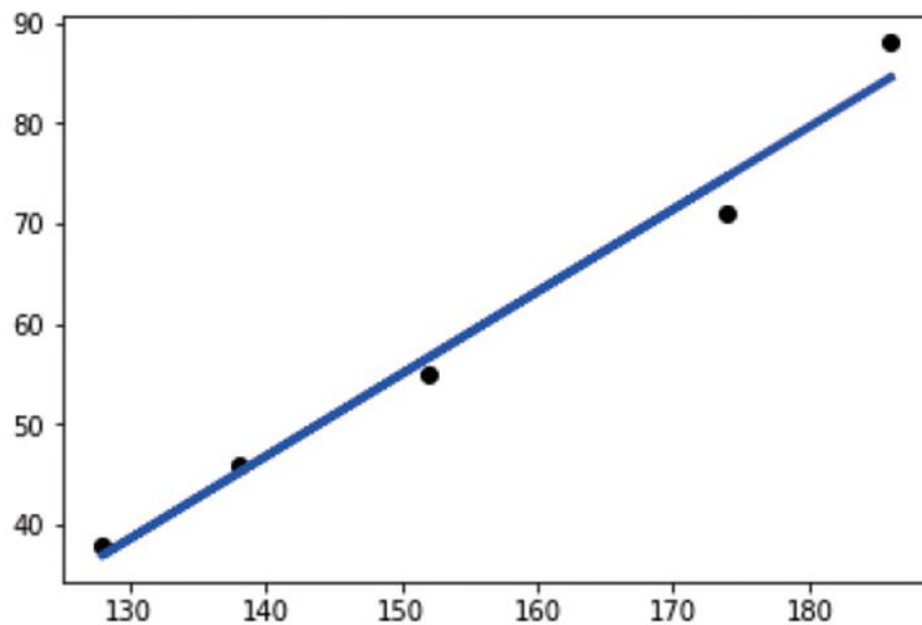
```
# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X)
```

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


선형 회귀 예제

인하공전 컴퓨터 정보과

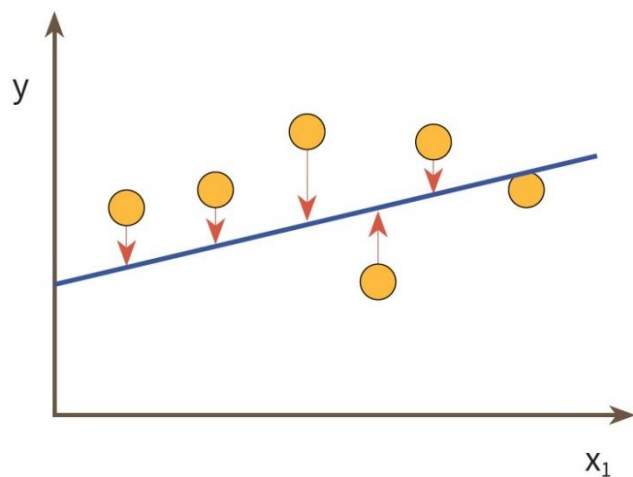
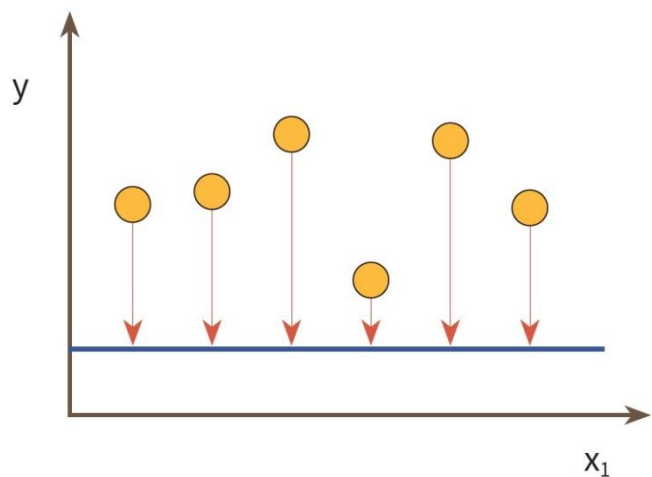
[67.30998637]



선형 회귀 예제

인하공전 컴퓨터 정보과

- 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

선형 회귀 예제 실습

인하공전 컴퓨터 정보과

다음은 집의 면적 당 가격을 정리한 표이다.

X 입력	면적	5	7	12	13	19
Y 출력	가격	12	19	28	37	48

자료실의 `regression1.ipnb` 프로그램을 이용하여 (학습률은(rate) 0.001로 변경하시오)

- 1) 선형회귀로 분석 하여 직선의 방정식 $y=wx+b$ 를 구하여라
- 2) 면적이 10일때의 가격을 예측해보자.

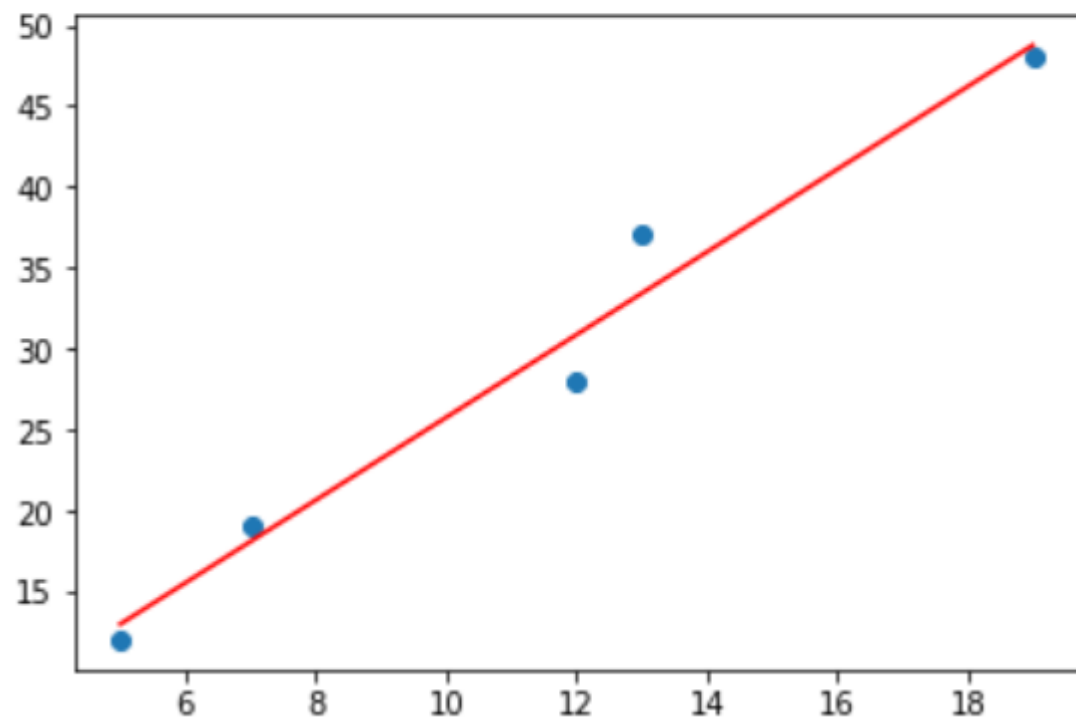
과제 1: code : regression1.ipnb
w,b 와
면적이 10일때의 가격의 예측값은?

선형 회귀 예제 실습

인하공전 컴퓨터 정보과

2.556392890499896 0.17527750947531007

25.73920641447427



다음은 CPU 속도와 프로그래밍 수행 시간을 정리한 표이다.

CPU	30	50	80	90	120
수행 시간	70	140	145	170	260

자료실의 **regression2.ipnb** 프로그램을 이용하여

- 1) 선형회귀로 분석 하여 직선의 방정식 $y=wx+b$ 를 구하여라
- 1) CPU 속도가 100일때의 수행 시간을 예측 하시오.

```
# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
x = [[30], [50], [80], [90], [120]] # 반드시 2차원으로 만들어야 함
y = [70, 140, 145, 170, 260]
```

```
print(reg.predict([[100]]))
```

과제 2: code : regression2.ipnb

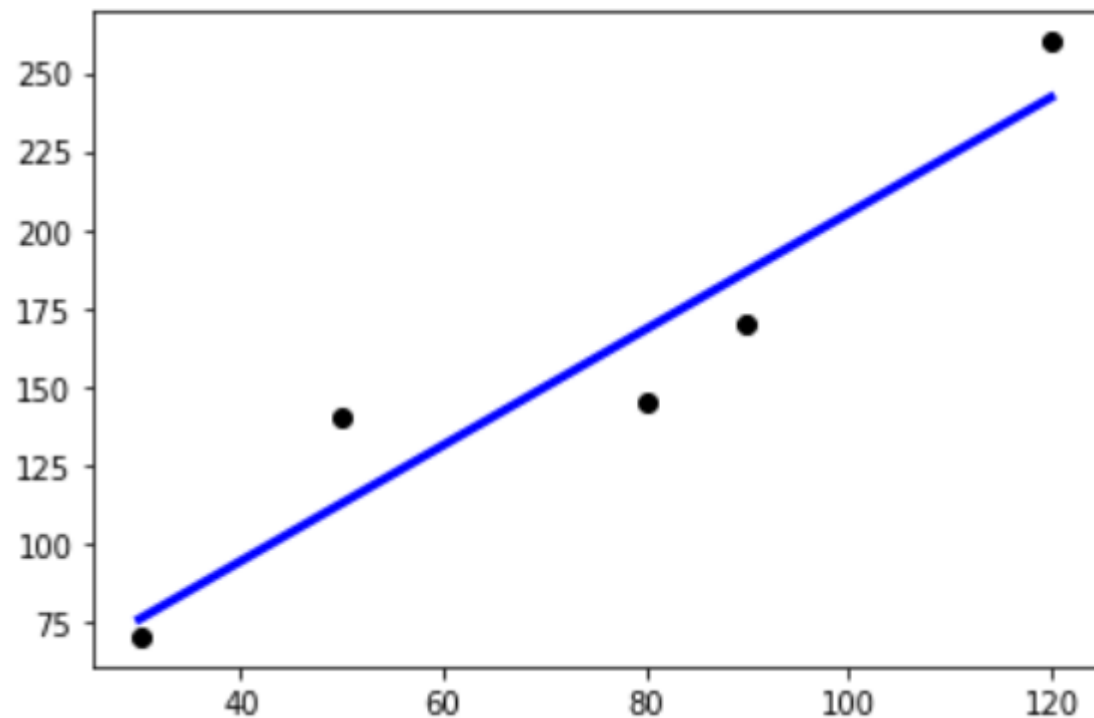
w,b 와

w,b, CPU 속도가 100일때의 수행시간 예측 값은?

선형회귀 예제 실습

인하공전 컴퓨터 정보과

[1.85162602]
19.97967479674793
0.8982062304648606
[205.14227642]



다음은 어떤 회사의 연도별 매출 실적을 정리한 표이다.

X(연도)	2015	2016	2017	2018	2019
Y(억원)	12	19	28	37	46

과제 3:

자료실의 `regression1_loss_noprint.ipynb` 프로그램을 이용하여

- 1) 선형회귀로 분석 하여 직선의 방정식 $y=wx+b$ 를 구하여라
- 1) 2020년의 매출을 예측 해보자

```
lr_rate = 0.0001 # 학습률
```

```
epochs = 1000 # 반복 횟수
```

다중 선형 회귀

인하공전 컴퓨터 정보과

공부한시간	2	4	6	7
과외 수업 횟수	0	4	2	3
성적	81	93	92	97

입력: X 1) 공부한시간, 2) 과외 수업횟수
출력 : y 성적

```
import matplotlib.pyplot as plt
from sklearn import linear_model
import numpy as np
```

```
regr = linear_model.LinearRegression()
```

```
X = np.array([[2, 0], [4, 4], [6, 2], [8, 3] ]) # 2차원 입력 데이터
```

```
y = np.array([81,93,92,97])
regr.fit(X, y) # 학습시키기
```

```
print('계수 :', regr.coef_)
print('절편 :', regr.intercept_)
```

```
fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')
```

```
ax.scatter(X[:,0],X[:,1],y)
```

```
plt.show()
```

```
# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = regr.predict([[2,4]]) # 공부시간 2, 과외 횟수 4일때의 점수 예측
```

```
print(y_pred)
```


다중 선형 회귀 예제 실습

인하공전 컴퓨터 정보과

다음은 집값을 정리한 자료이다.

면적	40	55	62	70
초등학교 접근성	6	2	5	7
가격	3.1	4.2	5.3	7.1

과제 4:
자료실의 `regression_multi.ipynb` 프로그램을 이용하여
면적이 50, 초등학교 접근성이 3일때의 가격을 예측 하시오

선형 회귀 예제 실습 - 당뇨병 예제

인하공전 컴퓨터 정보과

특징(10개)

age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
...									

데이터 개수 (442)

혈당
...

bmi

Bmi와 혈당간의 관계 예측

```
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

```
print('diabetes_X',diabetes_X.shape )
```

하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱스가 2이다.

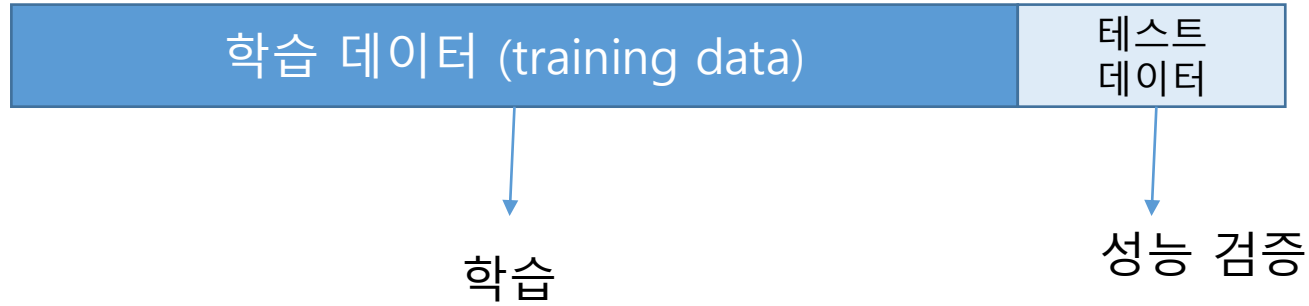
```
diabetes_X_new0 = diabetes_X[:, 2]
```

```
print('diabetes_X_new0',diabetes_X_new0.shape )
```

```
diabetes_X_new = diabetes_X_new0[:, np.newaxis]
```

```
print('diabetes_X_new',diabetes_X_new.shape )
```

```
diabetes_X (442, 10)
diabetes_X_new0 (442,)
diabetes_X_new (442, 1)
```



442개

```
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)
print('X_train',X_train.shape )
print('X_test',X_test.shape )
print('Y_train',y_train.shape )
print('Y_test',y_test.shape )
```

```
X_train (397, 1)
X_test (45, 1)
Y_train (397,)
Y_test (45,)
```

선형 회귀 예제 실습 - 당뇨병 예제

인하공전 컴퓨터 정보과

```
import matplotlib.pyplot 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,diabetes_X.shape )
# 하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱스가 2이다.
diabetes_X_new0 = diabetes_X[:, 2]
print('diabetes_X_new0,diabetes_X_new0.shape )
diabetes_X_new = diabetes_X[:, np.newaxis, 2]
print('diabetes_X_new,diabetes_X_new.shape )

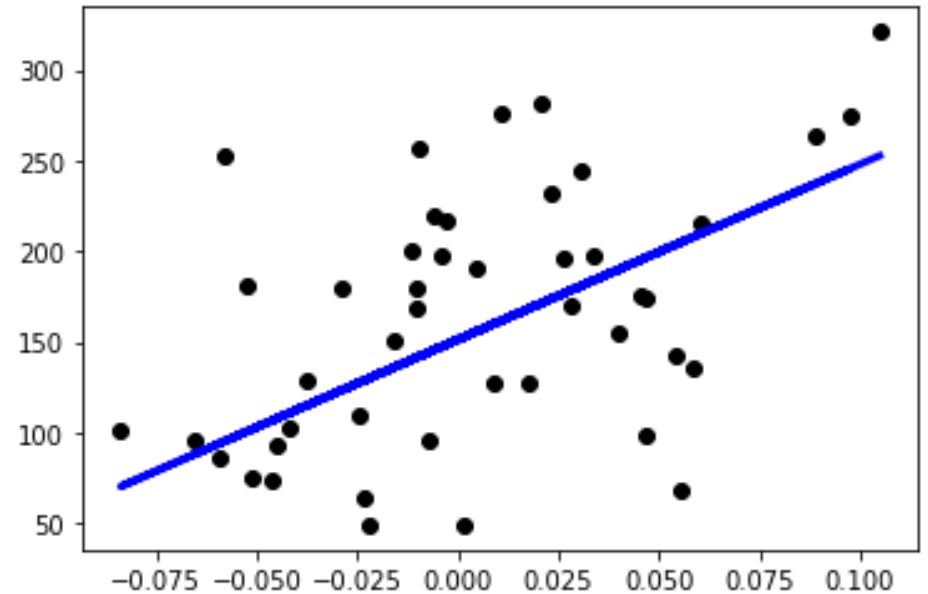
# 학습 데이터와 테스트 데이터를 분리한다.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes_X_new, diabetes_y, test_size=0.1,
random_state=0)
print('X_train,X_train.shape)
print('X_test,X_test.shape)
print('y_train,y_train.shape)
print('y_test,y_test.shape)

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

# 테스트 데이터로 예측해보자.
y_pred = regr.predict(X_test)
print(regr.predict([[0.01]])) # bmi가 0.01일때 혈당 예측값

# 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()
```



- 수치 계산을 위한 라이브러리
- `Import numpy as np`

딥러닝에서 넘파이가 중요한 이유

- 학습 데이터는 2차원 행렬이나 3차원 행렬에 저장된다.

```
>>> import numpy as np

>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])

>>> a[0]
```

1

```
>>> b = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> b
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

>>> b[0][2]
3
```

b[1,2] ?

b[2,0] ?

Numpy

```
>>> import numpy as np
>>> a = np.array([[ 0, 1, 2],
                  [ 3, 4, 5],
                  [ 6, 7, 8]])
>>> a.shape           # 배열의 형상
(3, 3)
>>> a.ndim            # 배열의 차원 개수
2
>>> a.dtype           # 요소의 자료형
dtype('int32')
>>> a.itemsize        # 요소 한개의 크기
4                      # 오타
>>> a.size            # 전체 요소의 개수
9
```

```
>>> import numpy as np
>>> a = np.array([[ 0, 1, 2],
                  [ 3, 4, 5],
                  [ 6, 7, 8]])
```

```
>>> np.zeros( (3, 4) )           # (3, 4)는 배열의 형상(행의 개수, 열의 개수)
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])

>>> np.ones((3, 4))
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])

>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
>>> import numpy as np
>>> a = np.zeros( (3, 4) )
      a.shape ?
      a.size?
```

Numpy-arange

```
>>> np.arange(5)
array([0, 1, 2, 3, 4])

>>> np.arange(1, 6)
array([1, 2, 3, 4, 5])

>>> np.arange(1, 10, 2)
array([1, 3, 5, 7, 9])
```

>>> np.arange(1, 11, 2) ?

Numpy-배열 합치기

인하공전 컴퓨터 정보과

```
>>> x = np.array([[1, 2], [3, 4]])  
>>> y = np.array([[5, 6], [7, 8]])  
  
>>> np.concatenate((x, y), axis=1)  
array([[1, 2, 5, 6],  
       [3, 4, 7, 8]])
```

```
>>> np.vstack((x, y))  
array([[1, 2],  
       [3, 4],  
       [5, 6],  
       [7, 8]])
```

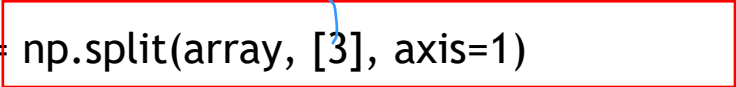
```
>>> a = np.arange(12)
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

# a에 대하여 reshape(3, 4)를 호출하면 1차원 배열이 2차원 배열로 바뀌게 된다.
>>> a.reshape(3, 4)
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

>>> a.reshape(6, -1)
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

```
>>> array = np.arange(30).reshape(-1, 10)
>>> array
Array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])

>>> arr1, arr2 = np.split(array, [3], axis=1)
>>> arr1
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22]])
>>> arr2
array([[ 3,  4,  5,  6,  7,  8,  9],
       [13, 14, 15, 16, 17, 18, 19],
       [23, 24, 25, 26, 27, 28, 29]])
```



```
>>> a = np.array([1, 2, 3, 4, 5, 6])
>>> a.shape
(6,)

>>> a1 = a[np.newaxis, :]
>>> a1
array([[1, 2, 3, 4, 5, 6]])
>>> a1.shape
(1, 6)

>>> a2 = a[:, np.newaxis]
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6]])
>>> a2.shape
(6, 1)
```

```
>>> ages = np.array([18, 19, 25, 30, 28])
>>> ages[1:3] # 인덱스 1에서 인덱스 2까지
array([19, 25])
>>> ages[:2] # 인덱스 0에서 인덱스 1까지
array([18, 19])
```

논리적인 인덱싱(logical indexing)

```
>>> y = ages > 20
>>> y
array([False, False,  True,  True,  True])
>>> ages[ ages > 20 ]
array([25, 30, 28])
```


Numpy-2차원 배열의 인덱싱

인하공전 컴퓨터 정보과

```
>>> a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> a[0, 2]
3
>>> a[0, 0] = 12
>>> a
array([[12, 2, 3],
       [ 4, 5, 6],
       [ 7, 8, 9]])
```

Numpy-2차원 배열의 슬라이싱

인하공전 컴퓨터 정보과

```
>>> a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
>>> a[0:2, 1:3]  
array([[2, 3],  
       [5, 6]])
```

```
a[::2, ::2]
```

```
>>> arr1 = np.array([[1, 2], [3, 4], [5, 6]])
>>> arr2 = np.array([[1, 1], [1, 1], [1, 1]])
>>> result = arr1 + arr2          # 넘파이 배열에 + 연산이 적용된다.
>>> result
array([[2, 3],
       [4, 5],
       [6, 7]])
```

```
>>> A = np.array([0, 1, 2, 3])  
>>> 10 * np.sin(A)  
array([0.          , 8.41470985, 9.09297427, 1.41120008])
```

Numpy-특정한 행과 열을 이용한 연산

인하공전 컴퓨터 정보과

```
>>> scores = np.array([[99, 93, 60], [98, 82, 93],  
...:                   [93, 65, 81], [78, 82, 81]])  
>>> scores.mean(axis=0)  
array([92. , 80.5 , 78.75])
```

```
>>> np.random.seed(100)
>>> np.random.rand(5)
array([0.54340494, 0.27836939, 0.42451759, 0.84477613, 0.00471886])

>>> np.random.rand(5, 3)
array([[0.12156912, 0.67074908, 0.82585276],
       [0.13670659, 0.57509333, 0.89132195],
       [0.20920212, 0.18532822, 0.10837689],
       [0.21969749, 0.97862378, 0.81168315],
       [0.17194101, 0.81622475, 0.27407375]])
```

```
>>> np.random.randn(5)
array([ 0.78148842, -0.65438103,  0.04117247, -0.20191691, -0.87081315])

>>> np.random.randn(5, 4)
array([[ 0.22893207, -0.40803994, -0.10392514,  1.56717879],
       [ 0.49702472,  1.15587233,  1.83861168,  1.53572662],
       [ 0.25499773, -0.84415725, -0.98294346, -0.30609783],
       [ 0.83850061, -1.69084816,  1.15117366, -1.02933685],
       [-0.51099219, -2.36027053,  0.10359513,  1.73881773]])

>>> m, sigma = 10, 2
>>> m + sigma*np.random.randn(5)
array([ 8.56778091, 10.84543531,  9.77559704,  9.09052469,  9.48651379])
```



```
arr=np.array([[1,2],[3,4],[5,6]])  
print(arr.T)
```

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

수고하셨습니다

jhmin@inhatec.ac.kr