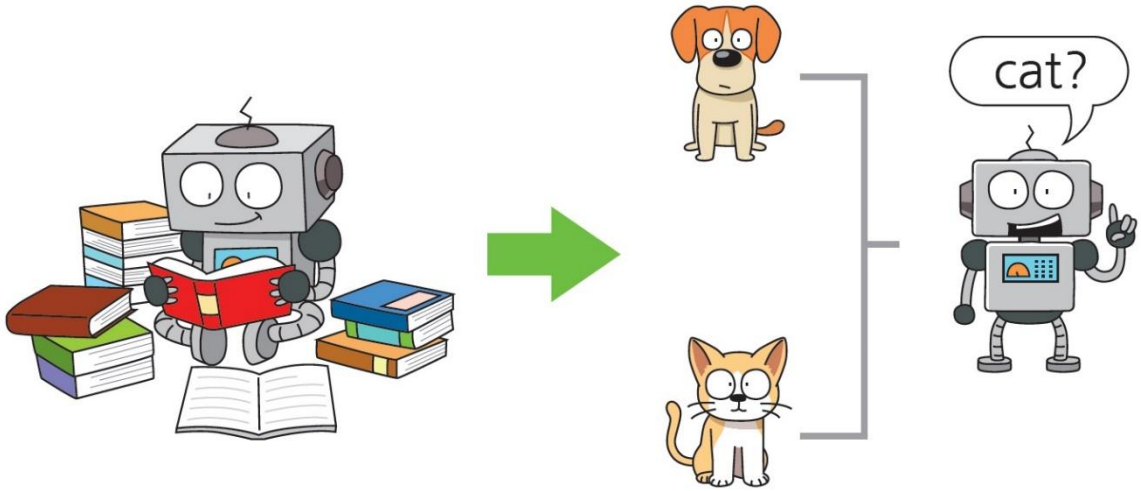
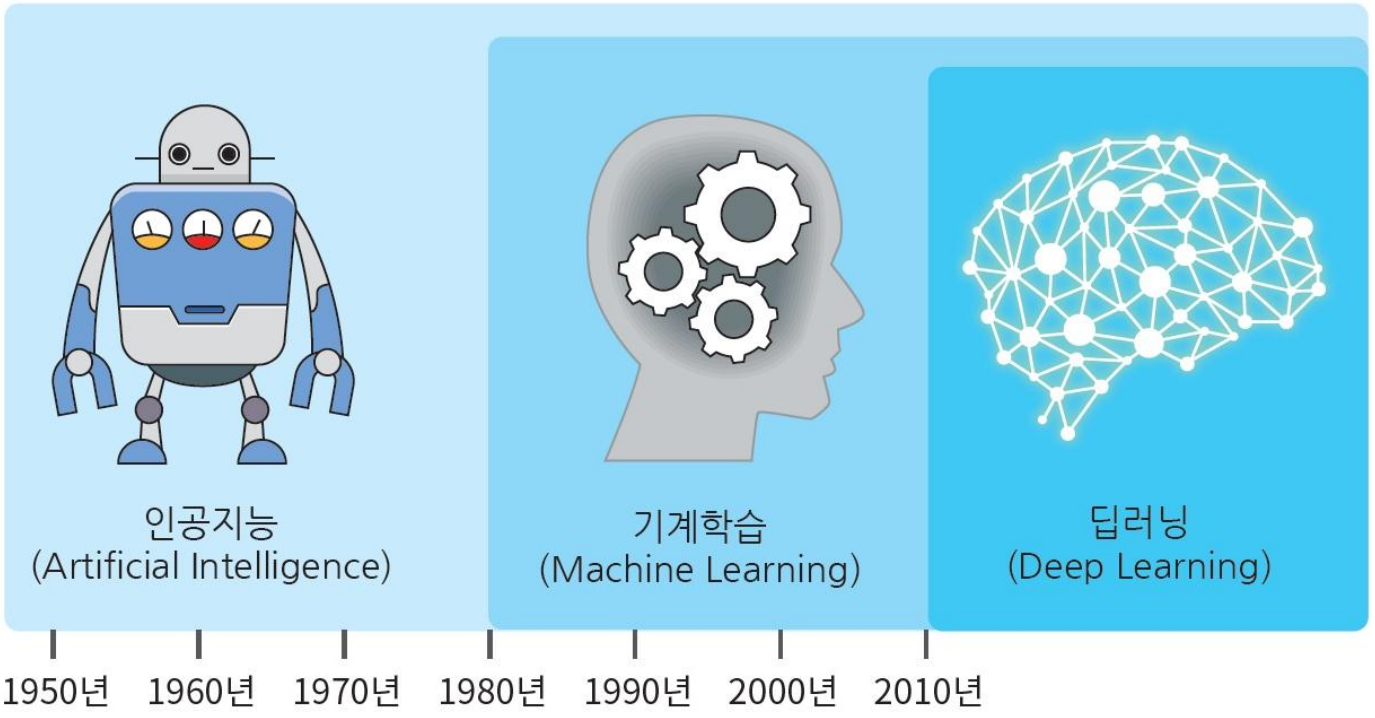
기계학습

기계 학습이란?

- 현재의 컴퓨터는 스스로 학습할 수 없기 때문에 우리가 컴퓨터에게 어떤 작업을 시키려면 반드시 프로그램을 작성하여 작업을 지시하여야 한다.
- 컴퓨터가 스스로 학습할 수 있다면 컴퓨터는 프로그램 없이도 여러 가지 일을 할 수 있을 것이다.



인공지능, 기계 학습, 딥러닝

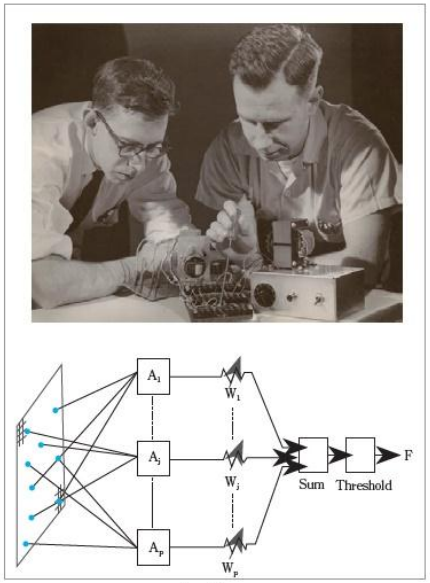


기계 학습은 어디에 이용되는가?

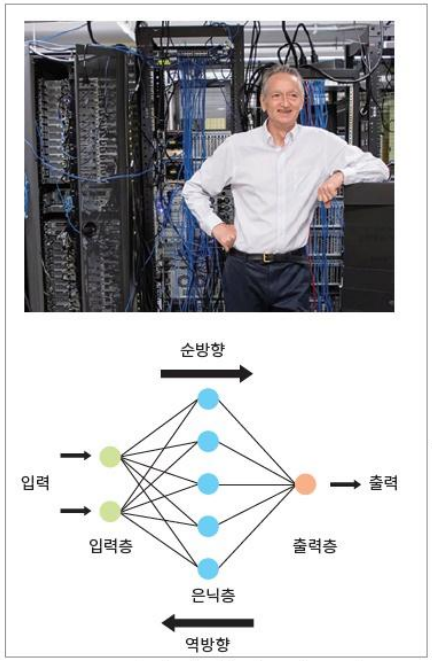
기계 학습은 어디에 이용되는가?

- 이들 분야들은 살펴보면 복잡한 데이터들이 있고, 이들 데이터에 기반하여 결정을 내려야 하는 분야이다.
  - 영상 인식, 음성 인식처럼 프로그램으로 작성하기에는 규칙과 공식이 너무 복잡할 때
  - 주식 거래나 에너지 수요 예측, 쇼핑 추세 예측의 경우처럼 데이터 특징이 계속 바뀌고 프로그램을 계속해서 변경해야 하는 상황일 때
  - 구매자가 클릭할 확률이 가장 높은 광고가 무엇인지를 알아내는 시스템
  - 넷플릭스에서 비디오 추천 시스템
  - 자율 주행자동차

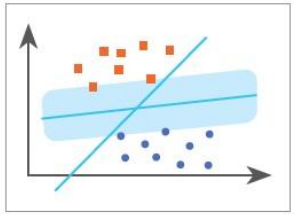
기계 학습의 역사



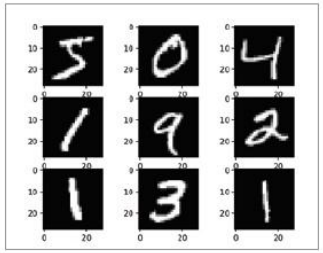
1957 퍼셉트론



1985 역전파 학습 알고리즘



1995 SVM



1998 필기체 인식

2012 ImageNet에서 우승

그림 9-4 기계학습의 역사

기계 학습의 종류

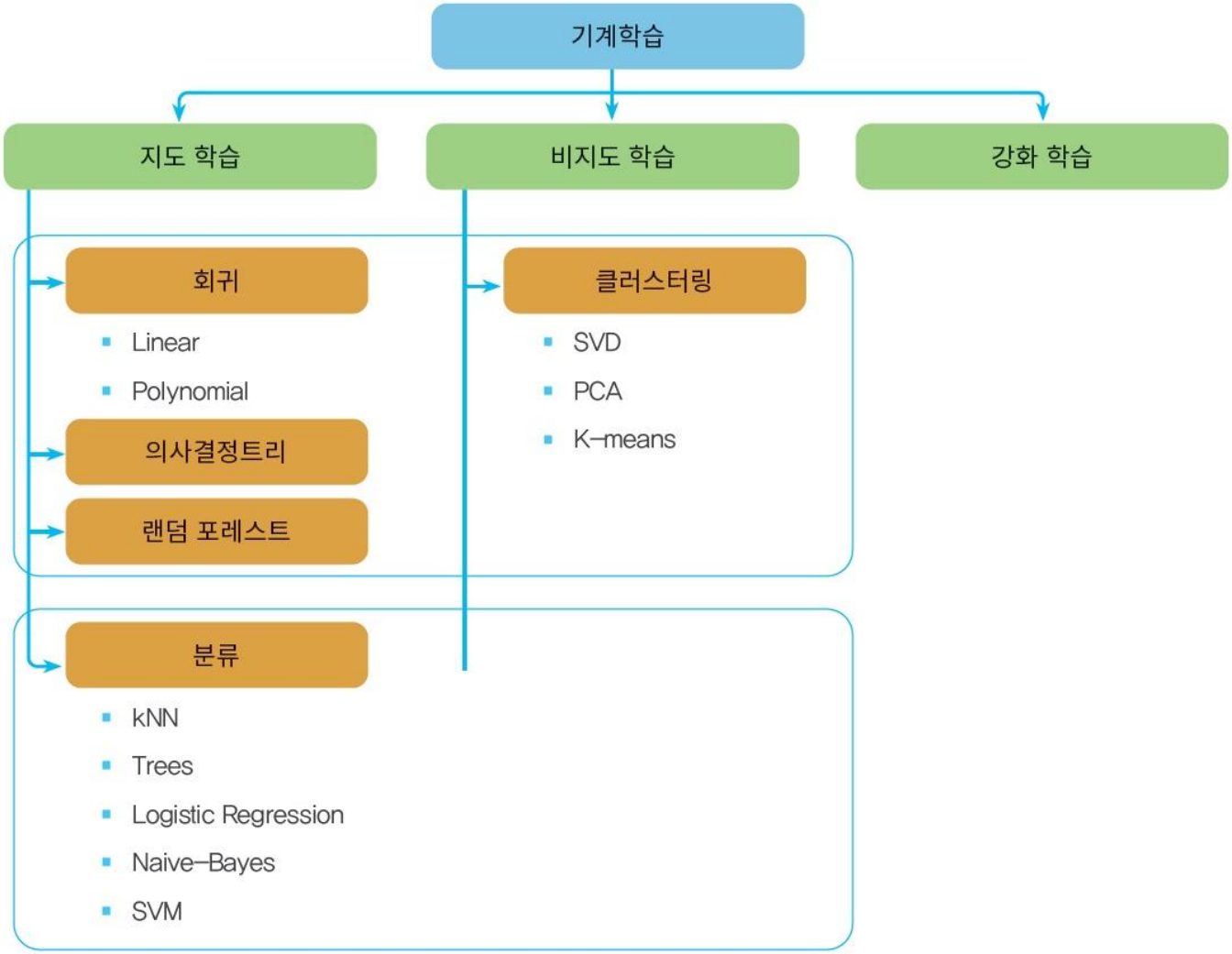
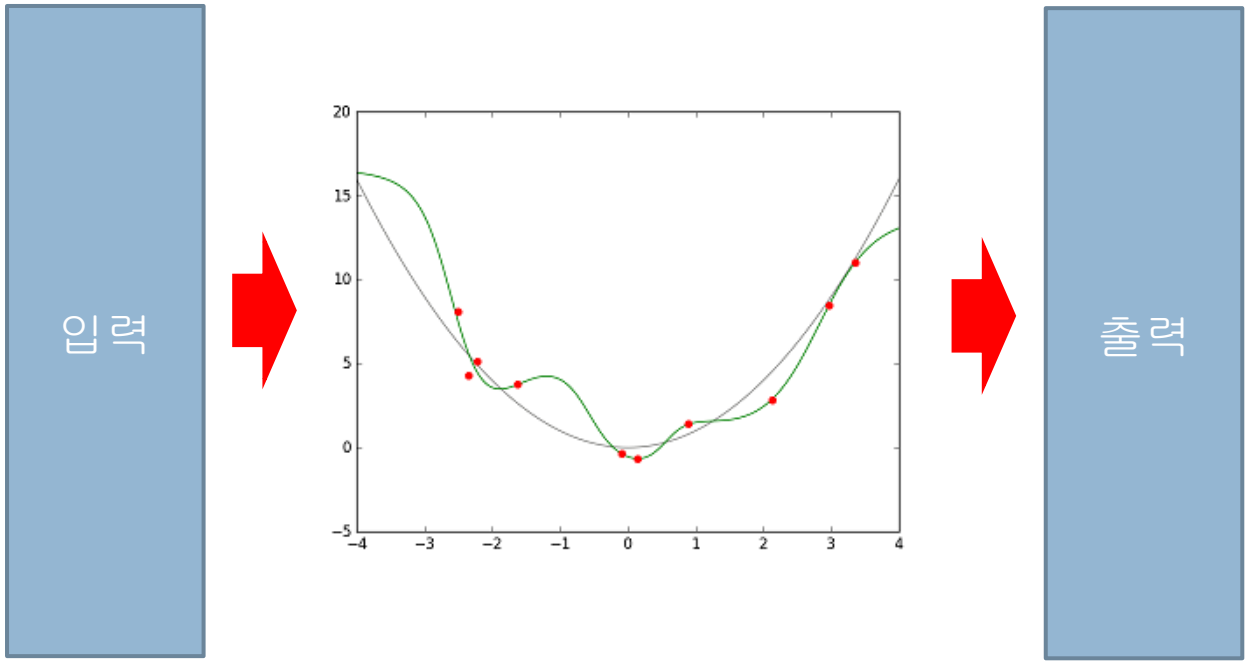


그림 9-5 기계학습의 종류

기계 학습

- 기계 학습은 항상 입력을 받아서 출력하는 함수 y=f(x)를 학습한다고 생각할 수 있다. (함수 근사)



기계 학습(machine learning) == 함수 근사(function approximation)

특징(features)

- 특징이란 우리가 학습 모델에게 공급하는 입력이다. 가장 간단한 경우에는 입력 자체가 특징이 된다.

- (예)
  - 이메일에 “검찰”이라는 문자 포함 여부(yes 또는 no)
  - 이메일에 “광고”, “선물 교환권”이나 “이벤트 당첨” 문자열 포함 여부(yes 또는 no)
  - 이메일의 제목이나 본문에 있는 ‘★’과 같은 특수 기호의 개수(정수)
  - …

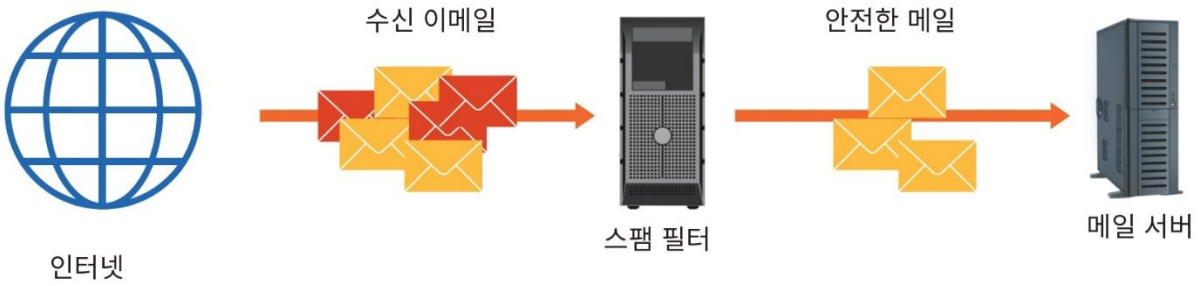


그림 9–6 스팸 필터링 시스템

레이블과 샘플

- 레이블(label)
  - y = f(X)에서 y 변수에 해당한다.
  - 예를 들어서 농작물의 향후 가격, 사진에 표시되는 동물의 종류, 동영상의 의미 등 무엇이든지 레이블이 될 수 있다.

- 샘플, 또는 예제
  - 샘플은 기계 학습에 주어지는 특정한 예이다. y = f(X)에서 X에 해당한다. 레이블이 있는 샘플도 있고 레이블이 없는 샘플도 있다. 지도 학습을 시키려면 레이블이 있어야 한다.

학습 데이터와 테스트 데이터

그림 9-7 학습 단계

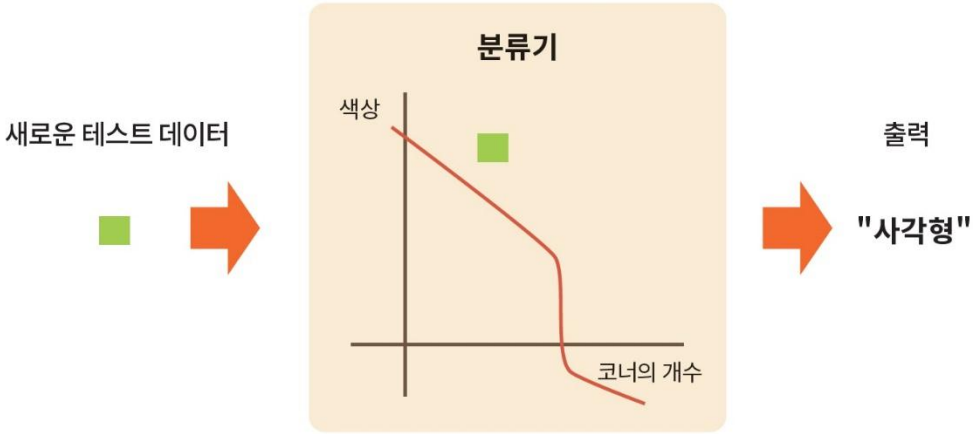


그림 9-8 테스트 단계

학습과 예측

- 학습(learning)은 모델을 만들거나 배우는 것을 의미한다.

- 예측(prediction)은 학습된 모델을 레이블이 없는 샘플에 적용하는 것을 의미한다. 즉 학습된 모델을 사용하여 유용한 예측(y')을 해내는 것이다.

지도 학습

- 회귀(regression) : 회귀에서는 입력과 출력이 모두 실수이다.
  - “사용자가 이 광고를 클릭할 확률이 얼마인가요?”
  -

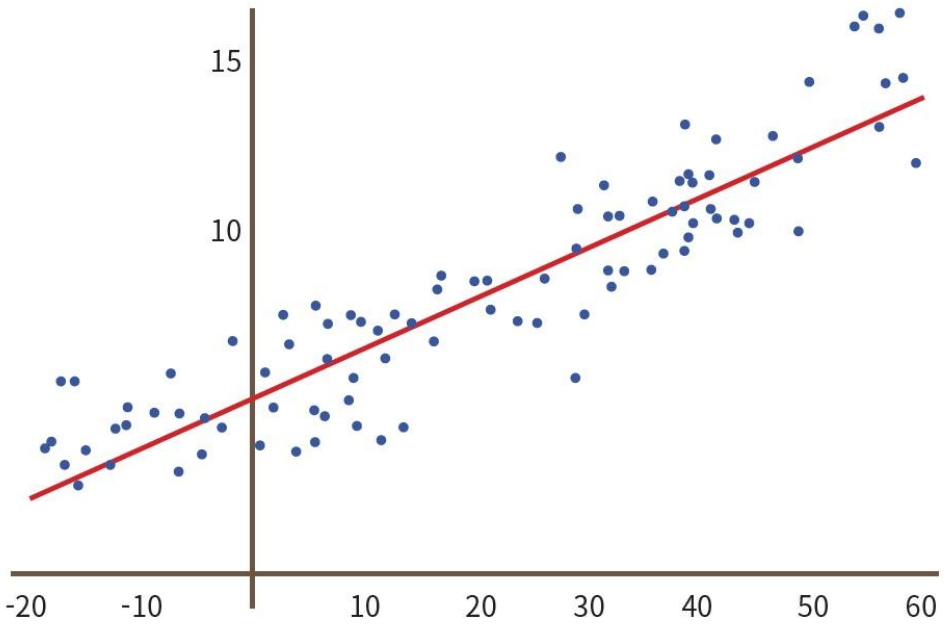
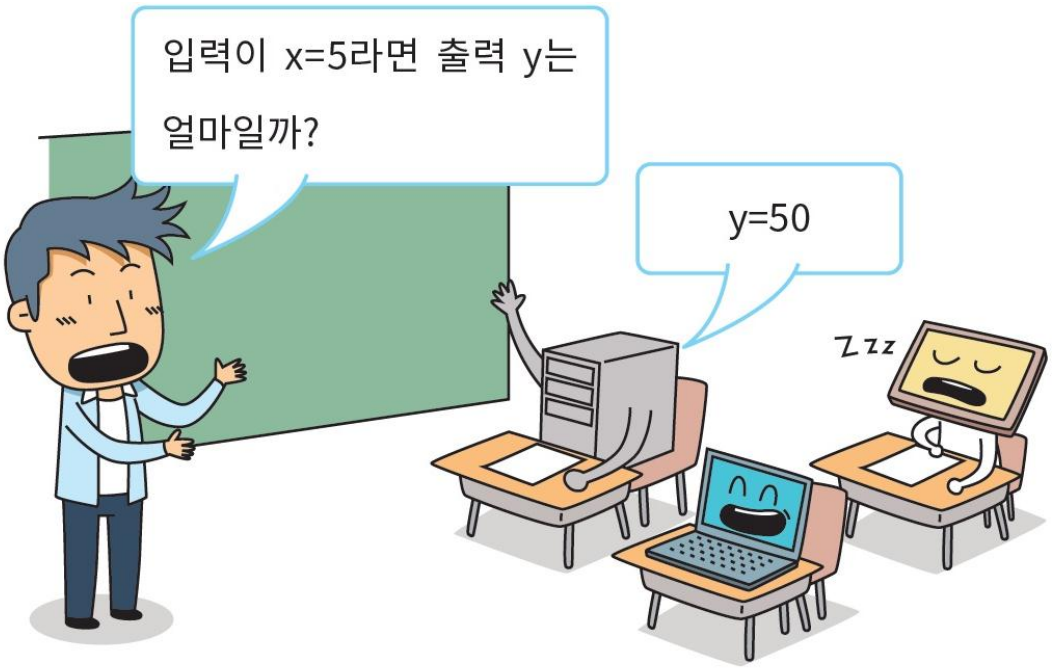


그림 9–10 회귀

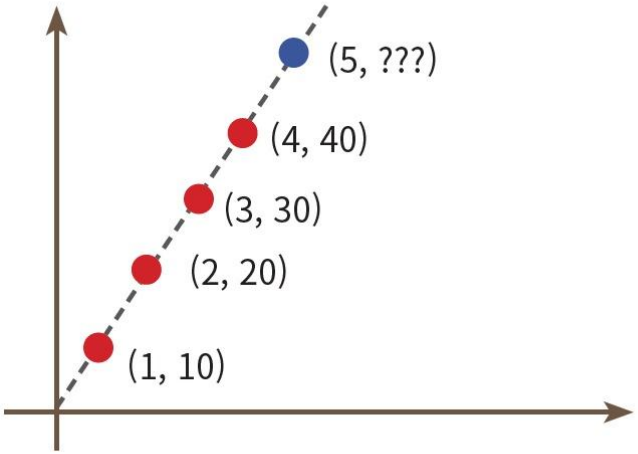
회귀

- 회귀는 실수 입력(x)과 실수 출력(y)이 주어질 때, 입력에서 출력으로의 매핑 함수를 학습하는 것이라 할 수 있다.

$$y = f(x)$$



회귀의 예



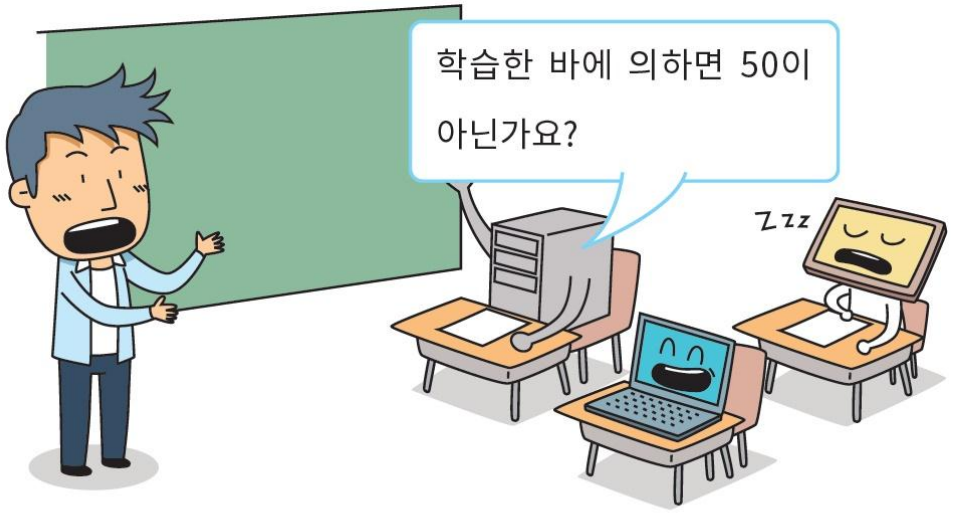


그림 9–13 회귀의 예

지도 학습

- 분류(classification): 입력을 두 개 이상의 레이블(유형)으로 분할하는 것
- 해당 모델을 학습시킬 때 우리는 레이블을 제공해야 한다.

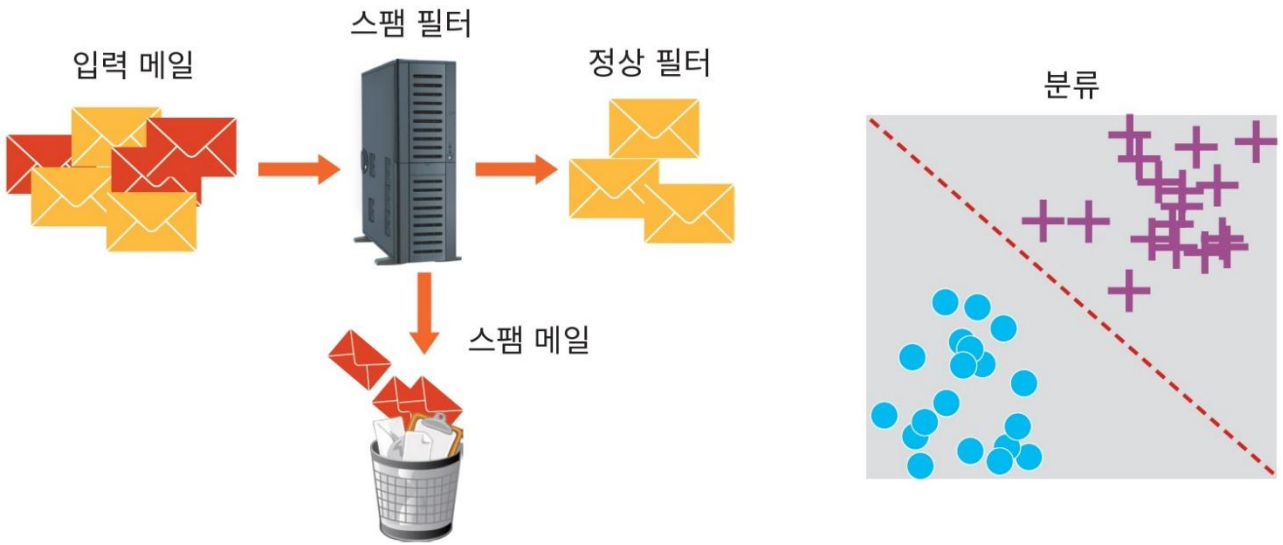
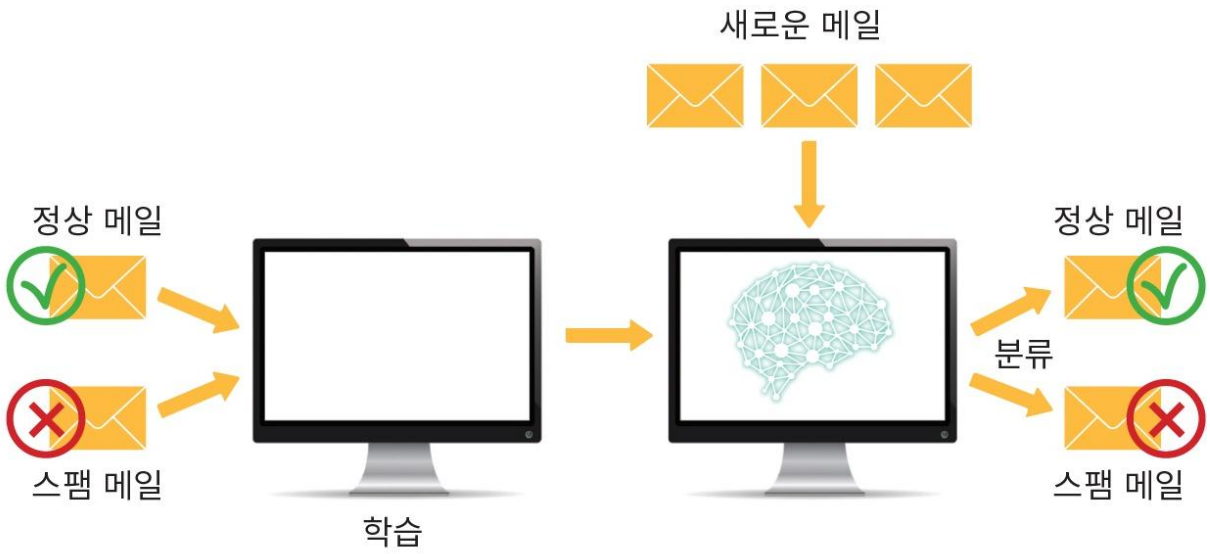


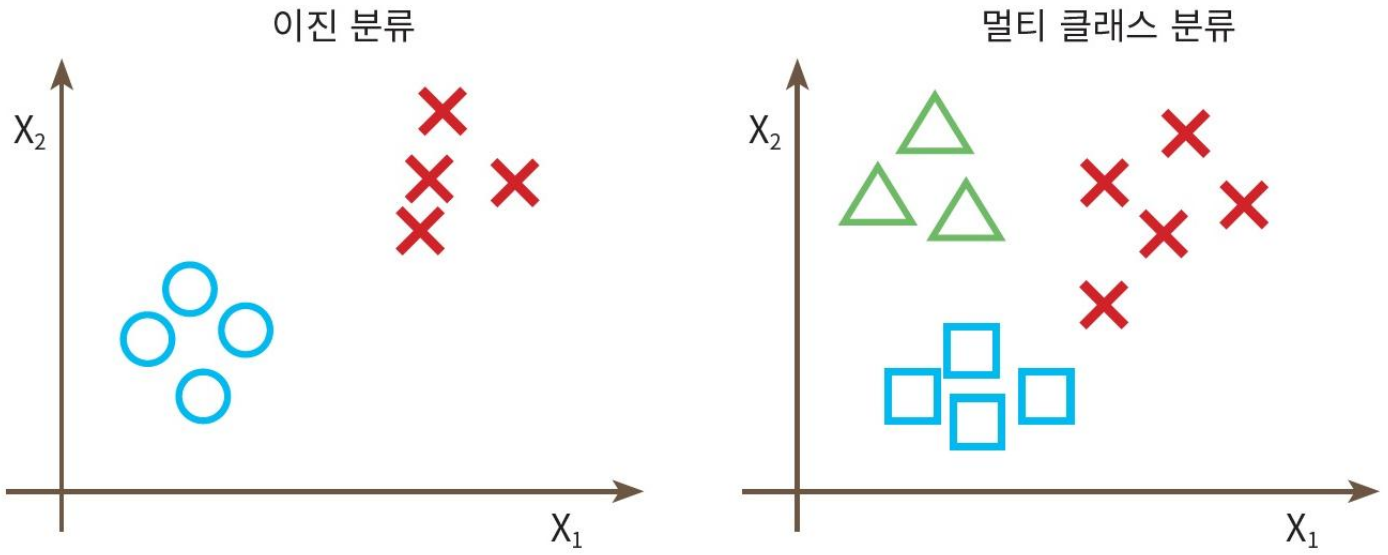
그림 9-11 분류

분류

- 앞에 나왔던 식 y = f(x)에서 출력 y가 이산적(discrete)인 경우에 이것을 분류 문제(또는 인식 문제)라고 부른다.
- 분류에서는 입력을 2개 이상의 클래스로 나누는 것이다.

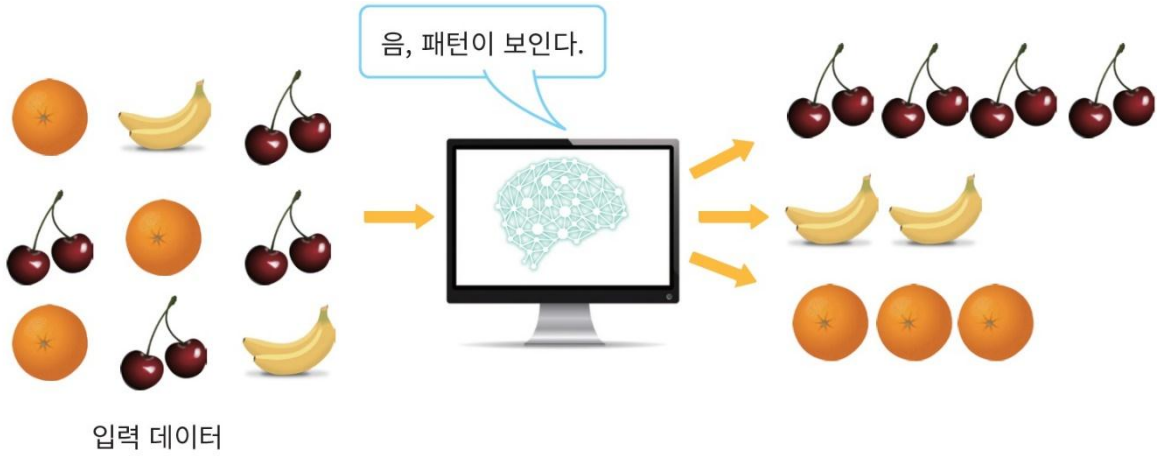


- 분류는 지도 학습의 형태로 이루어지는 것이 일반적이다.
- 분류를 수행하기 위한 일반적인 알고리즘에는 신경망, kNN(k-nearest neighbor), SVM(Support Vector Machine), 의사 결정 트리 등이 포함된다.



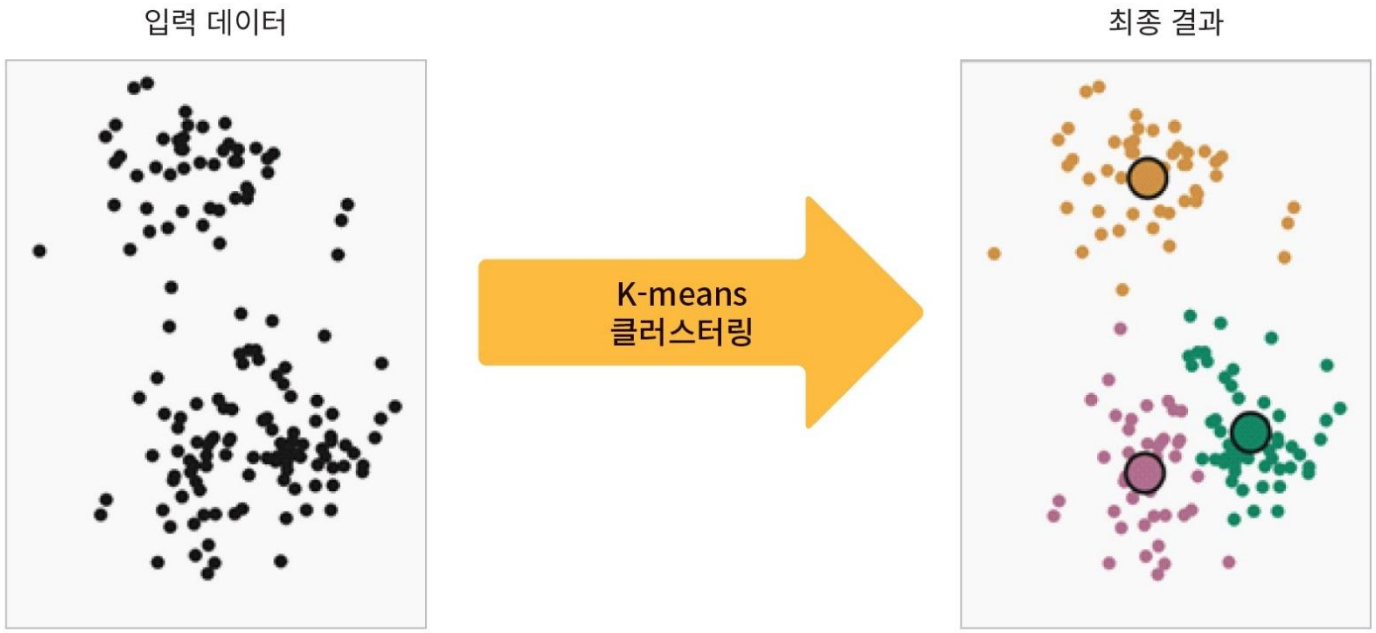
비지도 학습

- 비지도 학습(unsupervised Learning)은 "교사" 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다. 식 y = f(x)에서 레이블 y가 주어지지 않는 것이다.
- 데이터들의 상관도를 분석하여 유사한 데이터들을 모을 수는 있다.



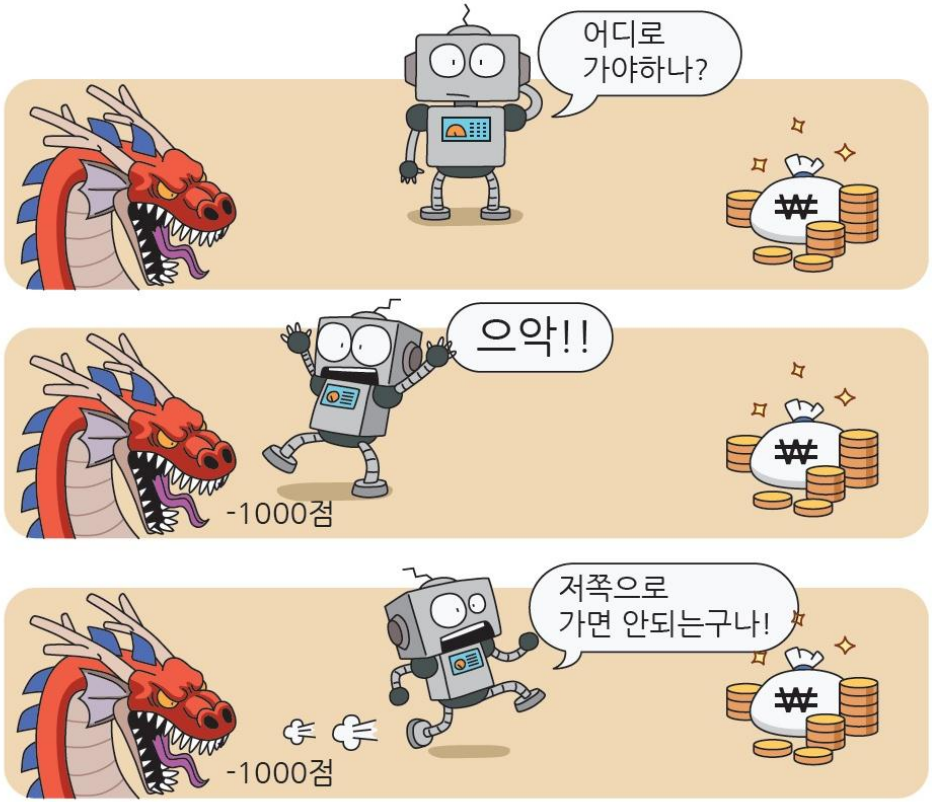
비지도 학습

- 가장 대표적인 비지도 학습이 클러스터링(군집화, clustering)이다.
- 클러스터링이란 데이터간 거리를 계산하여서 입력을 몇 개의 그룹으로 나누는 방법이다.



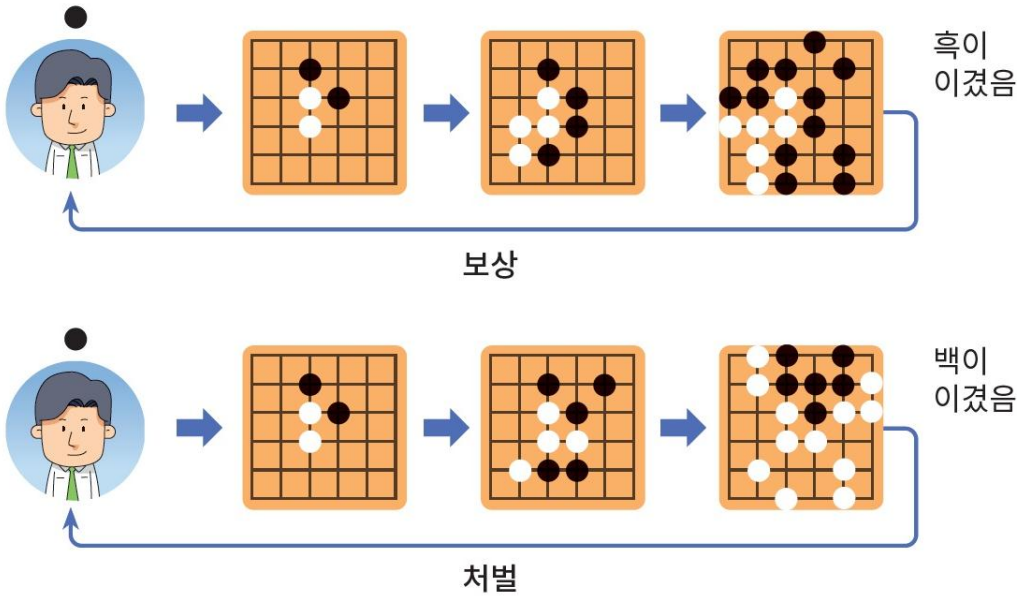
강화 학습

- 강화 학습(Reinforcement Learning)에서는 컴퓨터가 어떤 행동을 취할 때마다 외부에서 처벌이나 보상이 주어진다.



강화 학습

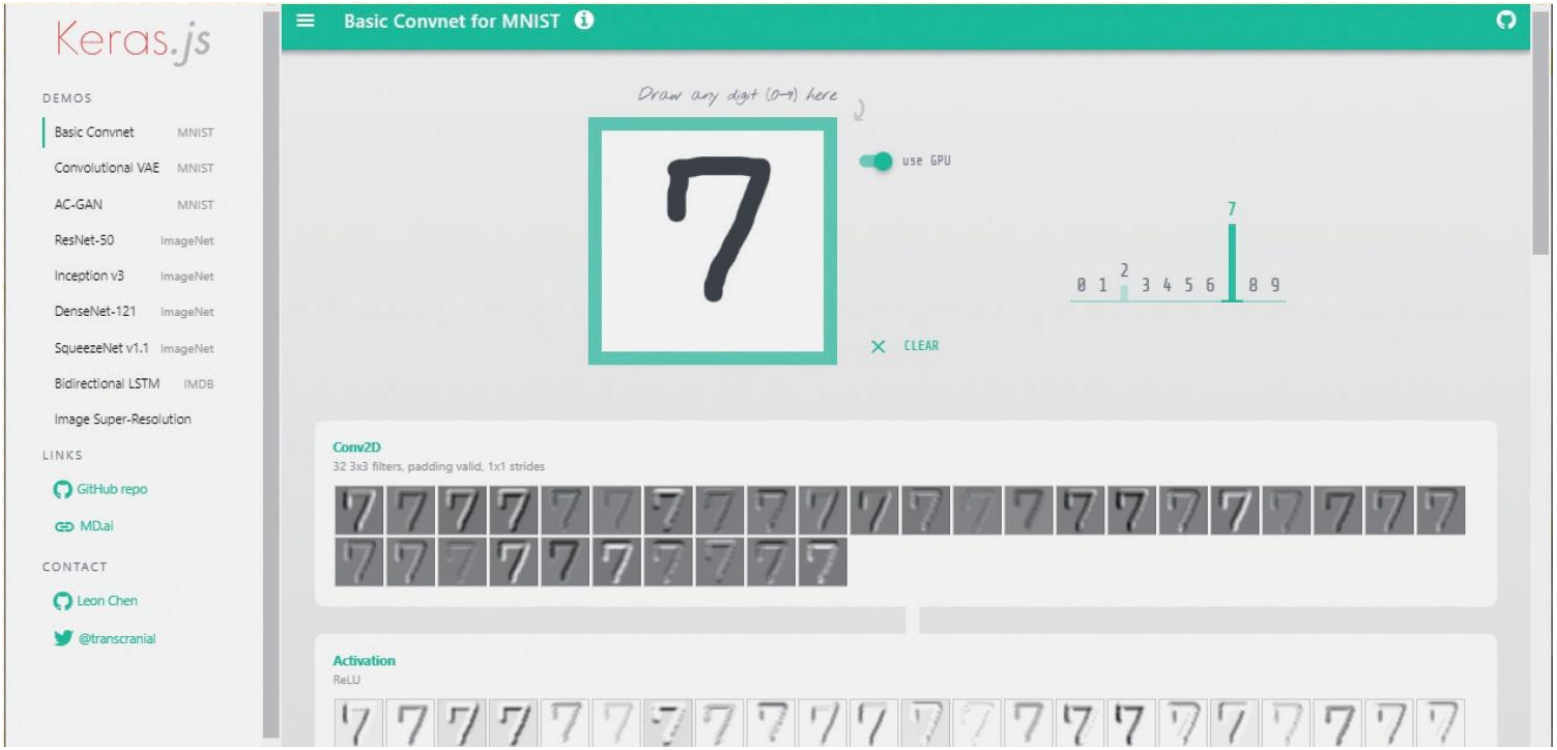
- 알파고 최종 버전도 강화 학습 사용
- 게임에서 많이 사용된다(예: Frozen Lake).



Lab: 기계 학습 체험하기

INFINITYBOOKS
인피니티북스

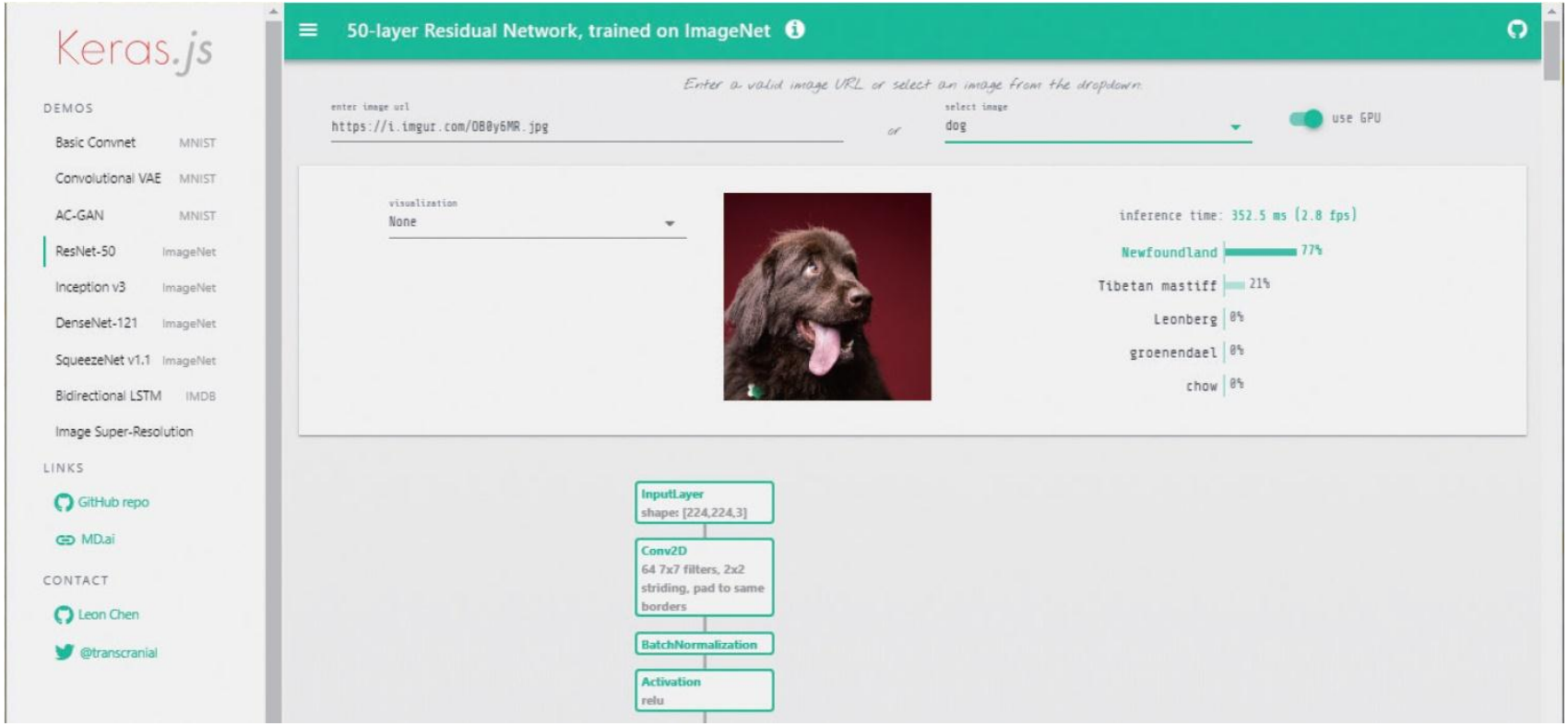
- https://transcranial.github.io/keras-js/#/



Lab: 기계 학습 체험하기

INFINITYBOOKS
인피니티북스

- https://transcranial.github.io/keras-js/#/

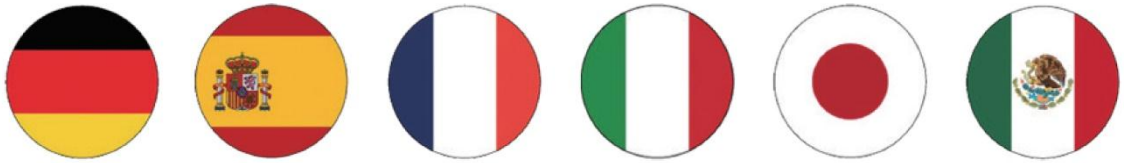


프로그래머로서 기계 학습의 실용적인 가치

- 첫 번째는 프로그래밍 시간을 줄일 수 있다는 점이다.
- 예를 들어서 맞춤법 오류를 수정하는 프로그램을 개발한다고 하자.
  - 전통적인 방법: 많은 맞춤법 규칙을 이용하여 작성할 수 있다. -> 상당한 시간이 필요
  - 기계 학습 이용: 많은 예제만 있다면 학습시켜서 빠른 시간 안에 신뢰성있는 프로그램을 완성할 수 있다.

프로그래머로서 기계 학습의 실용적인 가치

- 두 번째로 맞춤형 제품을 쉽게 개발할 수 있다.
- 예를 들어서 여러분이 한국어 맞춤법 수정 프로그램이 작성하여 가지고 있다고 하자. 제품이 성공적이어서 30개국 언어 버전으로 확장하려고 한다.
  - 전통적인 방법: 각 언어마다 새로 작성하려면 수년 이상의 엄청난 시간이 필요하다.
  - 기계 학습 이용: 너무나도 쉽다 예제만 있으면 된다.

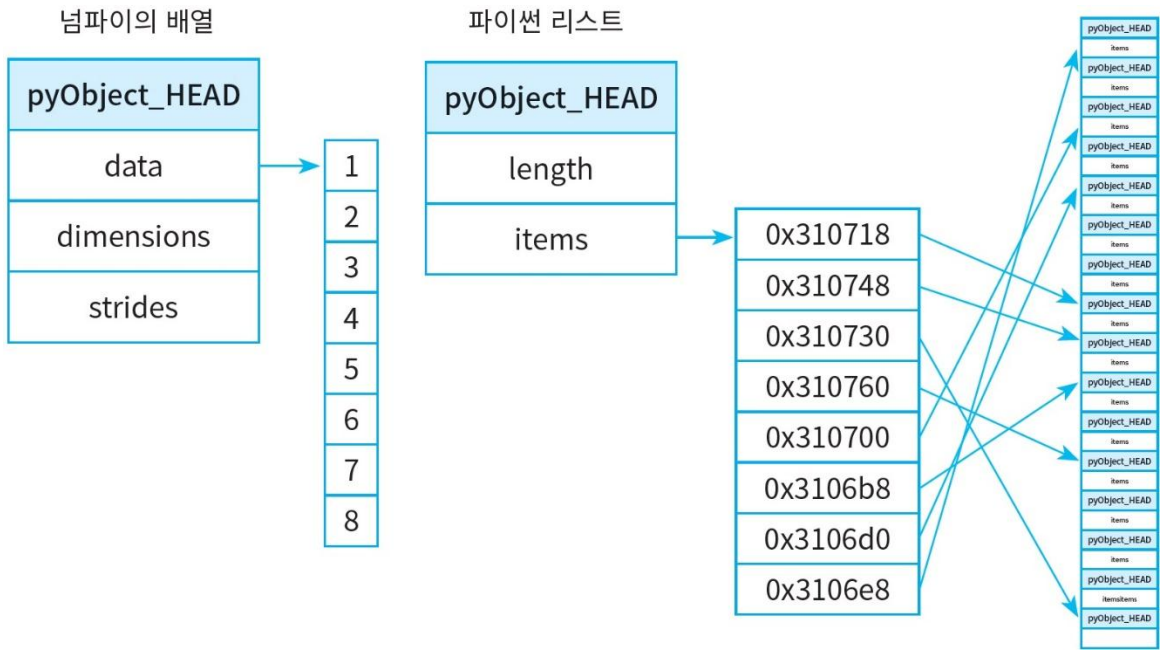


프로그래머로서 기계 학습의 실용적인 가치

- 세 번째로 기계 학습은 프로그래머로 시도할 알고리즘이 떠오르지 않는 문제들을 해결할 수도 있다.
- 예를 들어서 컴퓨터가 사람의 얼굴을 인식하는 프로그램을 작성
  - 전통적인 방법: 이런 문제를 작성하려면 컴퓨터 시각 분야의 수많은 지식과 경험이 필요한 작업이다.
  - 기계 학습 이용: 프로그램에 수많은 예제만 보여주기만 하면 문제가 해결된다. 편리하지 않은가?

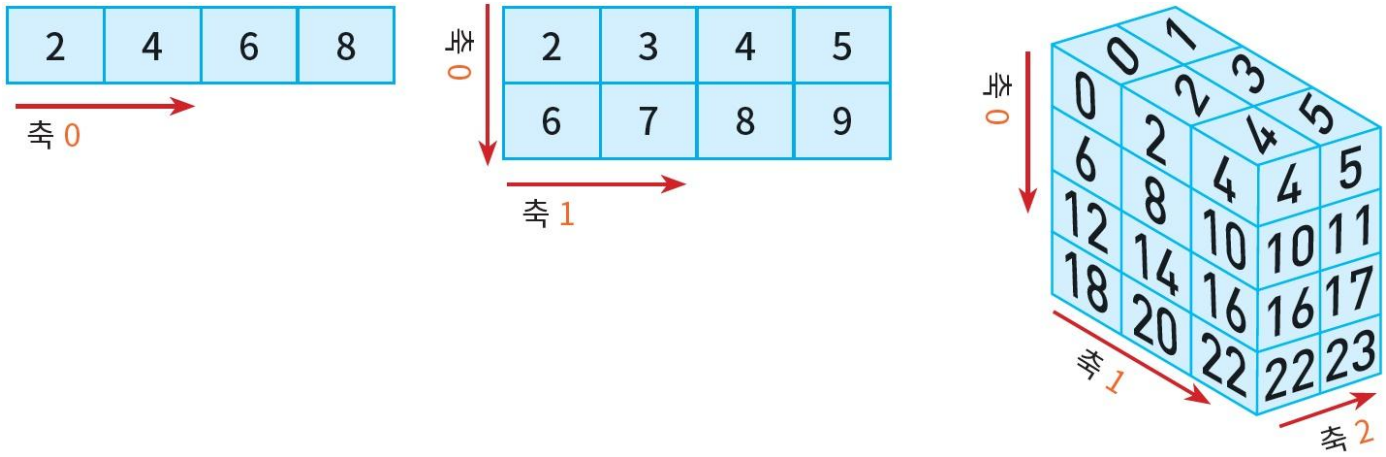
넘파이

- 기계 학습에서는 파이썬의 기본 리스트로 충분하지 않다.
- 데이터를 처리할 때는 리스트와 리스트 간의 연산이 가능해야 하는데 파이썬의 기본 리스트는 이것을 지원하지 않기 때문이다.
- 연산 속도도 중요하기 때문에 데이터 과학자들은 기본 리스트 대신에 넘파이(Numpy)를 선호한다.



넘파이

- 파이썬을 위한 행렬(matrix) 라이브러리



리스트 vs 넘파일 배열


```
mid_scores = [10, 20, 30]
final_scores = [70, 80, 90]
```

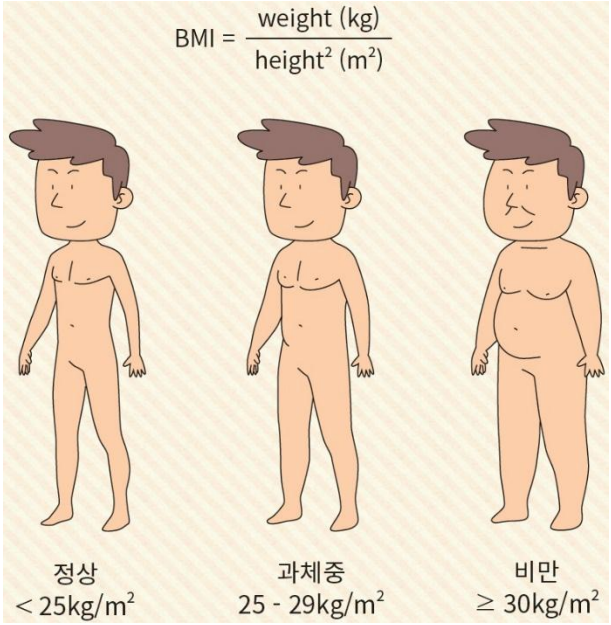
```
>>> total = (mid_scores + final_scores)
>>> total
[10, 20, 30, 70, 80, 90]
```

```
import numpy as np

mid_scores  = np.array([10, 20, 30])
final_scores = np.array([70, 80, 90])

total = mid_scores + final_scores
print(total)
```

```
array([ 80, 100, 120])
```

Lab: BMI 계산하기


```
heights = [ 1.83, 1.76, 1.69, 1. 86, 1.77, 1.73 ]
weights = [ 86,    74,    59,   95,    80,   68  ]
```



Lab: BMI 계산하기


```
import numpy as np

heights = [ 1.83, 1.76, 1.69, 1.86, 1.77, 1.73 ]
weights = [ 86,    74,    59,   95,    80,   68  ]

np_heights = np.array(heights)
np_weights = np.array(weights)

bmi = np_weights/(np_heights**2)
bmi
```

```
[25.68007405 23.88946281 20.65754    27.45982194 25.53544639 22.72043837]
```

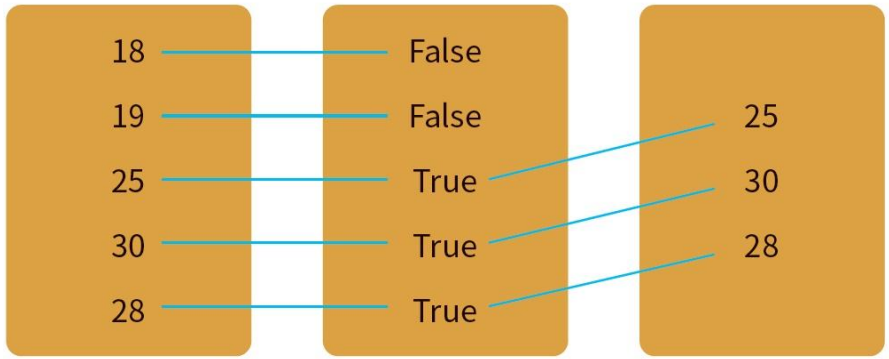
인덱싱과 슬라이싱


```
                0    1    2    3
grades=[88, 72, 93, 94]
>>> grades[2]
[93]
```

```
                0    1    2    3
grades=[88, 72, 93, 94]
>>> grades[1:3]
[72, 93]
```

논리적인 인덱싱


```
>>> ages = np.array([18, 19, 25, 30, 28])
>>> y = ages > 20
>>> y
array([False, False,  True,  True,  True])

>>> ages[ ages > 20 ]
array([25, 30, 28])
```



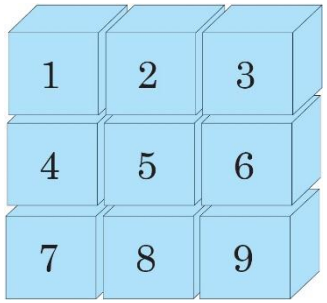
Lab: BMI가 20 이상인 사람 출력하기


```
import numpy as np

heights = [ 1.83, 1.76, 1.69, 1.86, 1.77, 1.73 ]
weights = [ 86,    74,    59,   95,    80,   68  ]

np_heights = np.array(heights)
np_weights = np.array(weights)

bmi = np_weights/(np_heights**2)
print(bmi[bmi > 25])
```

```
array([25.68007405, 27.45982194, 25.53544639])
```

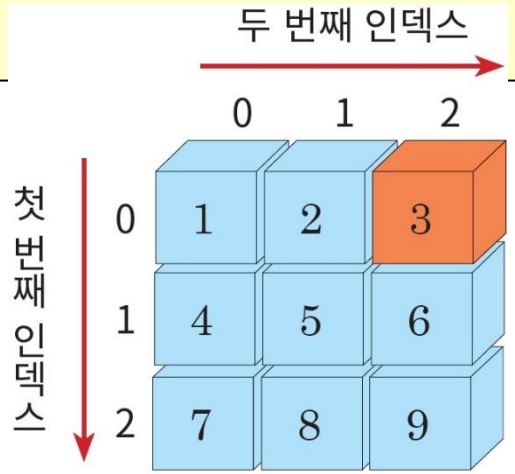
2차원 배열


```
>>> import numpy as np
>>> y = [[1,2,3], [4,5,6], [7,8,9]]
>>> y
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

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



2차원 배열에서 인덱스


```
>>> import numpy as np
>>> y = [[1,2,3], [4,5,6], [7,8,9]]
>>> y
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

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



arange() 함수


```
>>> import numpy as np
>>> 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])
```

linspace() 함수


```
>>> np.linspace(0, 10, 100)
array([ 0.        ,  0.1010101 ,  0.2020202 ,  0.3030303 ,  0.4040404 ,
        ...
        8.08080808,  8.18181818,  8.28282828,  8.38383838,  8.48484848,
        8.58585859,  8.68686869,  8.78787879,  8.88888889,  8.98989899,
        9.09090909,  9.19191919,  9.29292929,  9.39393939,  9.49494949,
        9.5959596 ,  9.6969697 ,  9.7979798 ,  9.8989899 , 10.        ])



>>> np.logspace(0, 5, 10)
array([1.00000000e+00, 3.59381366e+00, 1.29154967e+01, 4.64158883e+01,
       1.66810054e+02, 5.99484250e+02, 2.15443469e+03, 7.74263683e+03,
       2.78255940e+04, 1.00000000e+05])
```

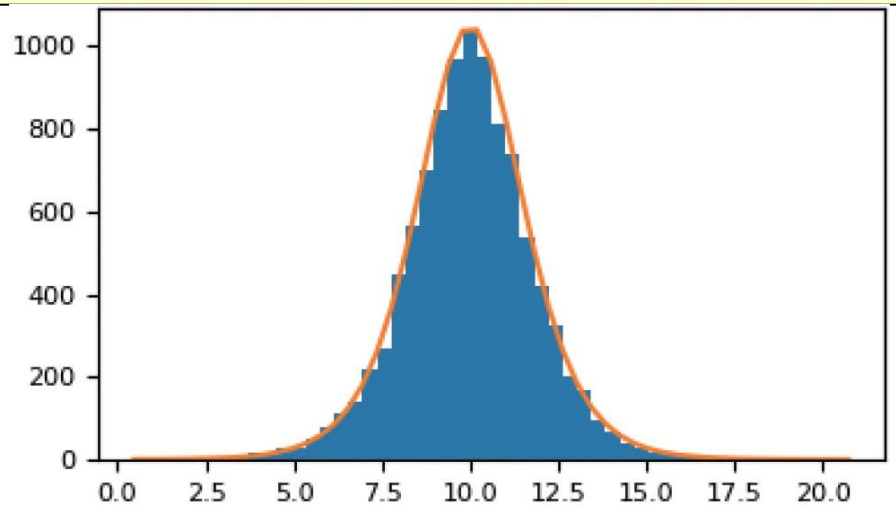
reshape() 함수


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


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

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

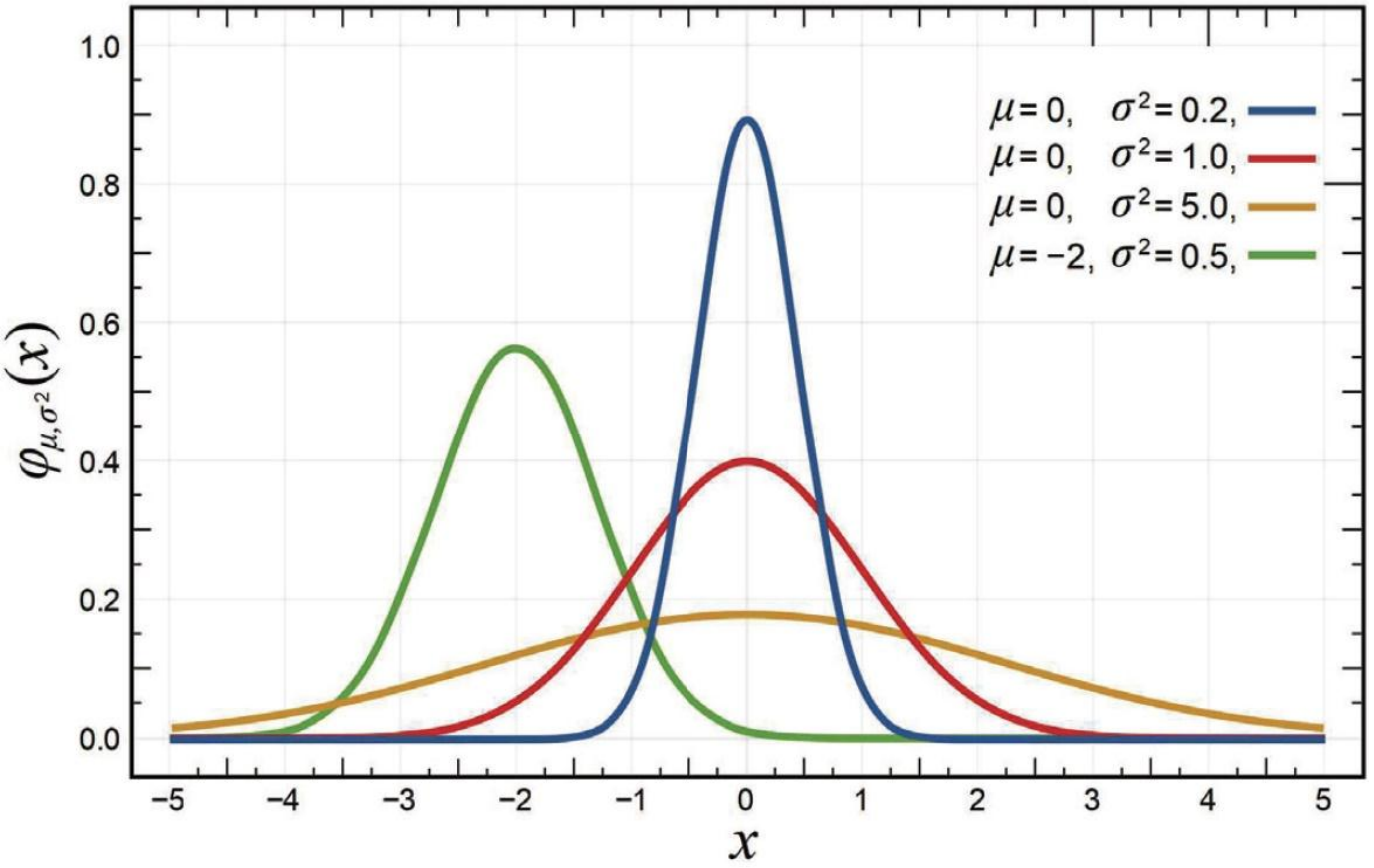
난수 생성

INFINITYBOOKS
인피니티북스

```
>>> 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]])
```



정규 분포 난수 생성



(이미지 출처: 위키 백과)

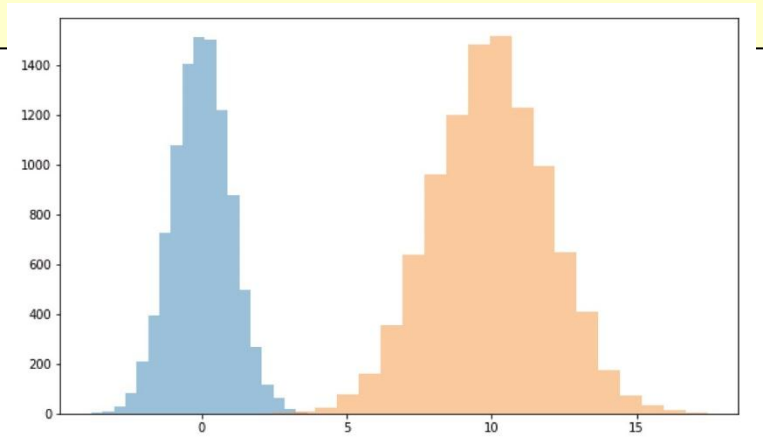
정규 분포 난수 생성


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

Lab: 정규 분포 그래프 그리기

INFINITYBOOKS
인피니티북스

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

m = 10; sigma = 2
x1 = np.random.randn(10000)
x2 = m+sigma*np.random.randn(10000)

plt.figure(figsize=(10,6))
plt.hist(x1, bins=20, alpha=0.4)          # 20개의 상자를 이용하여 히스토그램 계산
plt.hist(x2, bins=20, alpha=0.4)
plt.show()
```



Summary

- 기계 학습(machine learning)은 인공지능의 한 분야로, 컴퓨터에 학습 기능을 부여하기 위한 연구 분야이다.

- 기계 학습은 “교사”의 존재 여부에 따라 크게 지도 학습과 비지도 학습으로 나누어진다. 또 강화학습도 있다.
- 지도 학습은 크게 회귀와 분류로 나눌 수 있다. 회귀는 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것이다. 분류(classification): 입력을 두 개 이상의 유형으로 분할하는 것이다.
- 비지도 학습은 “교사” 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다. 비지도 학습에서는 데이터들의 상관도를 분석하여 유사한 데이터들을 모으게 된다.
- 강화 학습에서는 컴퓨터가 어떤 행동을 취할 때마다 외부에서 처벌이나 보상이 주어진다.

Q & A