AI 특강

Prof. Jibum Kim

Department of Computer Science & Engineering

Incheon National University

- **Python의 Pandas를 이용하여 data 시각화 및 상관 관계 분석 (핸즈온 책 chapter 2)**

- 오늘 최초 실습에서 사용할 실제 데이터
- 주어진 데이터: 미국 캘리포니아 인구조사 데이터
- 특징: 구역(block)별 인구, 중간 소득, 경도, 위도 등
- Label: 중간 주택 가격
- 학습 내용: 데이터 시각화, 특징간 상관관계
- 사용할 라이브러리: Python Pandas (데이터 처리 분석),NumPy (행렬 수치 연산) Matplotlib (시각화), Scikit-learn (머신러닝 알고리즘 구현)등

Take a Quick Look at the Data Structure

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

- **Python의 pandas의 dataframe 활용**
- **head(): 데이터의 상위 N행 봄. 기본 N=5**
- **info(): 데이터에 대한 전반적인 정보**
- **describe(): 열별 요약 통계량 (수치형만)**
- **hist(): histogram 보기**

- **특이점: 다른 열 (수치형)과 다르게 ocean_proximity열은 자료형이 다름 (object형)**

```
In [6]: housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

- **“ocean_proximity”: 해안 근접도**

```
In [7]: housing["ocean_proximity"].value_counts()

Out[7]: <1H OCEAN     9136
        INLAND        6551
        NEAR OCEAN    2658
        NEAR BAY      2290
        ISLAND           5
        Name: ocean_proximity, dtype: int64
```

각 열별 통계정보 보여줌

In [8]:
```
housing.describe()
```

Out[8]:

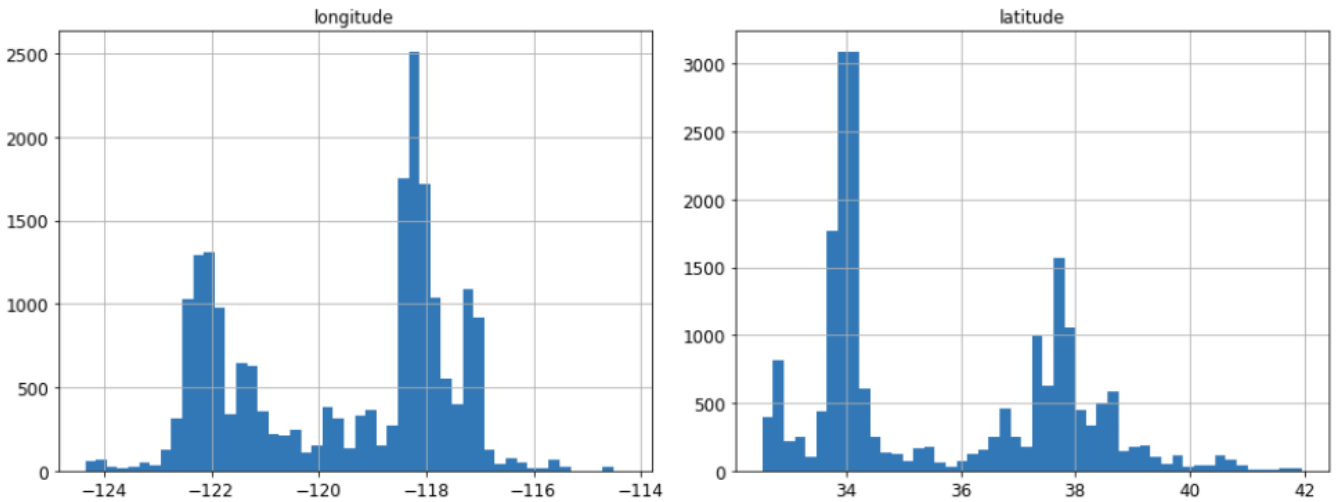
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

히스토그램 시각화 (각 열별)

In [9]:

```
%matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
save_fig("attribute_histogram_plots")
plt.show()
```

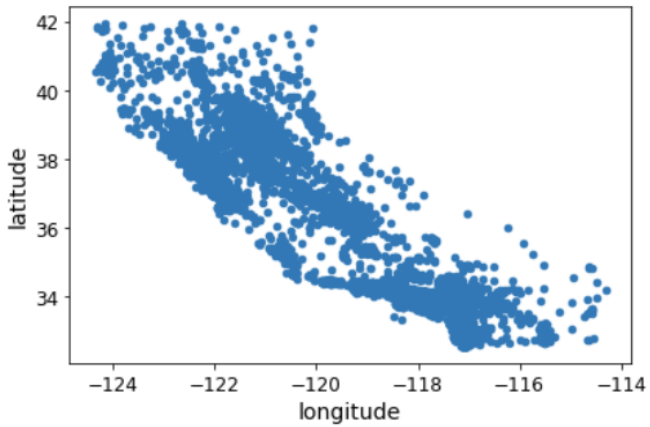
```
Saving figure attribute_histogram_plots
```



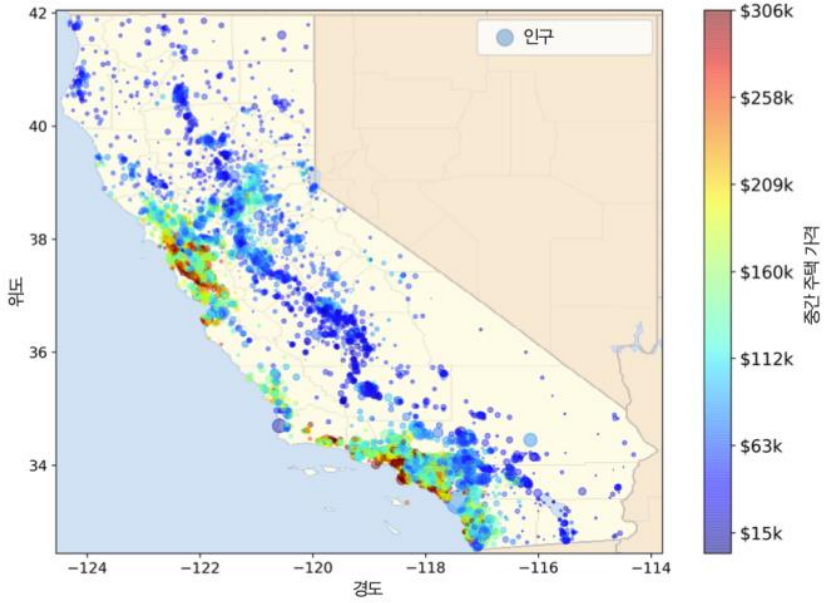
- **Training data 시각화:** 지리적 데이터 시각화 (scatter plot)
- 구역이 집결된 지역과 그렇지 않은 지역 구분 가능
- 샌프란시스코의 베이 에어리어, LA, 샌디에고 등 밀집된 지역 확인 가능
- 아래: bad visualization 예

```
In [33]: housing.plot(kind="scatter", x="longitude", y="latitude")
         save_fig("bad_visualization_plot")
```

```
Saving figure bad_visualization_plot
```



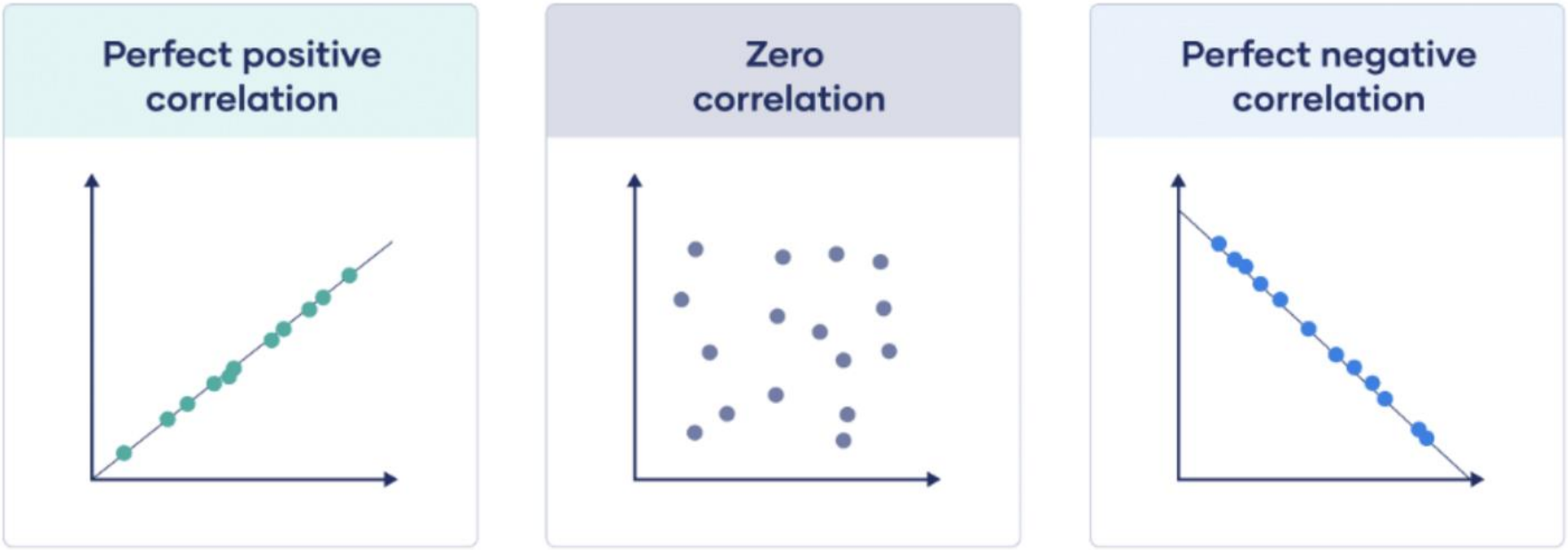
- 주택 가격이 해안 근접도, 인구 밀도와 관련이 큼
- 해안 근접도: 위치에 따라 다르게 작용 **대도시 근처: 해안 근처 주택 가격이 상대적 높음**
- 북부 캘리포니아 지역: 높지 않음



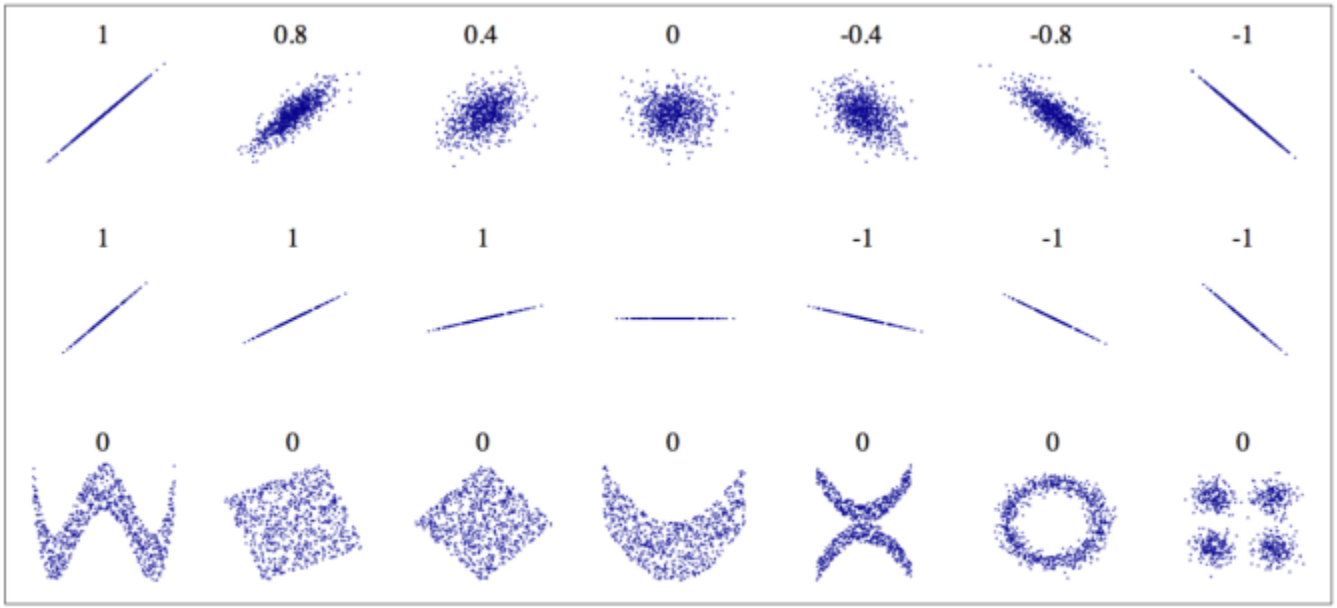
- **상관관계 조사**
- **중간 주택 가격 특성과 다른 특성 사이의 상관관계: 상관계수 활용**

```
In [39]: corr_matrix["median_house_value"].sort_values(ascending=False)

Out[39]: median_house_value    1.000000
         median_income         0.687160
         total_rooms           0.135097
         housing_median_age    0.114110
         households            0.064506
         total_bedrooms        0.047689
         population           -0.026920
         longitude            -0.047432
         latitude             -0.142724
         Name: median_house_value, dtype: float64
```



- **상관계수 (standard correlation coefficient)의 특징**
- **상관계수: [-1, 1]구간의 값**
- **1에 가까울 수록: 강한 양의 선형 상관관계**
- **-1에 가까울 수록: 강한 음의 선형 상관관계**
- **0에 가까울 수록: 매우 약한 선형 상관관계**



- 상관계수를 통해 확인할 수 있는 정보 중간 주택 가격과 중간 소득의 상관계수가 0.68로 가장 높음
중간 소득이 올라가면 중간 주택 가격도 상승하는 경향이 있음

```
In [39]: corr_matrix["median_house_value"].sort_values(ascending=False)

Out[39]: median_house_value    1.000000
         median_income         0.687160
         total_rooms           0.135097
         housing_median_age    0.114110
         households            0.064506
         total_bedrooms        0.047689
         population           -0.026920
         longitude            -0.047432
         latitude             -0.142724
         Name: median_house_value, dtype: float64
```

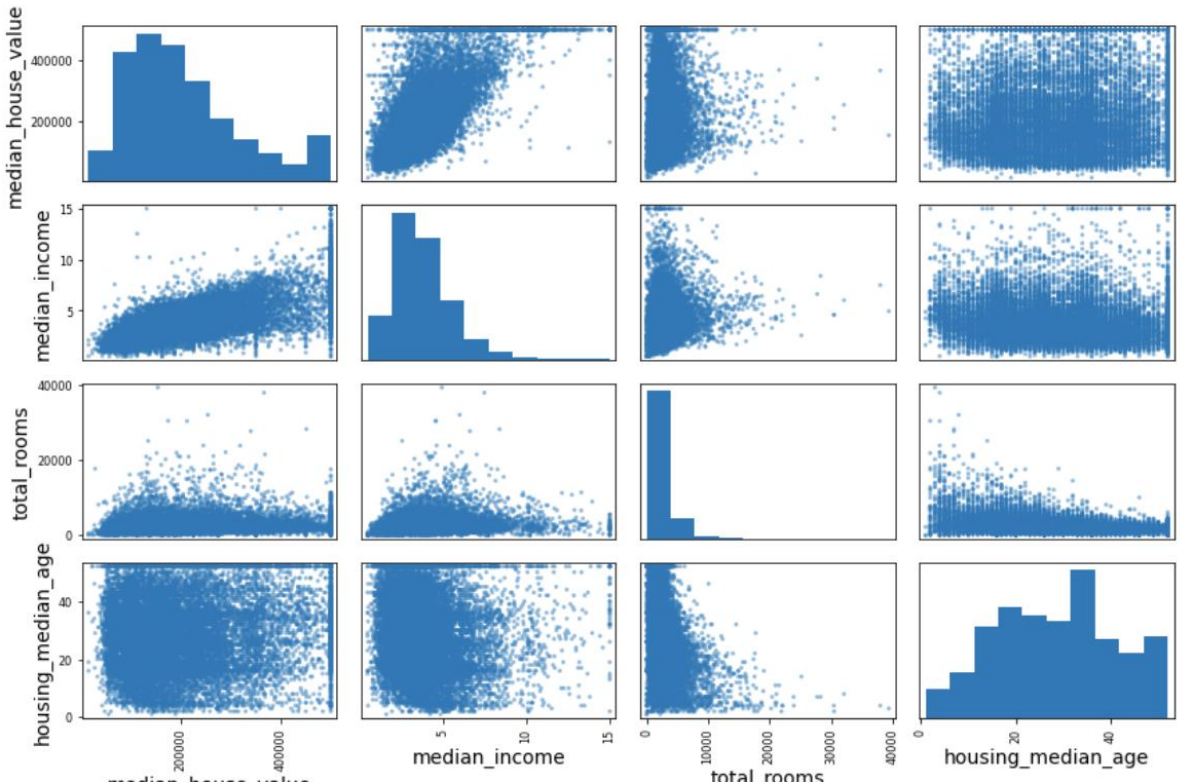
각 특징 (열)끼리의 상관관계 시각화

In [40]:

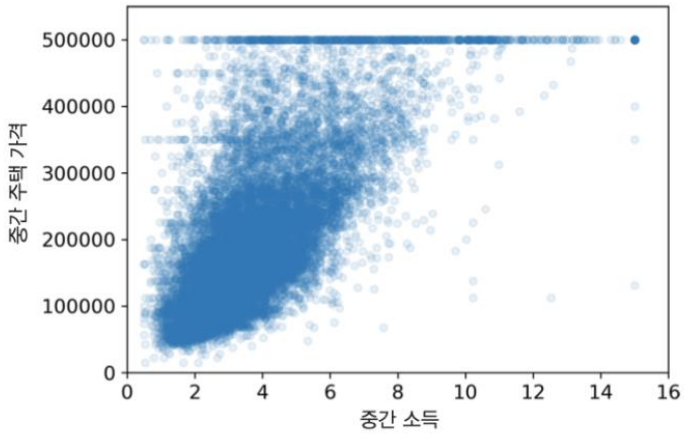
```
# from pandas.tools.plotting import scatter_matrix # For older versions of Pandas
from pandas.plotting import scatter_matrix

attributes = ["median_house_value", "median_income", "total_rooms",
              "housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
save_fig("scatter_matrix_plot")
```

```
Saving figure scatter_matrix_plot
```



- 중간 주택 가격과 중간 소득의 관계: 산점도 활용
- 점들이 너무 넓게 퍼져 있음. 완벽한 선형관계와 거리 멂.
- 50만 달러 수평선: 가격 제한
- 35만, 28만, 그 아래 정도에서도 수평선 존재



- 퍼셉트론 (Perceptron)

- 신경망은 machine learning (기계 학습) 역사에서 가장 오래된 기계 학습 모델

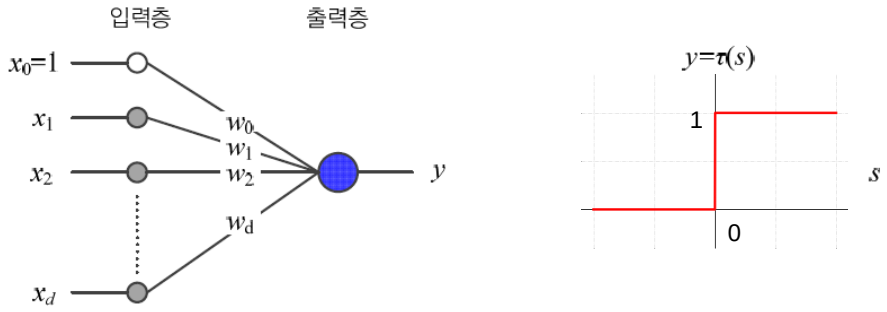
- 1950년대 퍼셉트론 -> 1980년대 다층 퍼셉트론
- 최근 딥러닝의 가장 기초

- 2000년대 딥러닝 등장

- 퍼셉트론 (perceptron)은 노드, 가중치, 층과 같은 새로운 개념을 도입하고 학습 알고리즘을 창안함

- 퍼셉트론은 원시적 신경망이지만, 딥러닝을 포함한 현대 신경망은 퍼셉트론을 병렬과 순차 구조로 결합하여 만듬 → 현대 신경망의 중요한 구성 요소

- 퍼셉트론이란 가장 간단한 형태의 인공 신경망으로 입력과 출력은 모두 숫자이며 입력은 가중치 (weight)라는 것과 연결되어 있다
- 여기서 젤 위에 있는 것은 바이어스 노드라 하고 출력은 한 개의 노드 이다
- 퍼셉트론에서는 다음 2가지 과정을 순차적으로 수행한다
- 1. 입력과 각각의 가중치를 모두 곱해서 더한 가중치합을 계산한다
- $s = w_0 + w_1 x_1 + \cdots w_d x_d = w_0 + \sum_{i=1}^{d} w_i x_i$
- 2. 이 가중치 합에 활성함수라고 하는 계단 함수를 통과시키킨다.
- 1에서 계산한 s가 0보다 크거나 같으면 1을 출력, s가 0보다 작으면 0을 출력



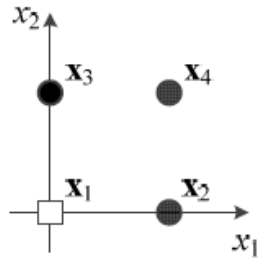
(a) 퍼셉트론의 구조 (b) 계단함수를 활성함수 $\tau(s)$로 이용함

그림 3-3 퍼셉트론의 구조와 동작

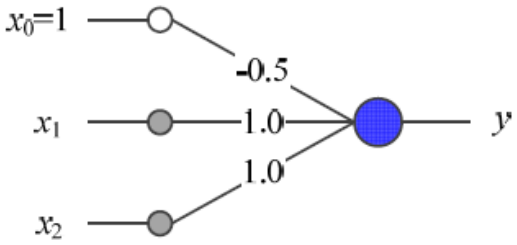
예제 3-1 퍼셉트론의 동작

2차원 특징 벡터로 표현되는 샘플을 4개 가진 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, $\mathbb{Y} = \{y_1, y_2, y_3, y_4\}$를 생각하자. [그림 3-4(a)]는 이 데이터를 보여준다.

$$\mathbf{x}_1 = \begin{pmatrix}0\\0\end{pmatrix},\ y_1 = -1,\ \ \mathbf{x}_2 = \begin{pmatrix}1\\0\end{pmatrix},\ y_2 = 1,\ \ \mathbf{x}_3 = \begin{pmatrix}0\\1\end{pmatrix},\ y_3 = 1,\ \ \mathbf{x}_4 = \begin{pmatrix}1\\1\end{pmatrix},\ y_4 = 0$$



(a) 훈련집합



(b) 퍼셉트론

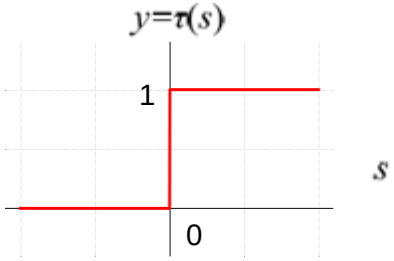
그림 3-4 OR 논리 게이트를 이용한 퍼셉트론의 동작 예시

▪OR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

샘플 4개를 하나씩 입력하여 제대로 분류하는지 확인해 보자.

$$\begin{aligned}
&\mathbf{x}_1\text{: } s = -0.5 + 0 * 1.0 + 0 * 1.0 = -0.5, && \tau(-0.5) = 0\\
&\mathbf{x}_2\text{: } s = -0.5 + 1 * 1.0 + 0 * 1.0 = 0.5, && \tau(0.5) = 1\\
&\mathbf{x}_3\text{: } s = -0.5 + 0 * 1.0 + 1 * 1.0 = 0.5, && \tau(0.5) = 1\\
&\mathbf{x}_4\text{: } s = -0.5 + 1 * 1.0 + 1 * 1.0 = 1.5, && \tau(1.5) = 1
\end{aligned}$$



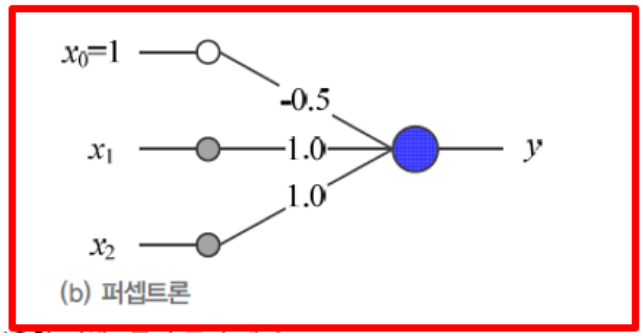
결국 [그림 3-4(b)]의 퍼셉트론은 샘플 4개를 모두 맞추었다. 이 퍼셉트론은 훈련집합을 100% 성능으로 분류한다고 말할 수 있다.

- Python으로 구현한 'OR' 퍼셉트론 예

```
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5,0.5])
    b = -0.4
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else :
        return 1

import numpy as np
print(OR(0,0))
print(OR(0,1))
print(OR(1,0))
print(OR(1,1))
```

- 이 퍼셉트론을 기하학적으로 설명하면
  - 결정 직선 $d(\mathbf{x}) = d(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_0 = 0$ → $x_1 + x_2 - 0.5 = 0$
    - w_1과 w_2는 직선의 방향, w_0은 절편을 결정
    - 결정 직선은 전체 공간을 1과 0의 두 부분공간으로 분할하는 분류기 역할
- 즉, 퍼셉트론은 선형 분류기라고 생각할 수 있다



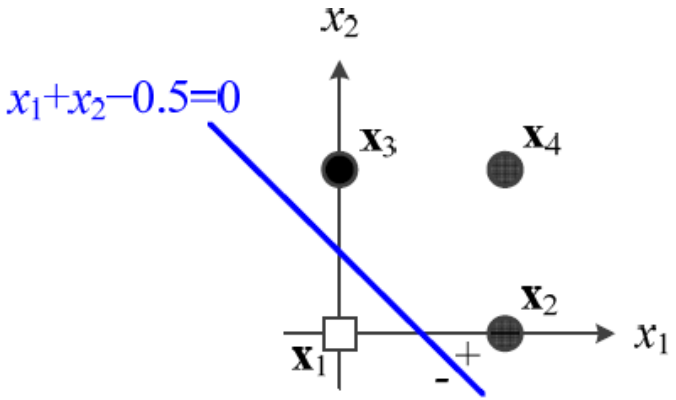
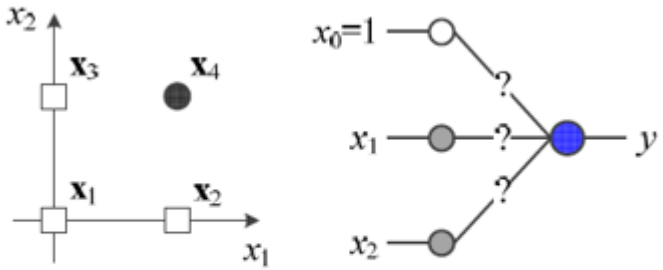


그림 3-5 [그림 3-4(b)]의 퍼셉트론에 해당하는 결정 직선

- 실습: 다음과 같은 AND Gate를 선형 분류할 수 있는 퍼셉트론을 직접 구현해보자

▪AND Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



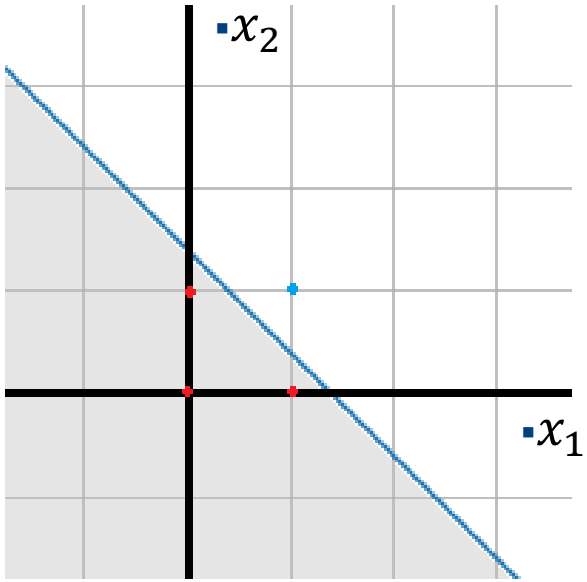
(a) AND 분류 문제

- **퍼셉트론으로 나타낸 AND Gate** $(x_1, x_2, \theta) = (0.5, 0.5, 0.7)$

- $$y = \begin{cases} 0 & (0.5x_1 + 0.5x_2 \leq 0.7) \\ 1 & (0.5x_1 + 0.5x_2 > 0.7) \end{cases}$$

▪AND Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



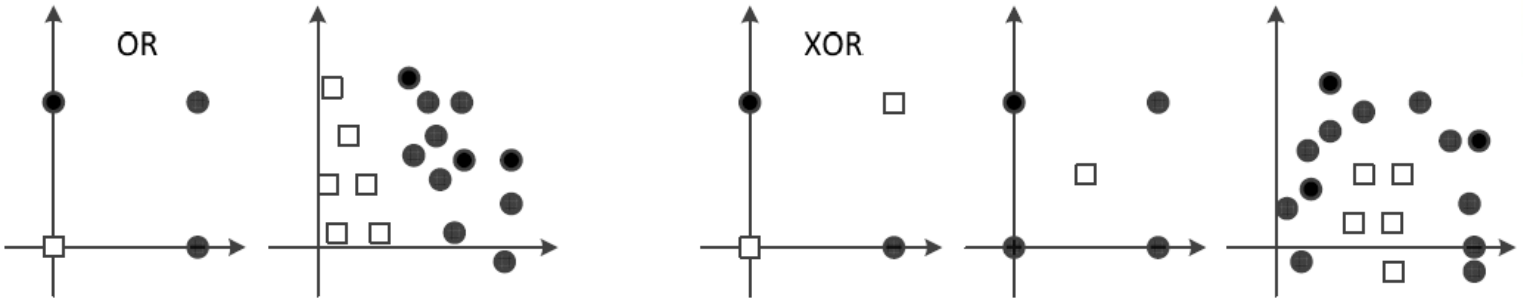
```
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5,0.5])
    b = -0.7
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else :
        return 1

import numpy as np
print(AND(0,0))
print(AND(0,1))
print(AND(1,0))
print(AND(1,1))
```

- **단층 퍼셉트론의 한계**
- **지금까지 배운 내용을 단층 퍼셉트론이라고 한다**
- **단층 퍼셉트론은 단순한 선형 분류기 이기 때문에 XOR에 대한 선형 분류가 불가능하다**

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



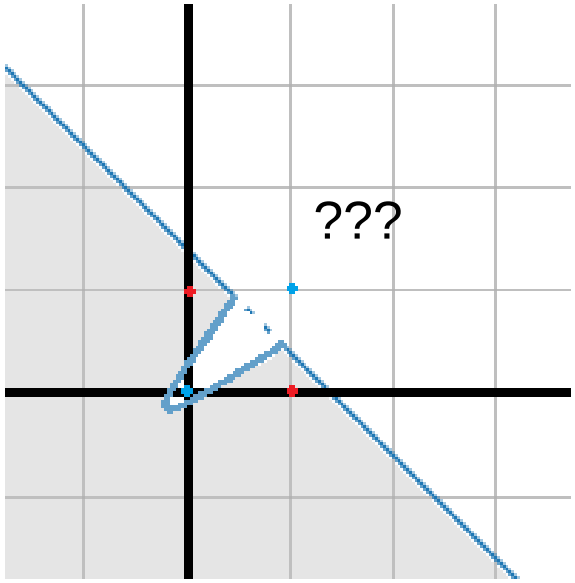
(a) 선형분리 가능 (b) 선형분리 불가능

그림 3-7 선형분리가 가능한 상황과 불가능한 상황

- **다층 퍼셉트론 (MLP)**

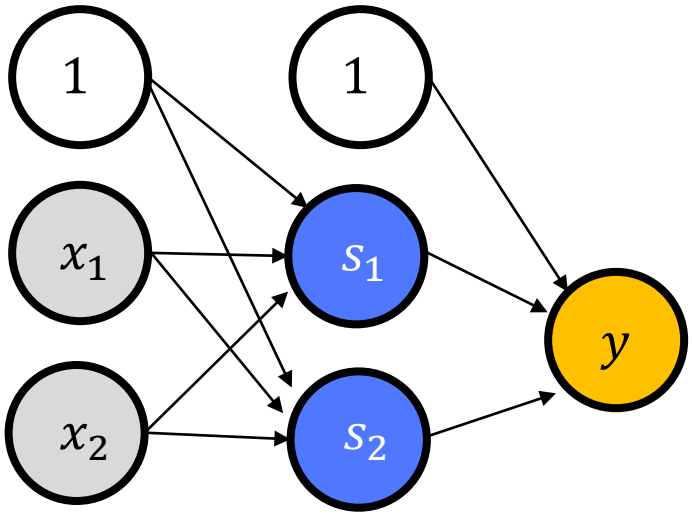
- 다층 퍼셉트론 (Multi-Layer Perceptron, MLP) 개요
- XOR과 같이 직선 하나로는 표현 불가능한 영역도 존재한다.

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

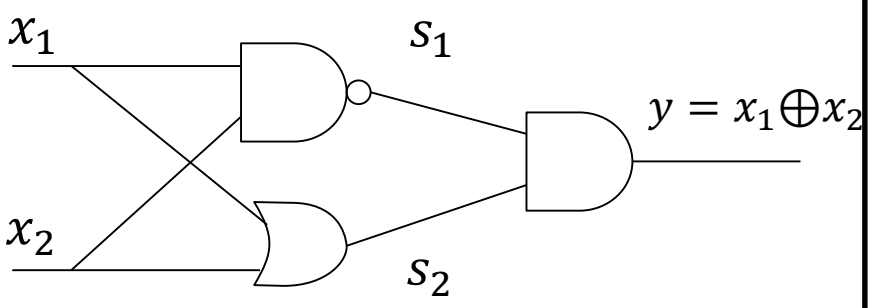


- 다층 퍼셉트론은 말 그대로, 퍼셉트론을 여러층 쌓는 것이다.
- i층의 퍼셉트론의 출력들이 i+1층의 퍼셉트론 입력이 된다.

- 입력데이터의 층을 입력층,
- 출력 신호가 나오는 퍼셉트론층을 출력층,
- 그 외 층들은 보이지 않는다 하여 은닉층이라 한다.



- 다층 퍼셉트론으로 XOR을 나타내보자.
- XOR은 NAND (Not-AND), OR, AND Gate로 나타낼 수 있다.

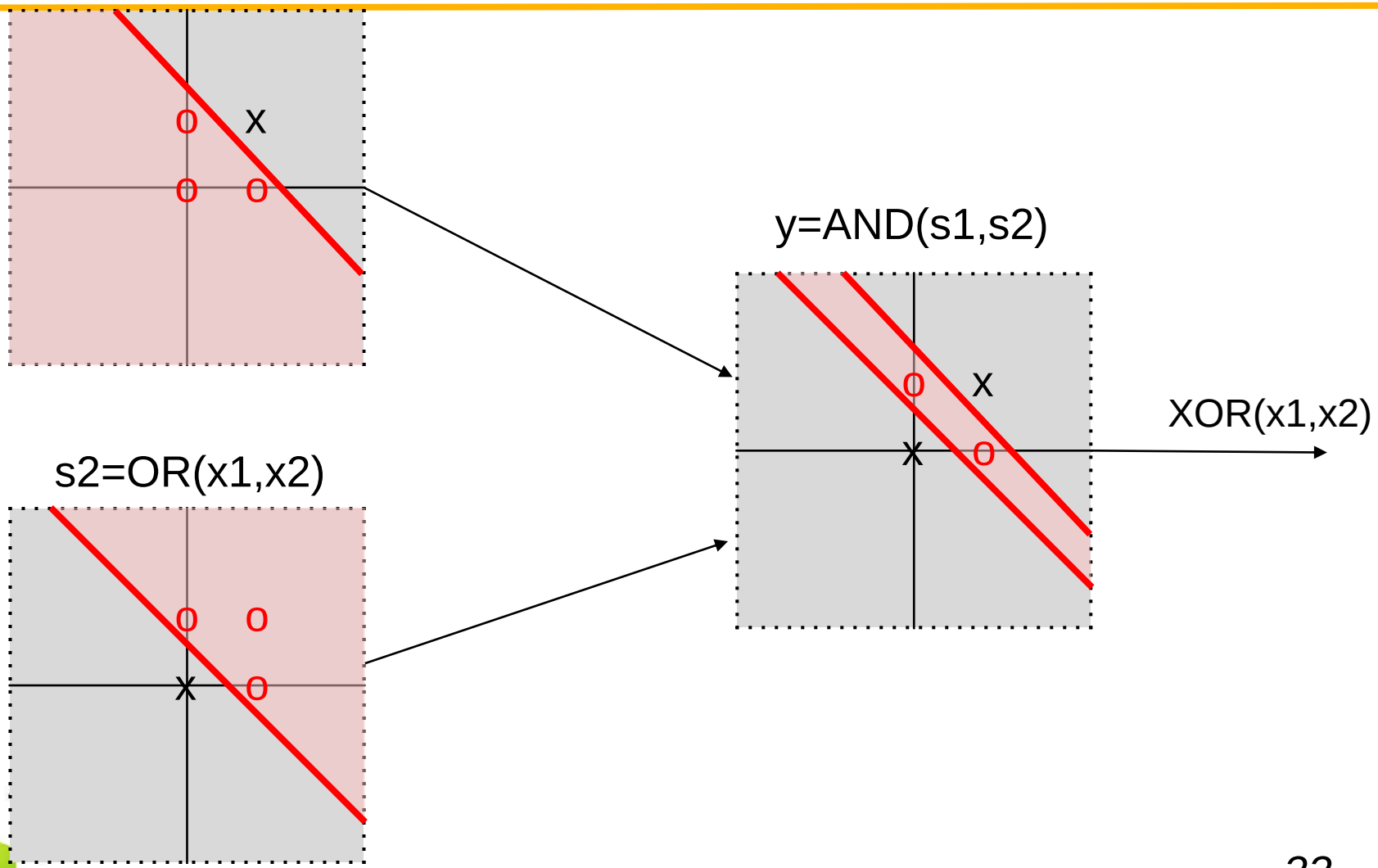



```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

x1	x2	s1	s2	y
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



- **실습**