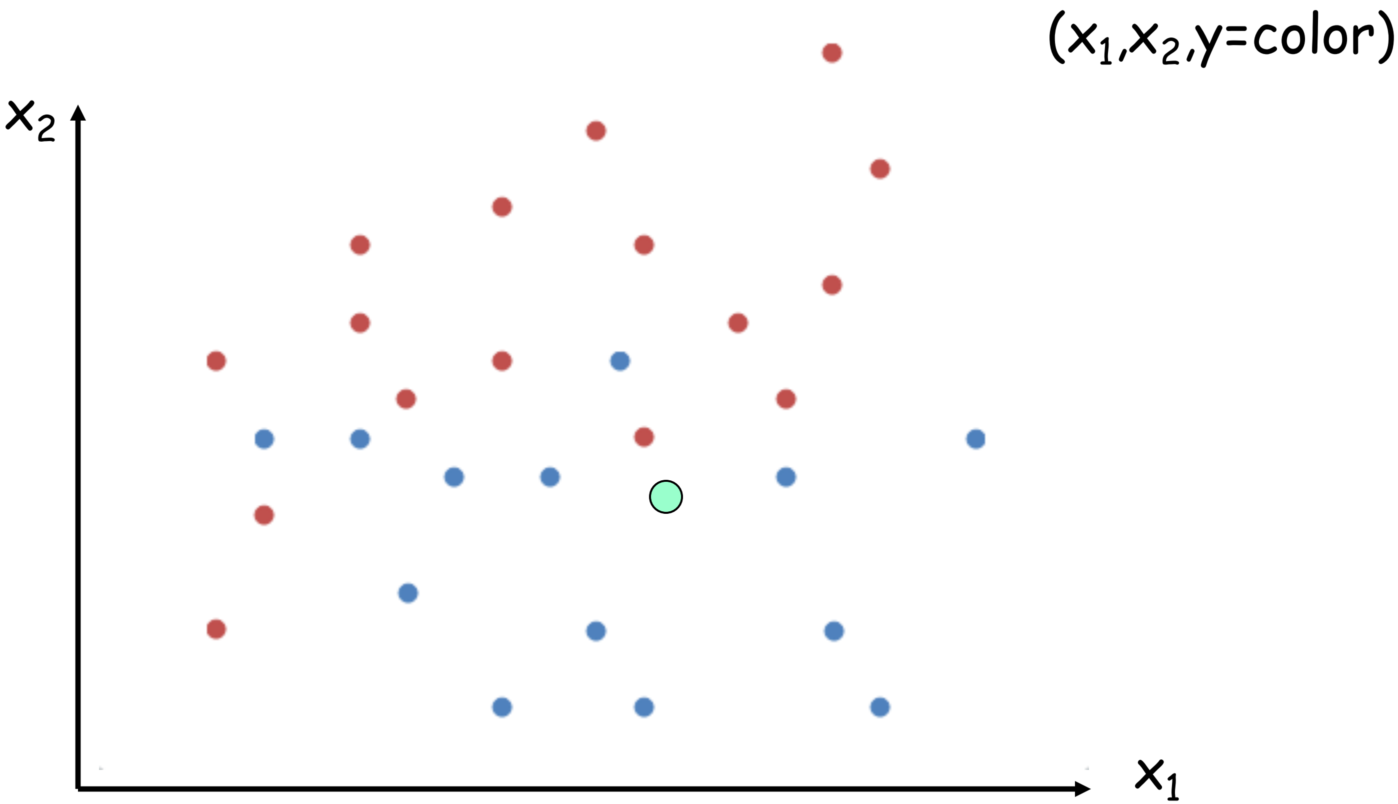
k-Nearest Neighbors (k-NN)

If your friends are RED, you must be RED.

Contents

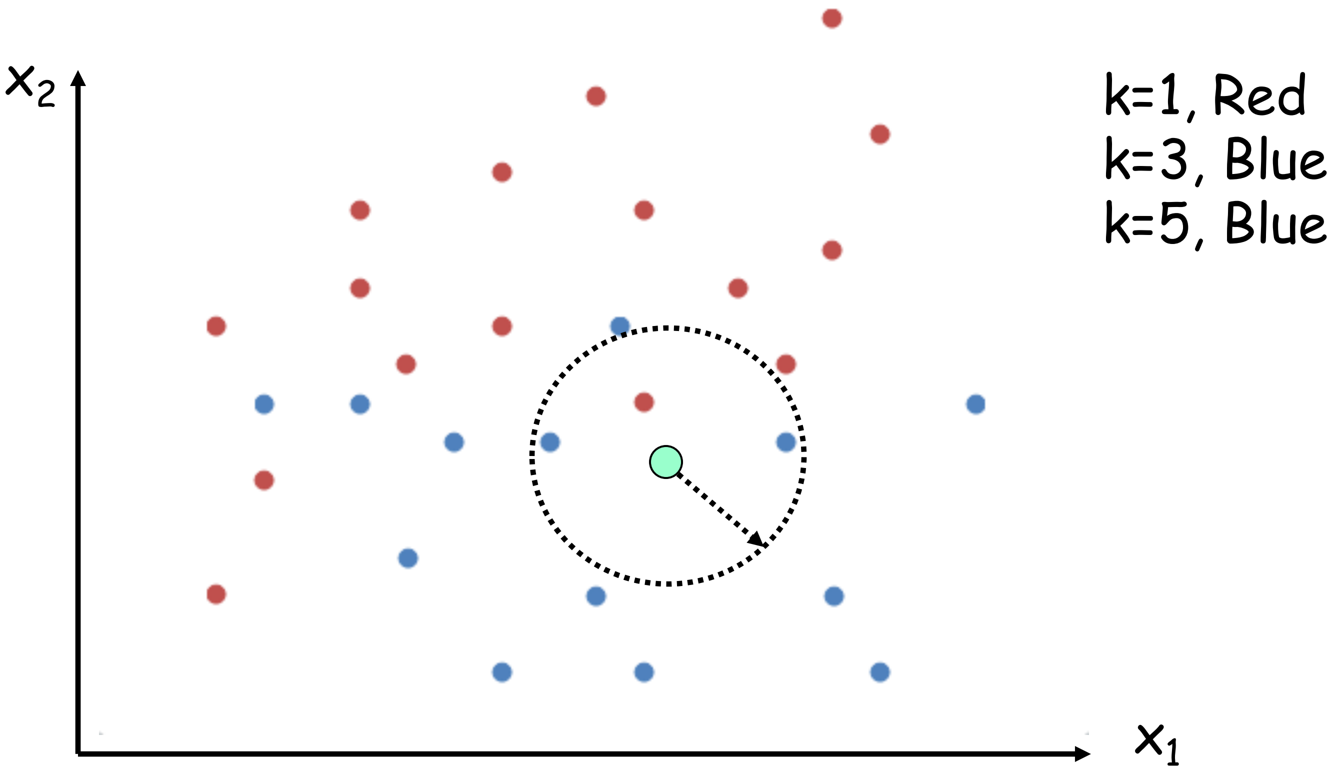
Classification

- **How to Predict Class of Unknown Data?**



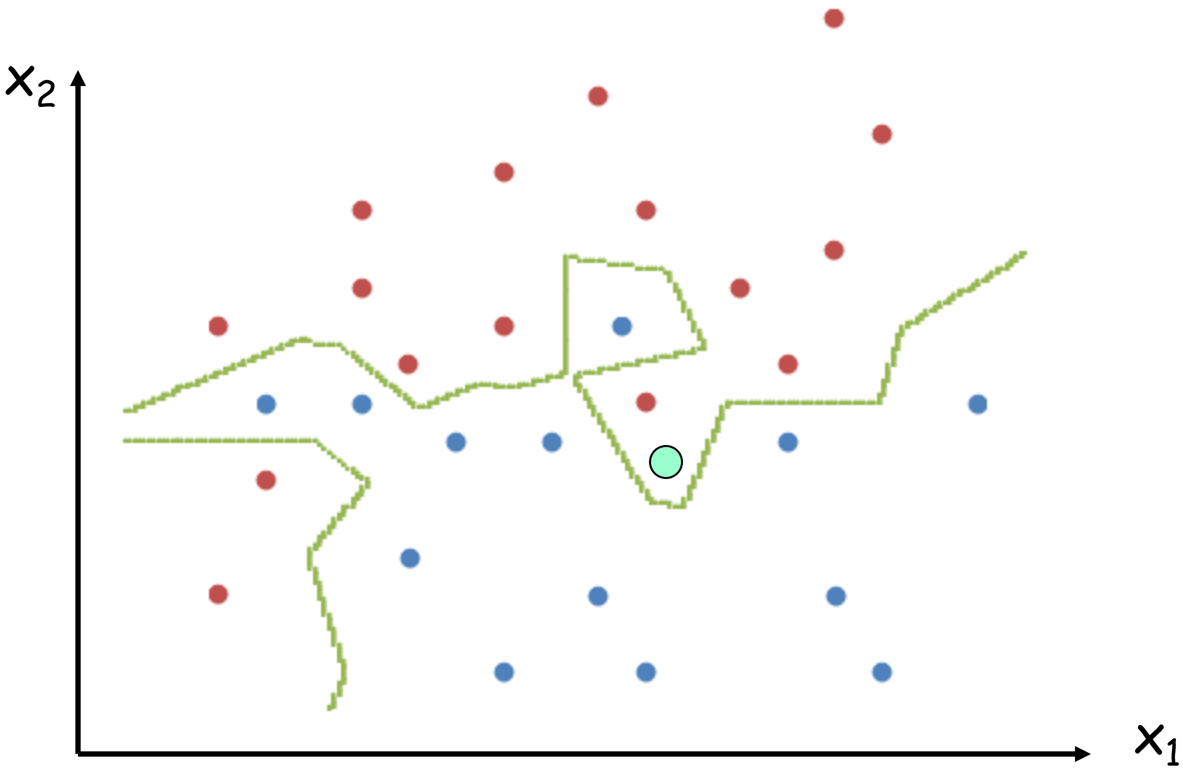
Classification

- **K-Nearest Neighbors**
  - Choose *k* nearest neighbors
  - Determine the class based on the majority



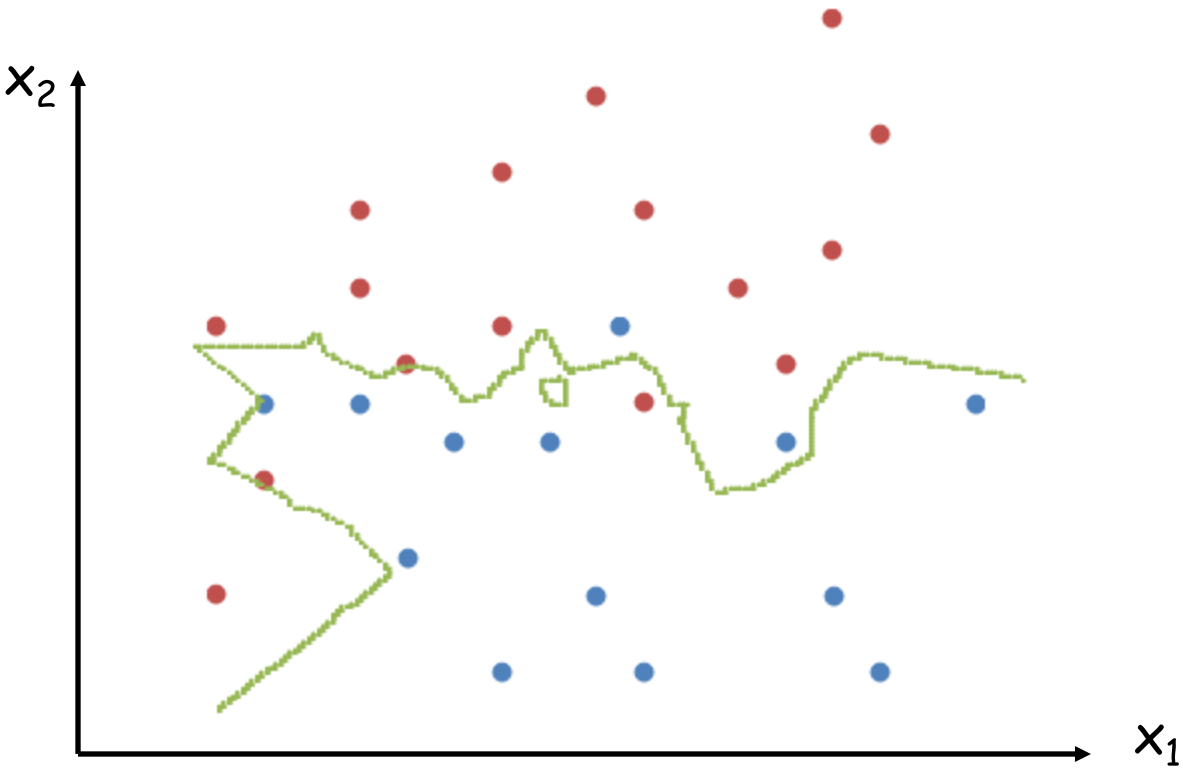
Classification

- **K-Nearest Neighbors**
  - K =1



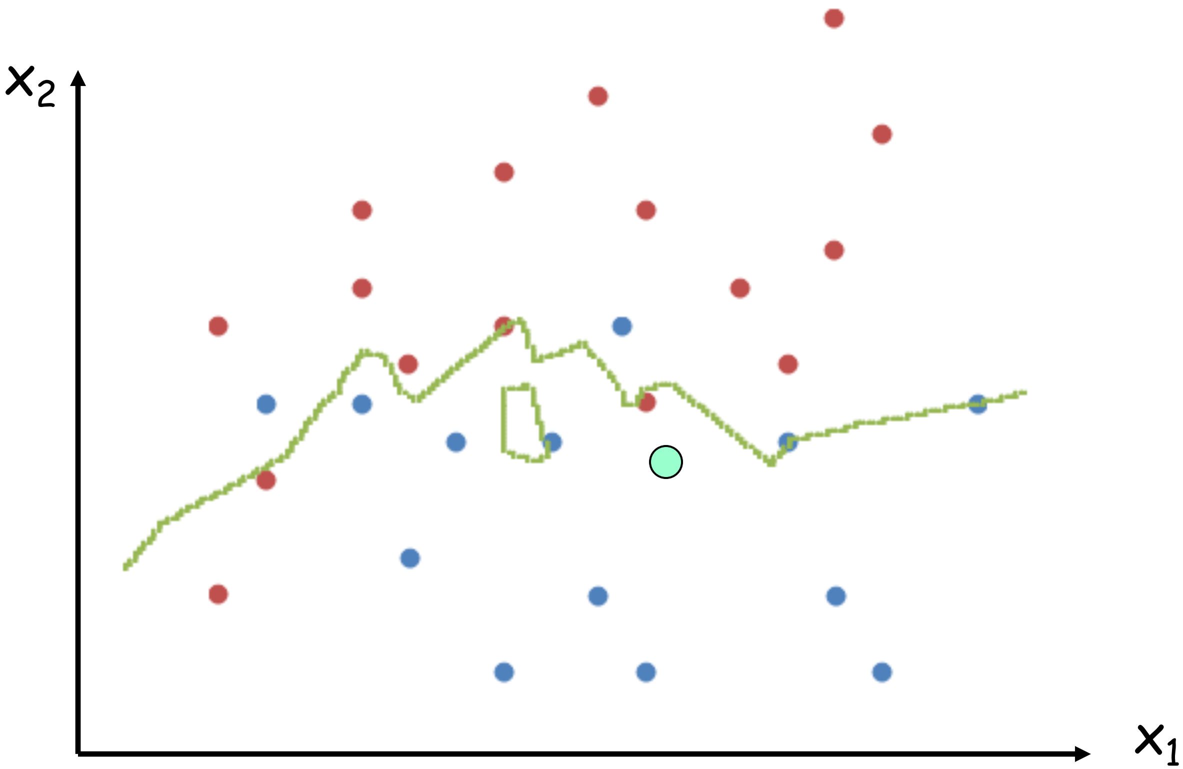
Classification

- **K-Nearest Neighbors**
  - K =3



Classification

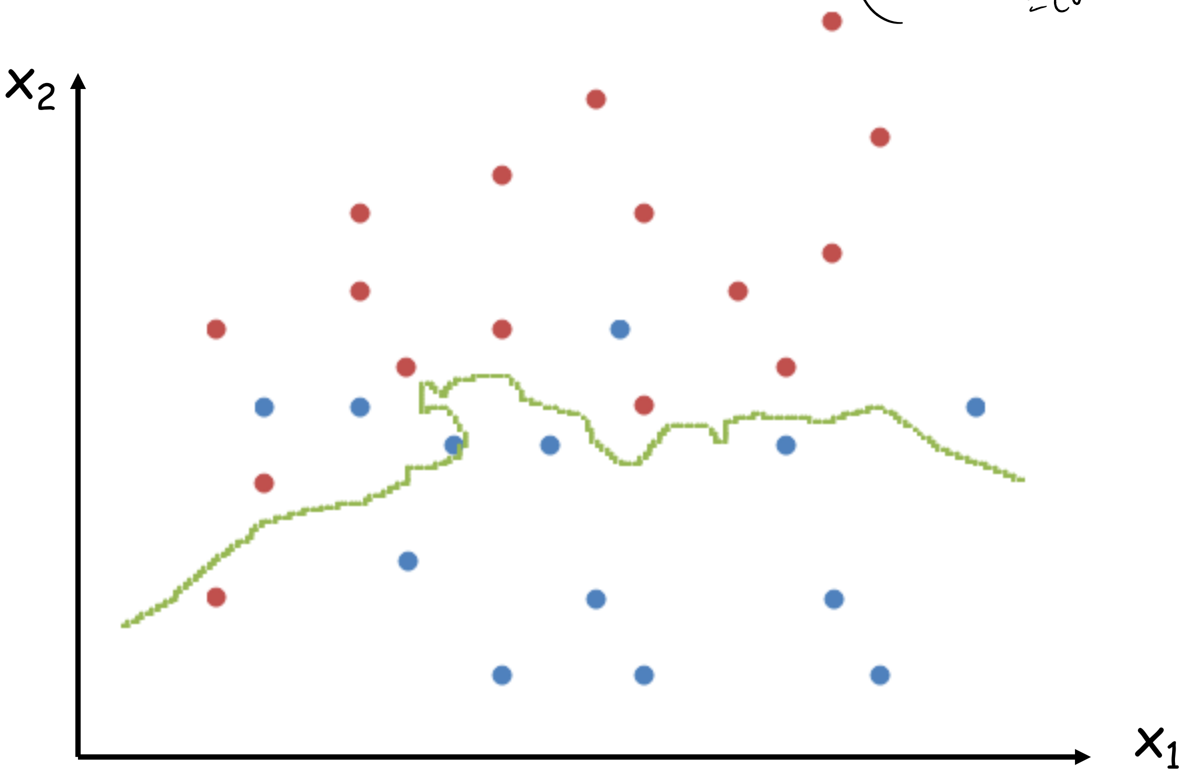
- **K-Nearest Neighbors**
  - K =5



Classification

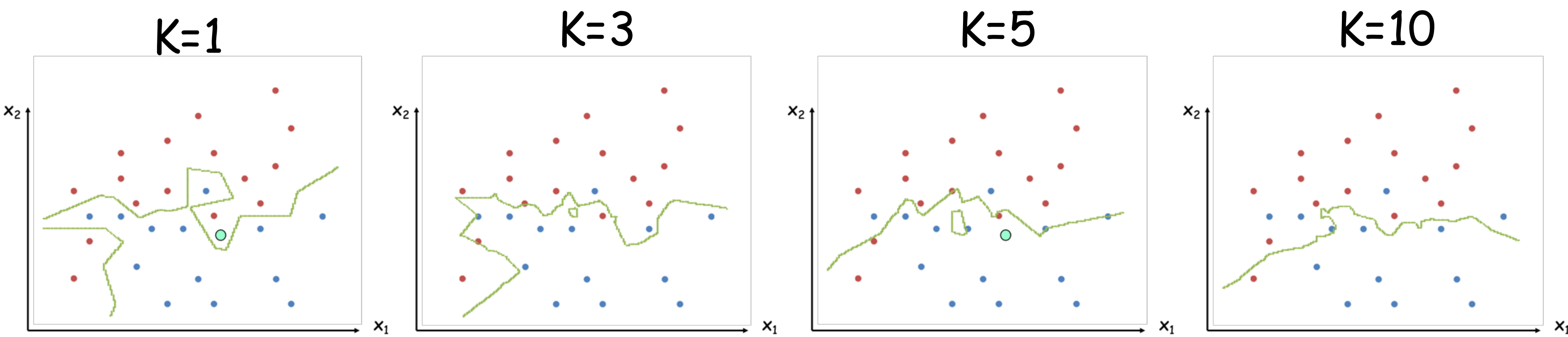
- **K-Nearest Neighbors**
  - K =10

label
(x_1, x_2, y)
$= color$



Classification

- **Then, which "k"?**



복잡

Boundary

간단

Complex — Simple

충실도 (정확성)

Fidelity to data

High — Low

민감성

Sensitivity to noisy data

Unreliable

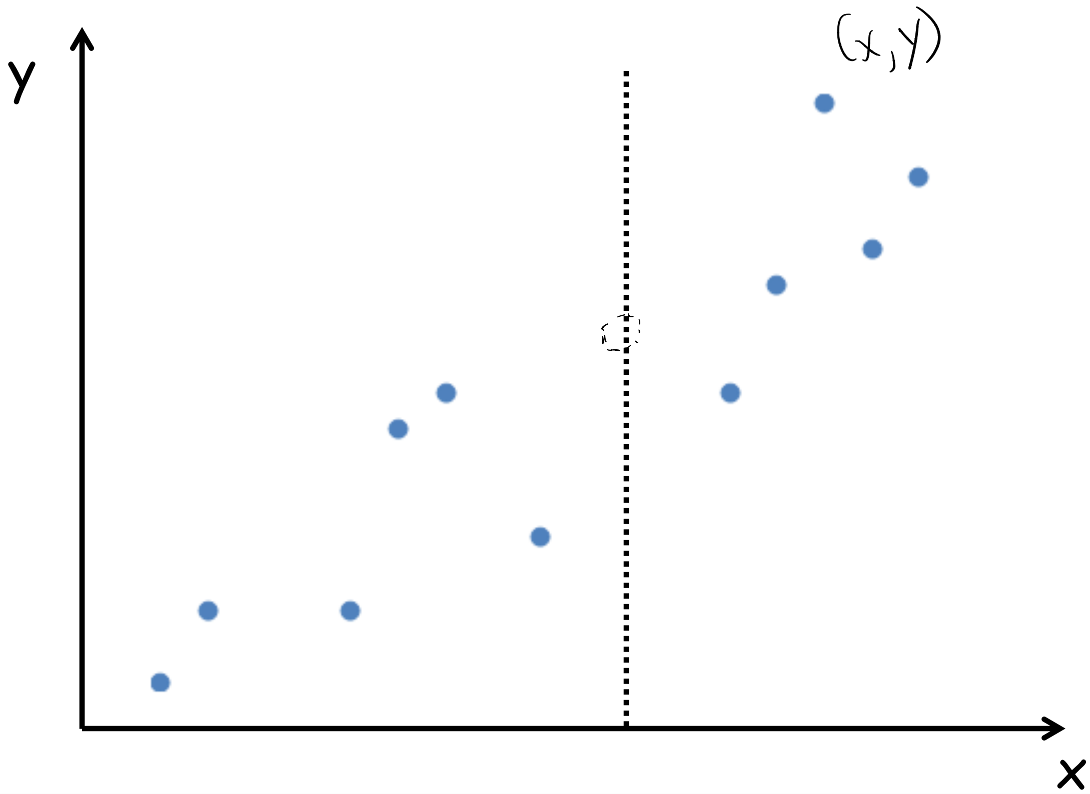
잘 안바뀜

High — Low

Stable

Regression

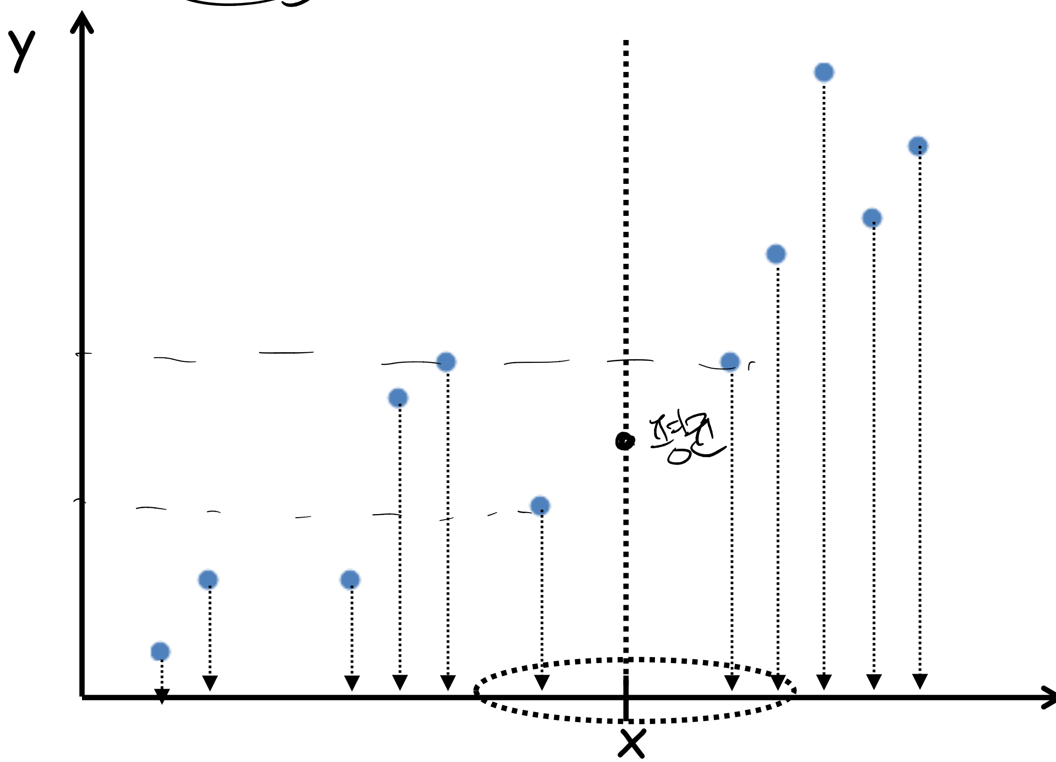
- **How to predict "y" value of unknown data**



Regression

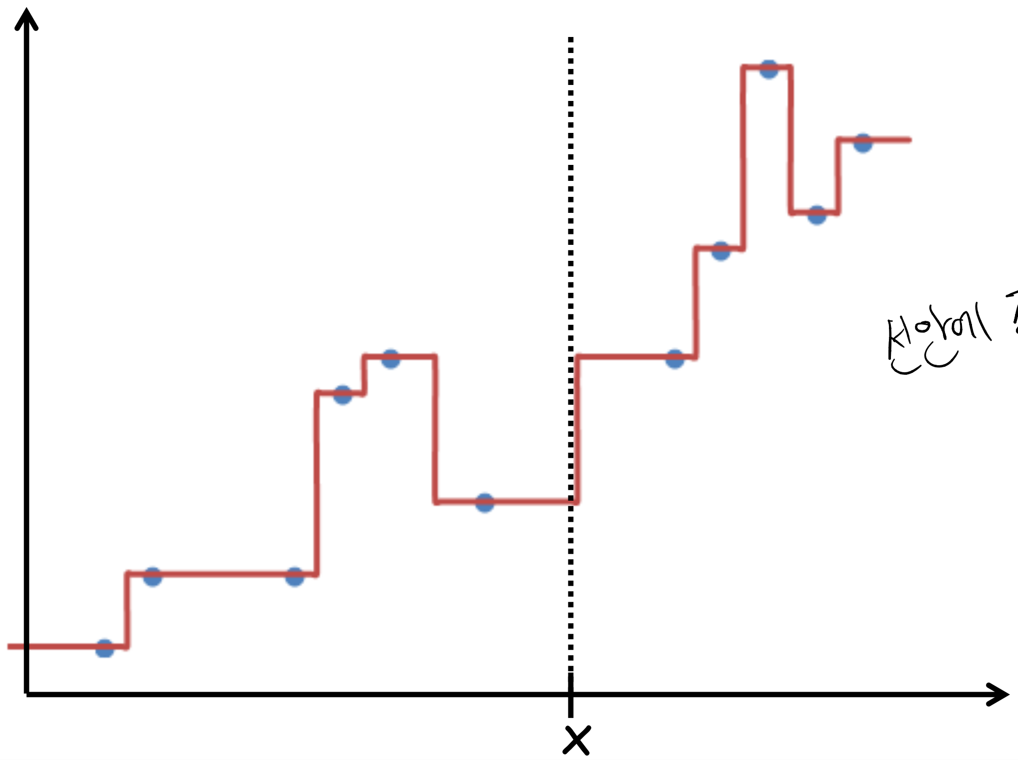
- **K-Nearest Neighbors**
  - Choose *k* nearest neighbors in X axis X축만
  - Predict the average of their “y”

(x_1, y_1)
(x_2, y_2)
(x_3, y_3)
$\vdots$
(x_{11}, y_{11})



Regression

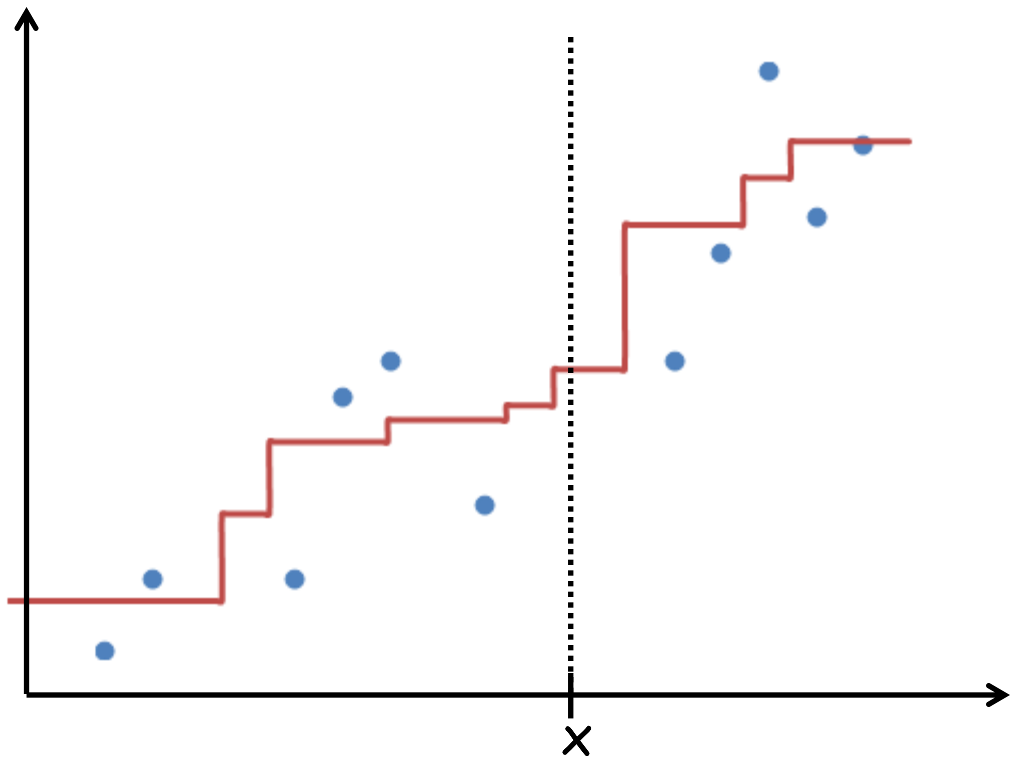
- **K-Nearest Neighbors**
  - K=1



선안에 점이 존재할 것

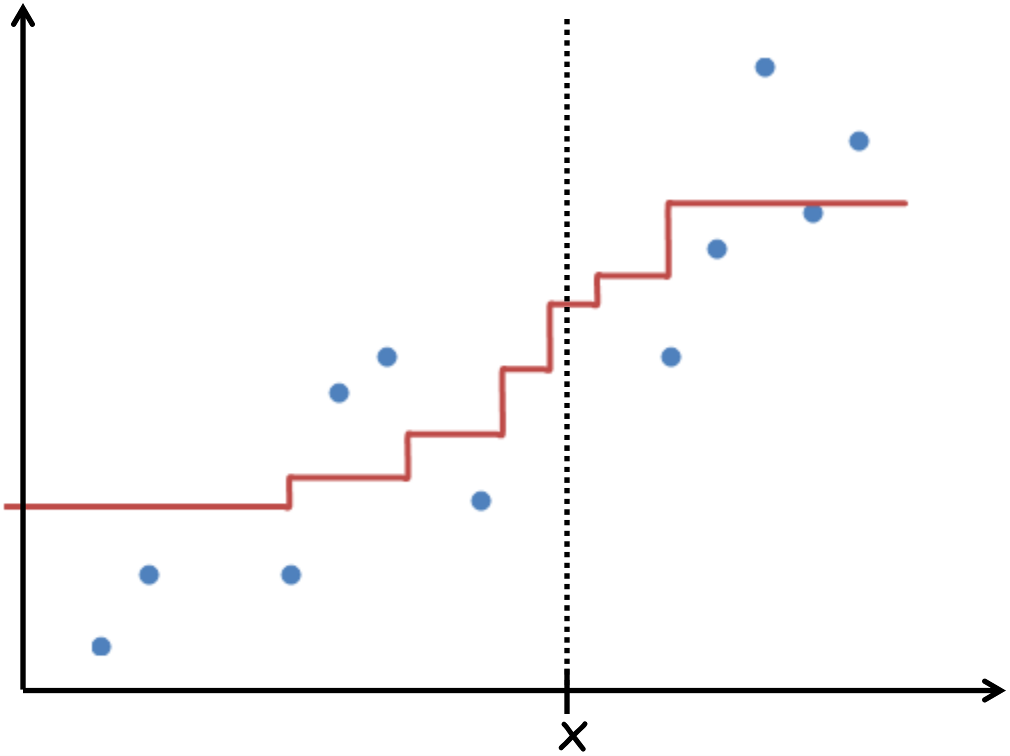
Regression

- **K-Nearest Neighbors**
  - K=3



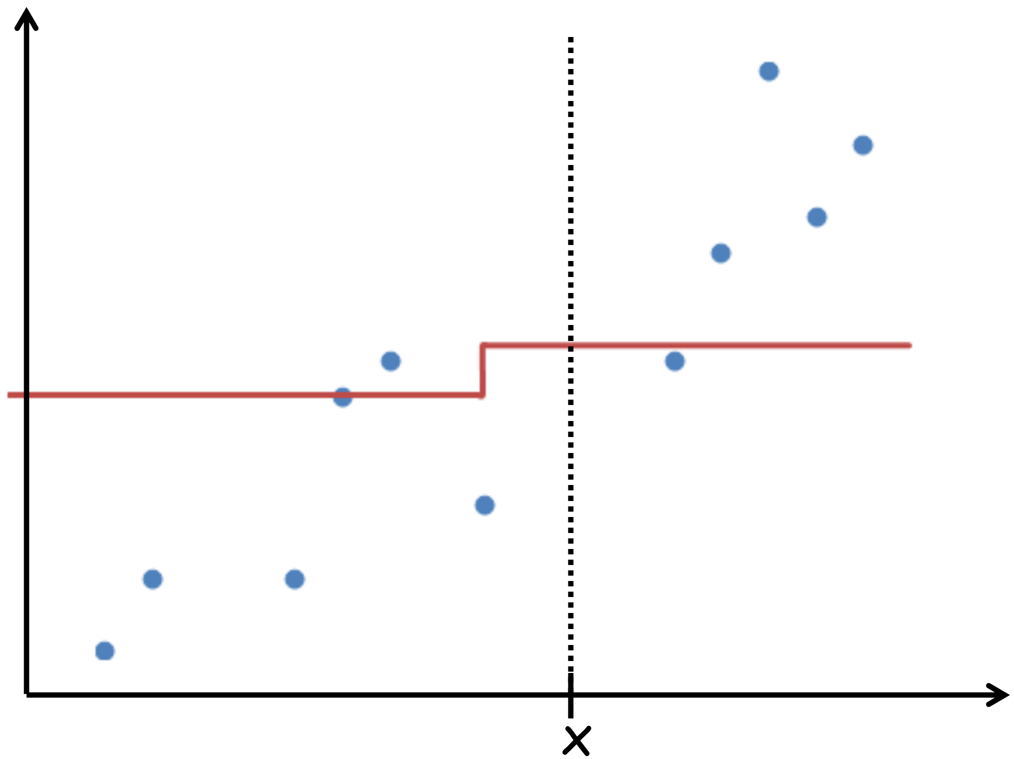
Regression

- **K-Nearest Neighbors**
  - K=5



Regression

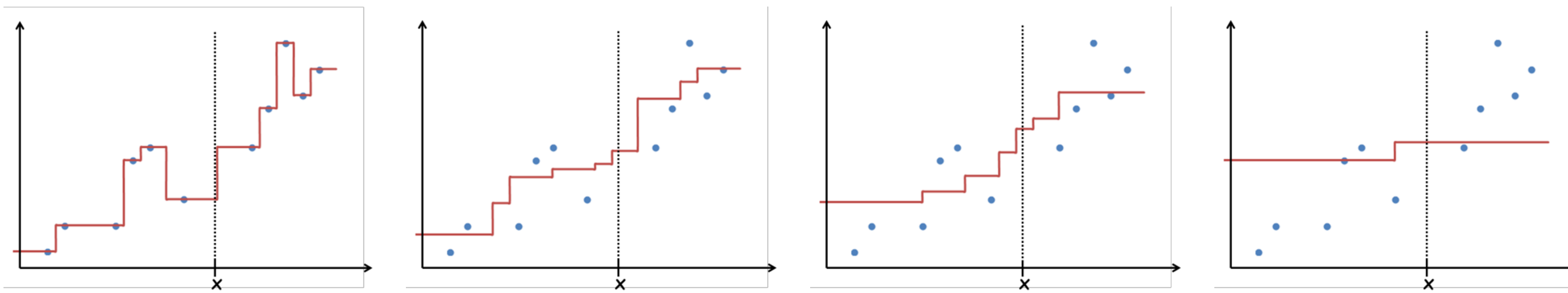
- **K-Nearest Neighbors**
  - K=10



~~Classification~~ Regression

- **Then, which "k"?** 뭐가 좋을진 상황에따라!

K=1 K=3 K=5 K=10



Model output

Complex ⟷ Simple

Fidelity to data

High ⟷ Low

Sensitivity to noisy data

High ⟷ Low

Variation

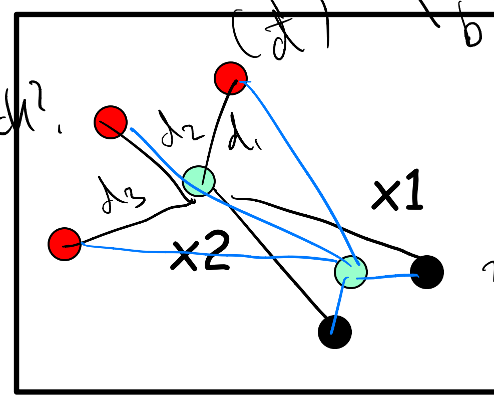
- **Why "just" counting or averaging?**

가중치비교 $\left(\frac{1}{d}\right)$

Red: $\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3}$

black: $\frac{1}{d_4} + \frac{1}{d_5}$

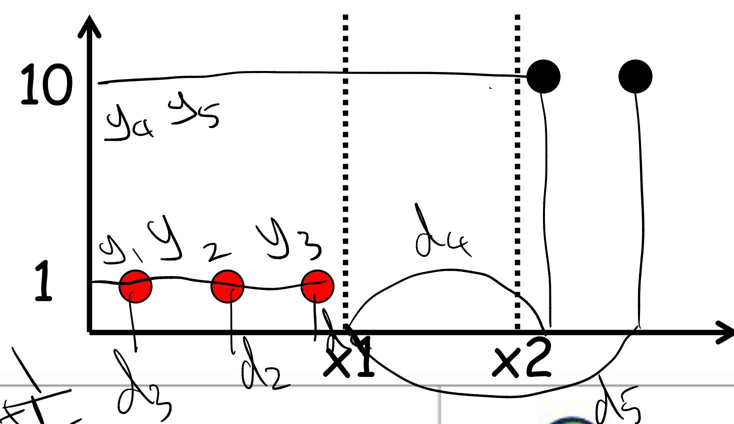
- **Classification (k=5)**
  - X1: Red or Black? Red?, Black?,
  - X2: Red or Black? Red



- **Regression (k=5)**
  - X1: close to 10 or 1
  - X2: close to 10 or 1

x_1의 y값: $\frac{y_1+y_2+y_3+y_4+y_5}{5}$

$= \frac{\frac{y_1}{d_1} + \frac{y_2}{d_2} + \cdots + \frac{y_5}{d_5}}{\frac{1}{d_1} + \frac{1}{d_2} + \cdots + \frac{1}{d_5}}$



Variation

- **More weight to closer one**
  - Different weight depending on the distance from **x'**
- **Classification: Not just counting**

$$S(\mathbf{x}',R)=\sum_{\mathbf{x}\in N(\mathbf{x}',R)} w(\mathbf{x})$$

$$S(\mathbf{x}',B)=\sum_{\mathbf{x}\in N(\mathbf{x}',B)} w(\mathbf{x})$$

if $S(\mathbf{x}',R) > S(\mathbf{x}',B)$ then $\mathbf{x}'$ is R

else $\mathbf{x}'$ is B

$N(\mathbf{x}', R)$: the set of *Red* data among the nearest neighbors of $\mathbf{x}'$

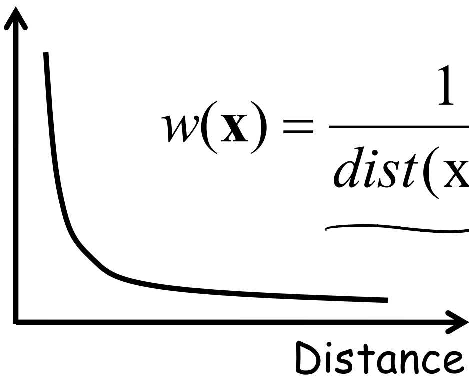
- **Regression: Not just averaging**

$$f(\mathbf{x}')=\frac{\sum_{\mathbf{x}\in N(\mathbf{x}')} w(\mathbf{x})\cdot f(\mathbf{x})}{\sum_{\mathbf{x}\in N(\mathbf{x}')} w(\mathbf{x})}$$

$N(\mathbf{x}')$: the nearest neighbors of $\mathbf{x}'$

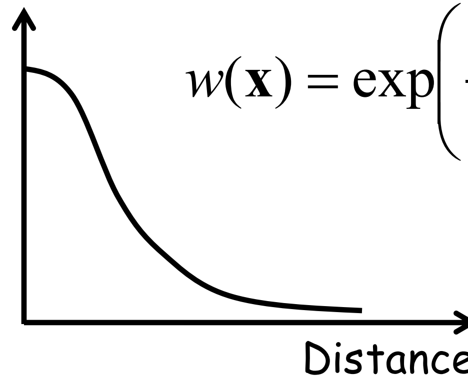
Variation

- **How to determine weight considering distance**



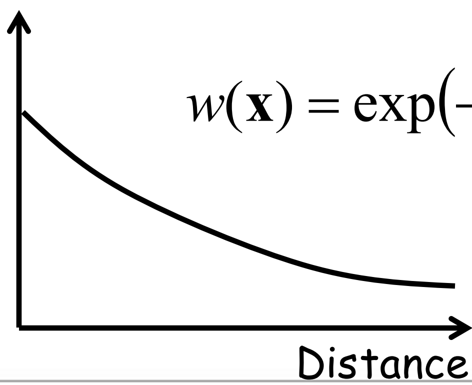
$$w(\mathbf{x}) = \frac{1}{dist(\mathrm{x}, \mathrm{x}')}$$

d_1



$$w(\mathbf{x}) = \exp\left(-\frac{dist(\mathrm{x}, \mathrm{x}')^2}{2\lambda^2}\right)$$

$-\frac{d_1^2}{2\lambda^2}$



$$w(\mathbf{x}) = \exp(-dist(\mathrm{x}, \mathrm{x}'))$$

$-d_1$

Do we need to choose k NNs?
-Yes, if you want
-Not necessarily

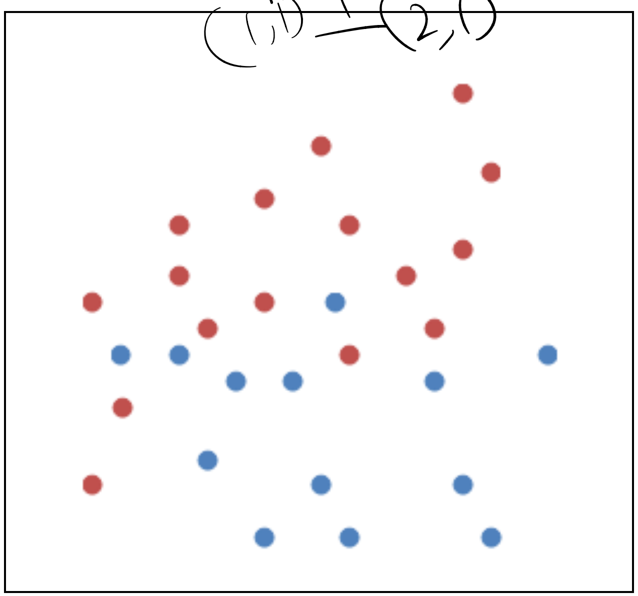
Distance Measure

- **Two data sets**
  - Compare the boundaries in D1 and D2 (1-NN)

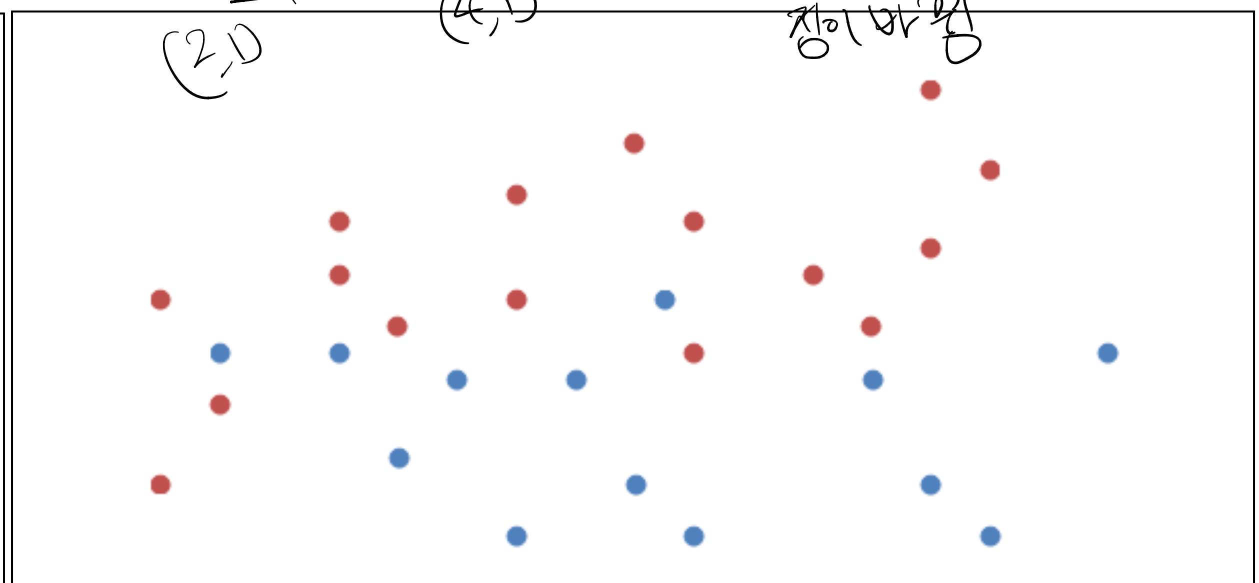
(1,2)
1
(1,1) 1 (2,1)

(2,1)
1 2
(2,1) (4,1)

늘리면 nearest 한 점이 바뀜



D1

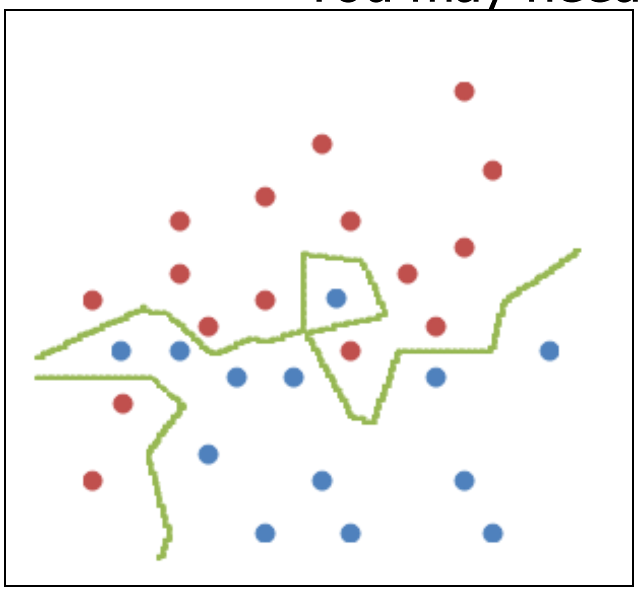


D2

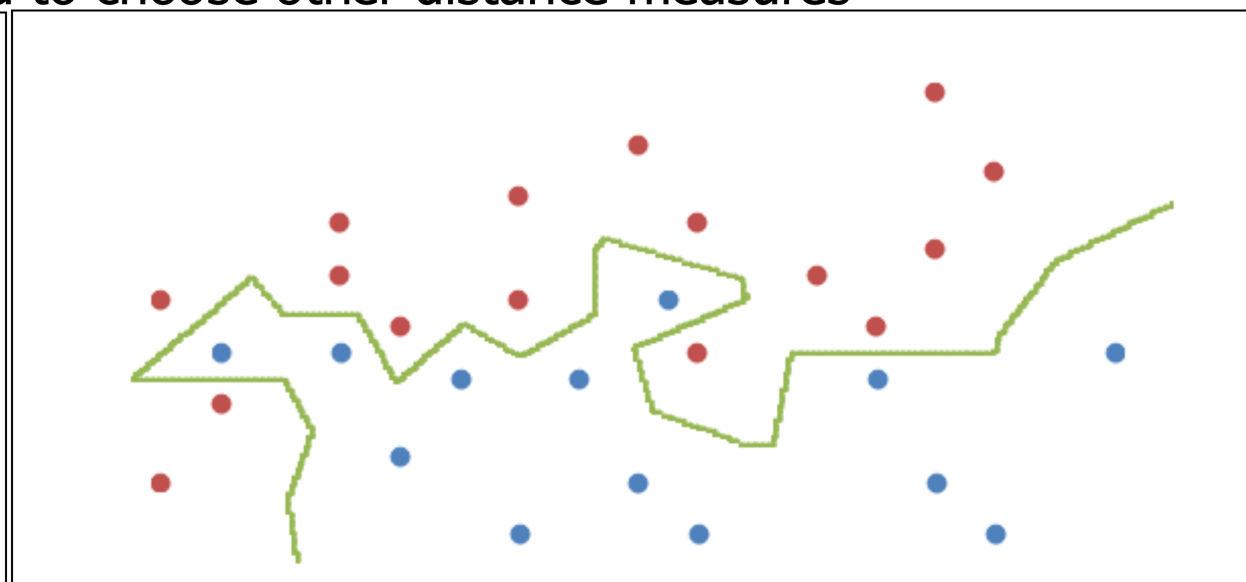
$$\underbrace{D2}_{\text{D의 x축 x2배}} = \{(x, y) \mid x = 2x', (x', y) \in D1\}$$

Distance Measure

- **Two data sets**
  - Even though the data are linearly scaled, the boundary changes!! 변화한다
  - You may need to choose other distance measures



D1



D2 $D2 = \{(x, y) \mid x = 2x', (x', y) \in D1\}$

Summary

- **Which k is better?**
  - Small k : higher variance (less stable) -> possibly overfitted
  - Large k : higher bias (less precise) -> possibly underfited

- **Proper choice of k**
  - Depending on the data
  - Use Cross-validation ; k의 좋은 밸류 선택가능

Summary

- **Advantage** of K-NN
  - No training (Only inference step)
  - Complexity of target functions do not matter
  - No loss of information

- **Disadvantage**
  - Have to keep all data -> Memory space a lot of Time, space
  - Sensitive to noise
  - If training data is imbalanced, major class may dominate
  - Need to calculate the distance from all training data -> Time
    - Especially in high dimensional space, expensive

Summary

- **Reducing Computational Cost**
  - Finding k nearest neibhbors is expensive: O(nd)
  - Space partitioning
    - quad-tree, locality sensitive hashing, etc.
  - Preprocessing
    - Reduce dimensions: Remove less important features, Vector quantization
    - Reduce size of data: Sampling, Clustering

K-NN in Scikit-Learn

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

x,y = make_blobs(50, n_features=2, centers=2, random_state=0)
knn = KNeighborsClassifier(n_neighbors = 3, \
                           weights = 'uniform', \
                           metric = 'euclidean')
knn.fit(x, y)

test_data = np.array([[1.4, 0.2], [1.4,0.5],[0.9, 4.0], \
                      [-0.1, 3.0], [2.5, 0.1]])

prediction = knn.predict(test_data)
print(prediction)
```