딥러닝/클라우드

Chapter 5

Clustering, knn

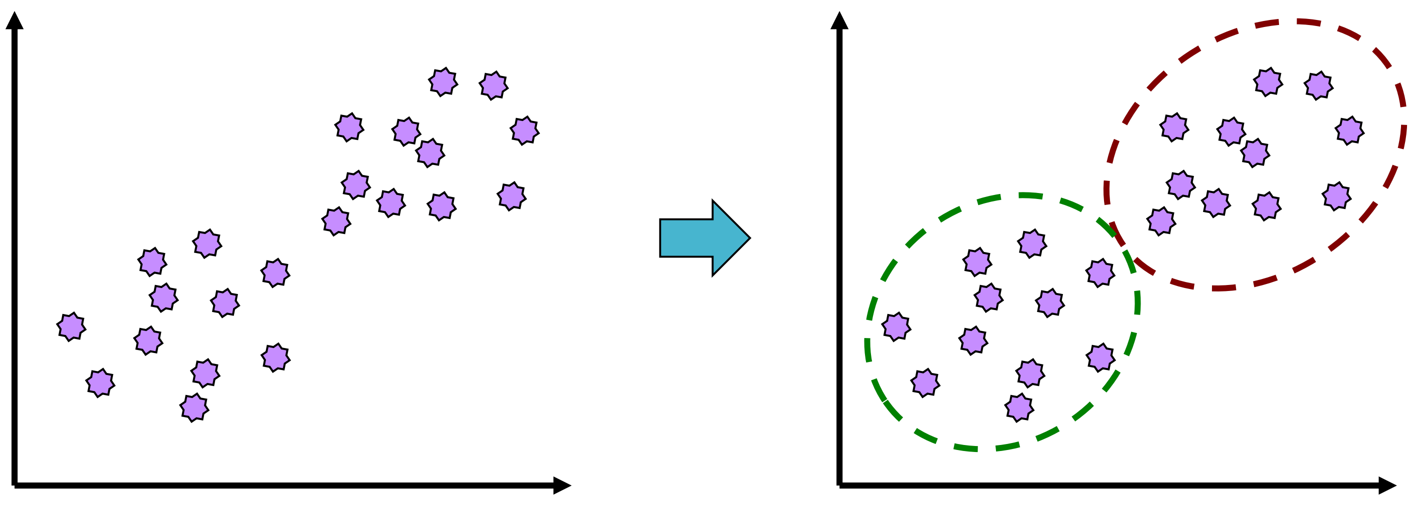
오 세 종

DANKOOK UNIVERSITY

Contents

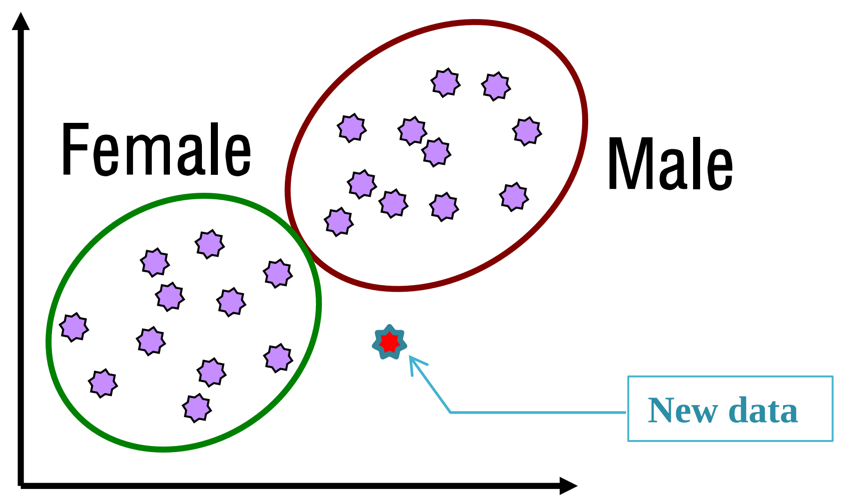
1. remind: clustering, classification

- Clustering
  - Grouping target data into some category (class)
  - Data in same group has similar characteristics
  - Group points into clusters based on how “near” they are to one another
  - Unsupervised learning



1. remind: clustering, classification

- Classification
  - Classify new data into one of known category.
  - The category has “label”
  - Application: prediction, diagnosis in medcine
  - Supervised learning



1. remind: clustering, classification

- Example : clustering
  - 차량의 특성을 가지고 grouping 을 해 보자

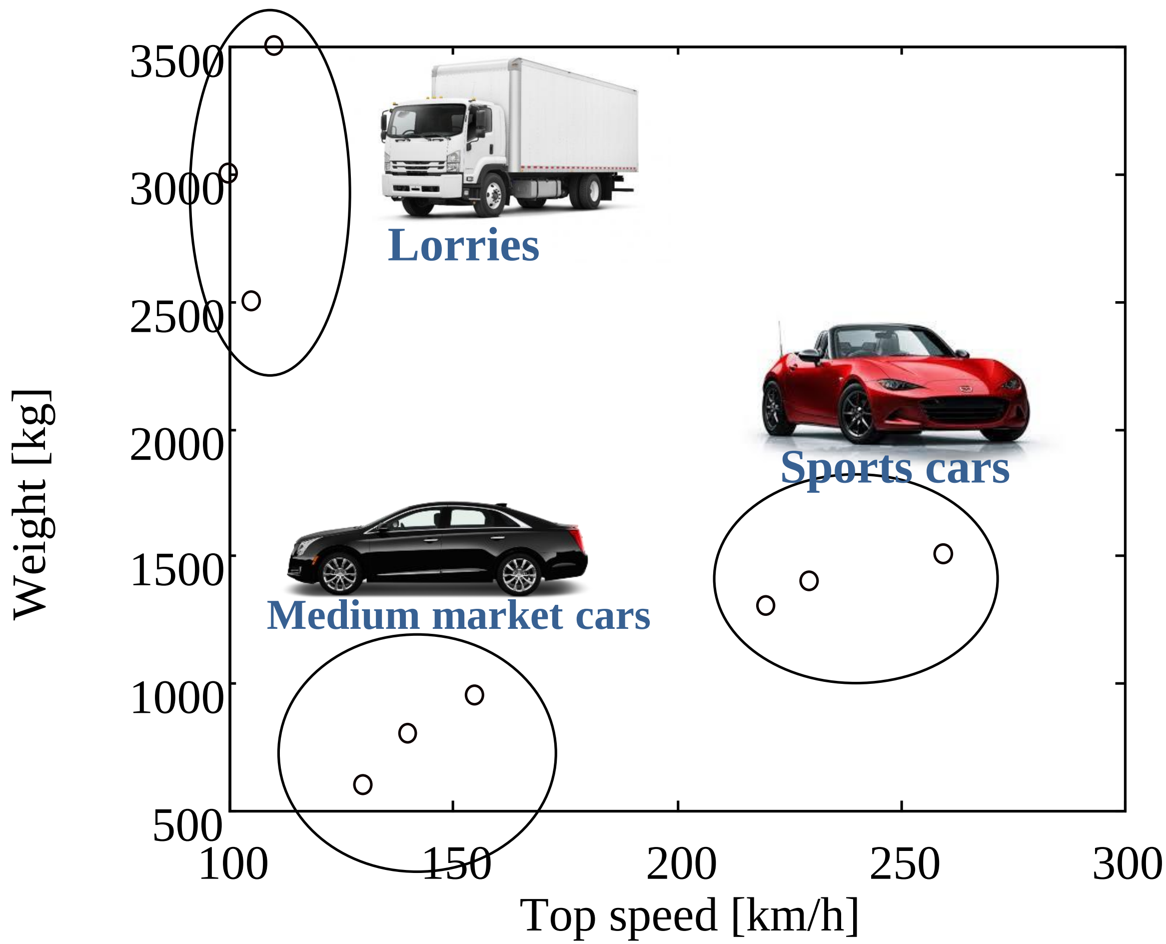
Can you see some group?

Vehicle	Top speed km/h	Color	Air resistance	Weight Kg
V1	220	red	0.30	1300
V2	230	black	0.32	1400
V3	260	red	0.29	1500
V4	140	gray	0.35	800
V5	155	blue	0.33	950
V6	130	white	0.40	600
V7	100	black	0.50	3000
V8	105	red	0.60	2500
V9	110	gray	0.55	3500

https://www.slideshare.net/picasso544/clustering-tutorial

1. remind: clustering, classification

- Example : clustering



1. remind: clustering, classification

- Example : classification

No	Height	Weight	running hour	working hour	Category
1	0.41	0.36	0.27	0.65	Patient
2	0.23	0.37	0.34	0.68	patient
3	0.38	0.38	0.46	0.95	patient
4	0.45	0.31	0.37	0.75	patient
5	0.37	0.45	0.48	0.75	patient
6	0.28	0.26	0.36	0.86	patient
7	0.66	0.44	0.51	0.98	patient
8	0.55	0.43	0.43	0.91	patient
9	0.23	0.44	0.28	0.78	patient
10	0.41	0.53	0.46	0.86	patient
11	0.65	0.38	0.74	0.51	normal
12	0.89	0.53	0.67	0.46	normal
13	0.58	0.54	0.56	0.43	normal
14	0.78	0.55	0.67	0.34	normal
15	0.89	0.56	0.81	0.56	normal
16	0.65	0.57	0.81	0.43	normal
17	0.75	0.67	0.76	0.35	normal
18	0.46	0.48	0.65	0.42	normal
19	0.89	0.69	0.78	0.23	normal
20	0.78	0.81	0.88	0.26	normal

Disease A

Height	Weight	running hour	working hour
0.5	0.44	0.45	0.61

Patient or Normal ?

1. remind: clustering, classification

- example: image classification

(1) take a picture by phone camera

Apple iPhone

Tokyo tower

Empire state building

Big Ben

(2) Search similar image and shows detail information about it

1. remind: clustering, classification

- Binary vs. multiple classification

- Binary classification
  - # of class is two

Male	Female
Patient	Normal
Yes	No

- multiple classification
  - # of class over two

Well-done	medium	rare

university	High school
Middle school	Elementary school

2. k-means clustering

- 금이간 타일과 정상 타일 군집화

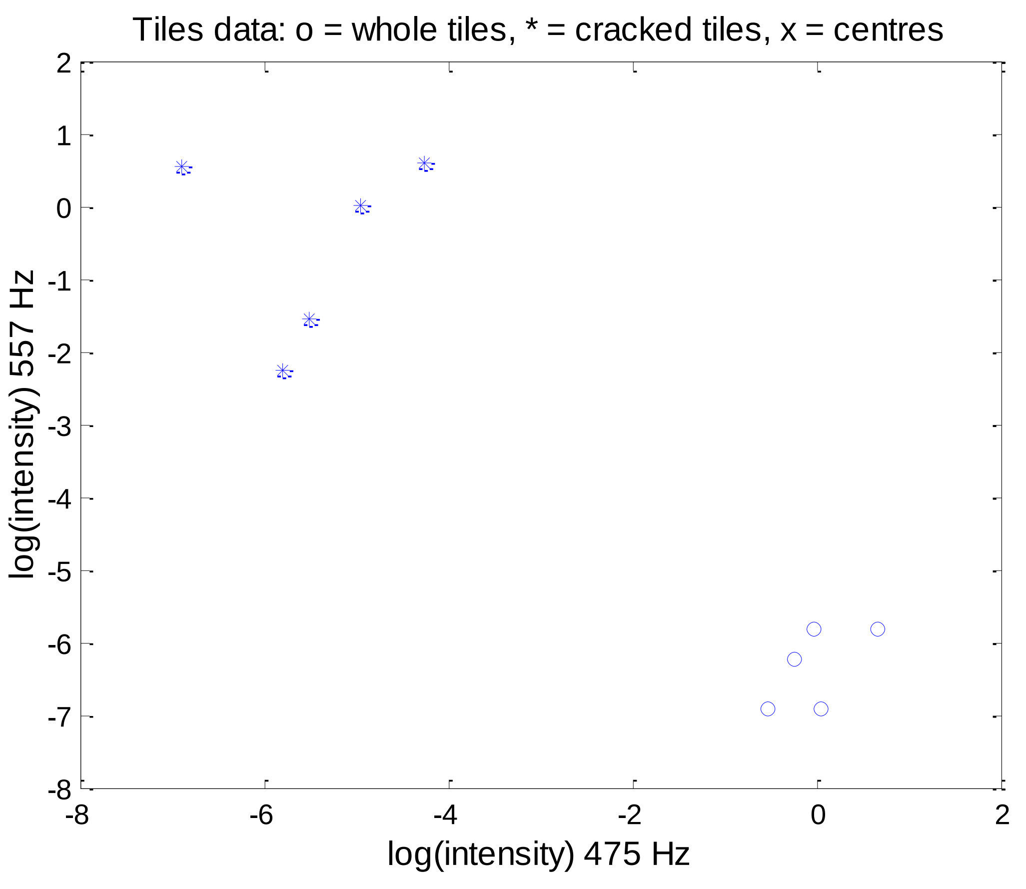
https://www.slideshare.net/picasso544/clustering-tutorial

475Hz	557Hz
0.958	0.003
1.043	0.001
1.907	0.003
0.780	0.002
0.579	0.001
0.003	0.105
0.001	1.748
0.014	1.839
0.007	1.021
0.004	0.214

Table 1: frequency intensities for ten tiles.

Tiles are made from clay moulded into the right shape, brushed, glazed, and baked. Unfortunately, the <u>baking may produce invisible cracks</u>. Operators can detect the cracks by hitting the tiles with a hammer, and in an automated system the response is recorded with a microphone, filtered, Fourier transformed, and normalised. A small set of data is given in TABLE 1 (adapted from MIT, 1997).

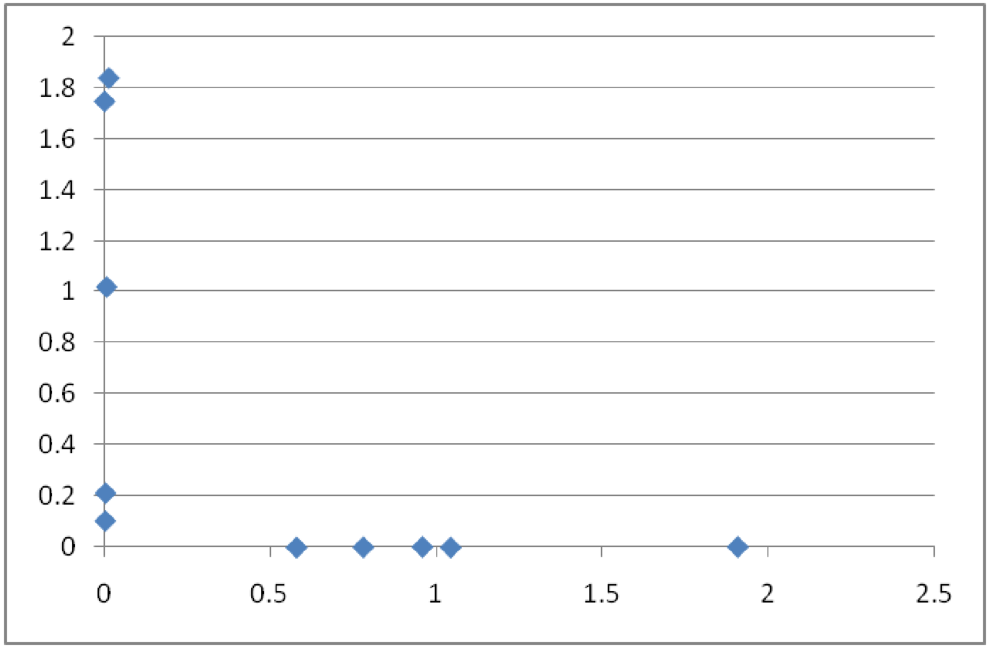
2. k-means clustering



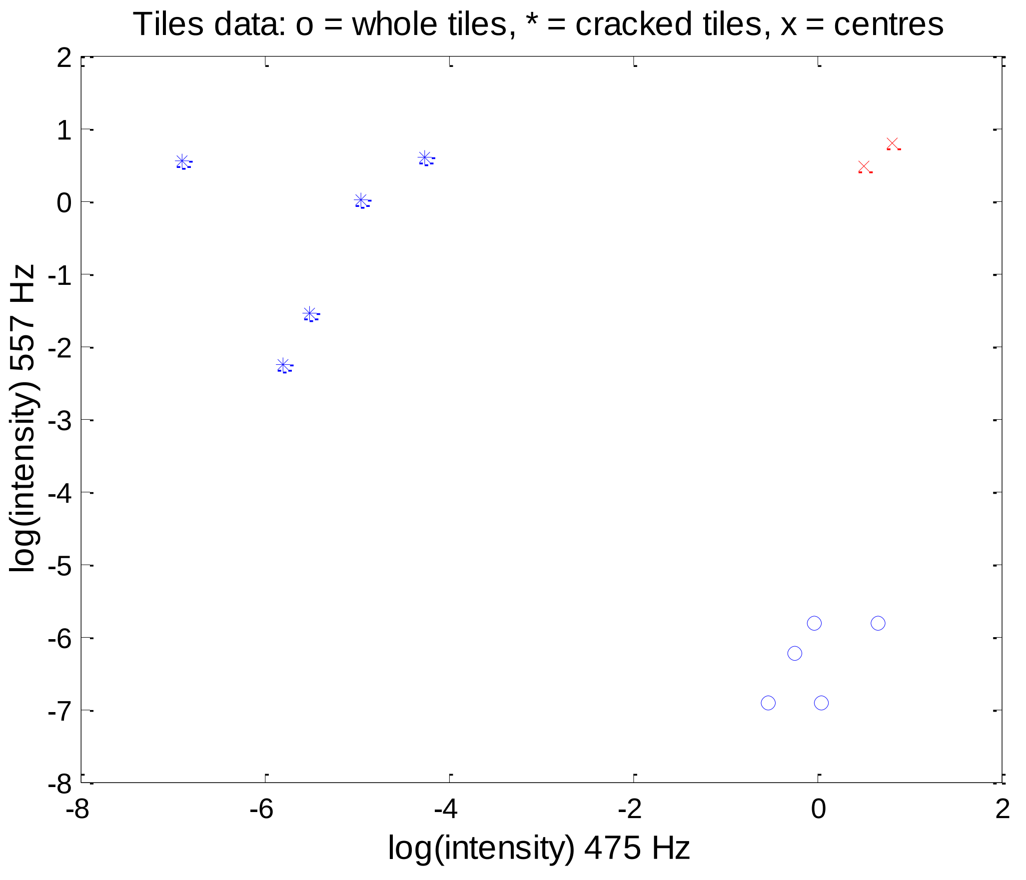
Plot of tiles by frequencies (logarithms). The whole tiles (o) seem well separated from the cracked tiles (*). The **objective** is to find the two clusters.

2. k-means clustering

- Before logarithms

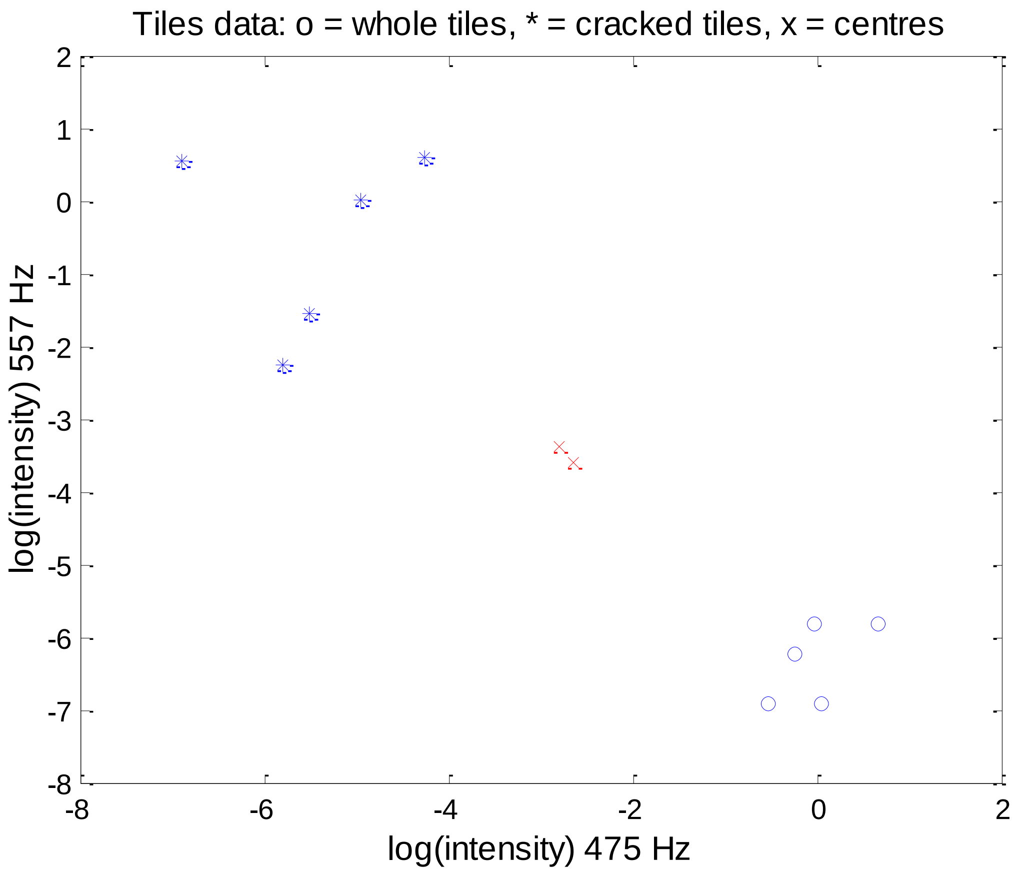


2. k-means clustering



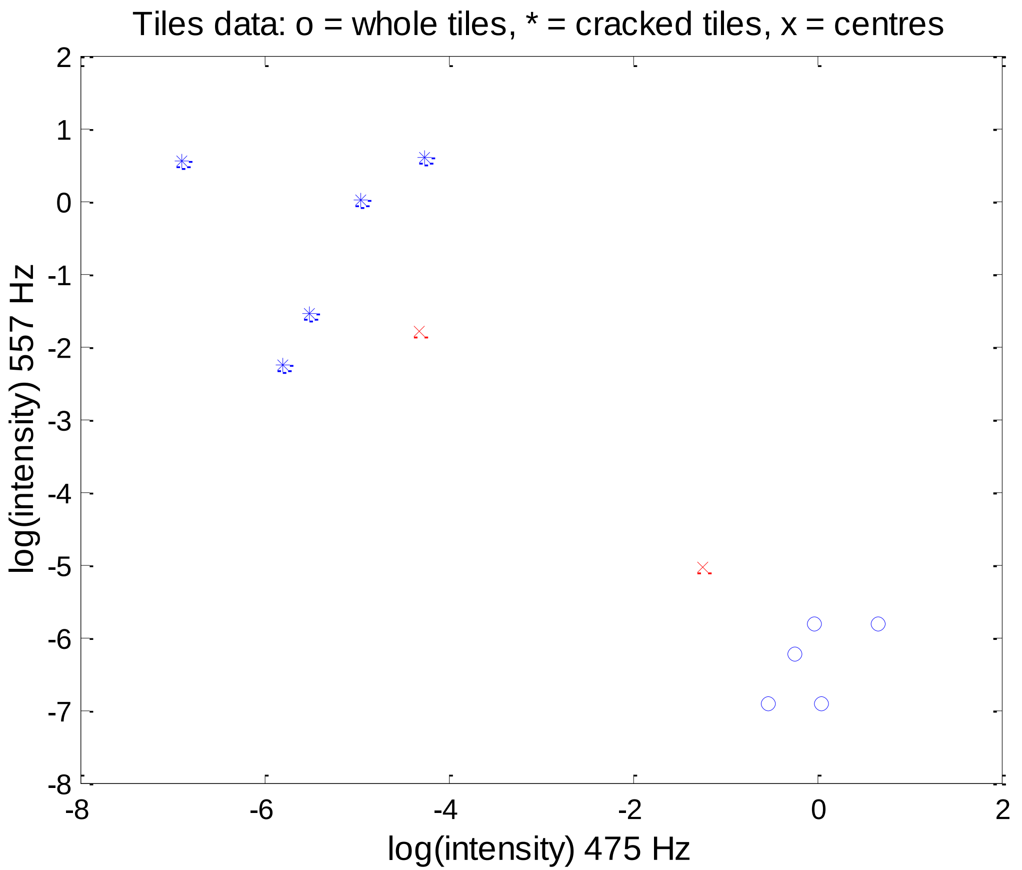
1. Place two cluster centres (x) at random.
2. Assign each data point (* and o) to the nearest cluster centre (x)

2. k-means clustering



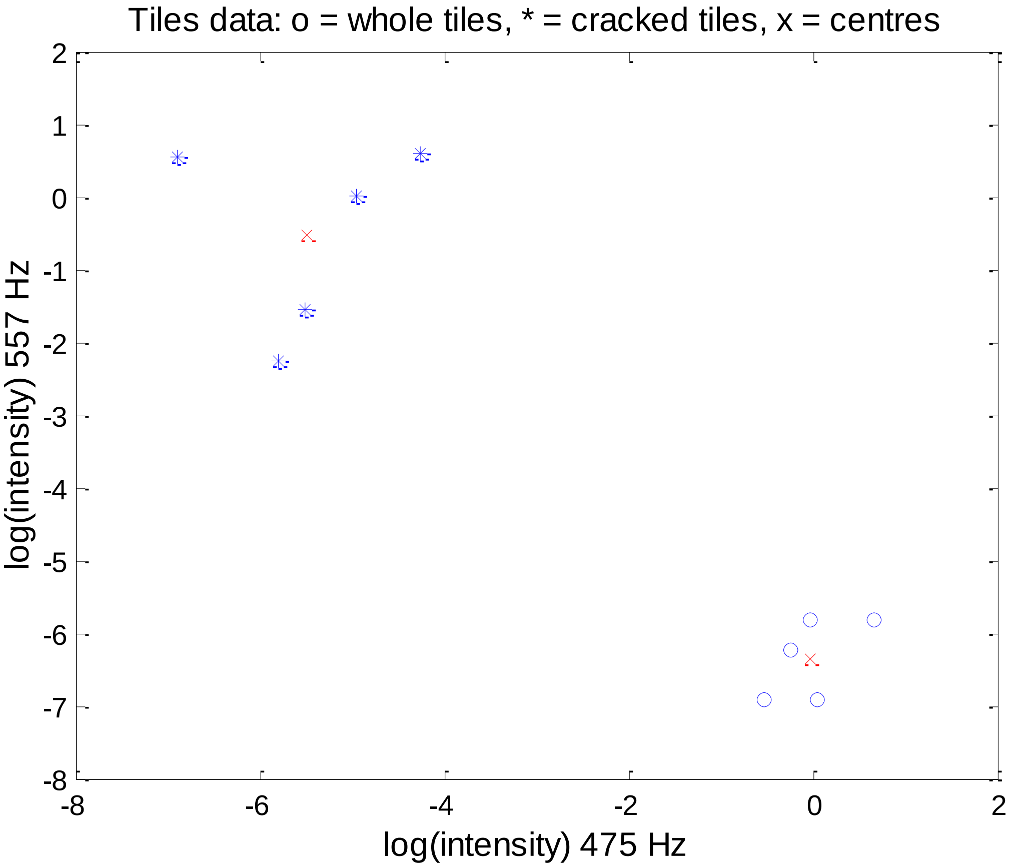
1. Compute the new centre of each class
2. Move the crosses (x)

2. k-means clustering



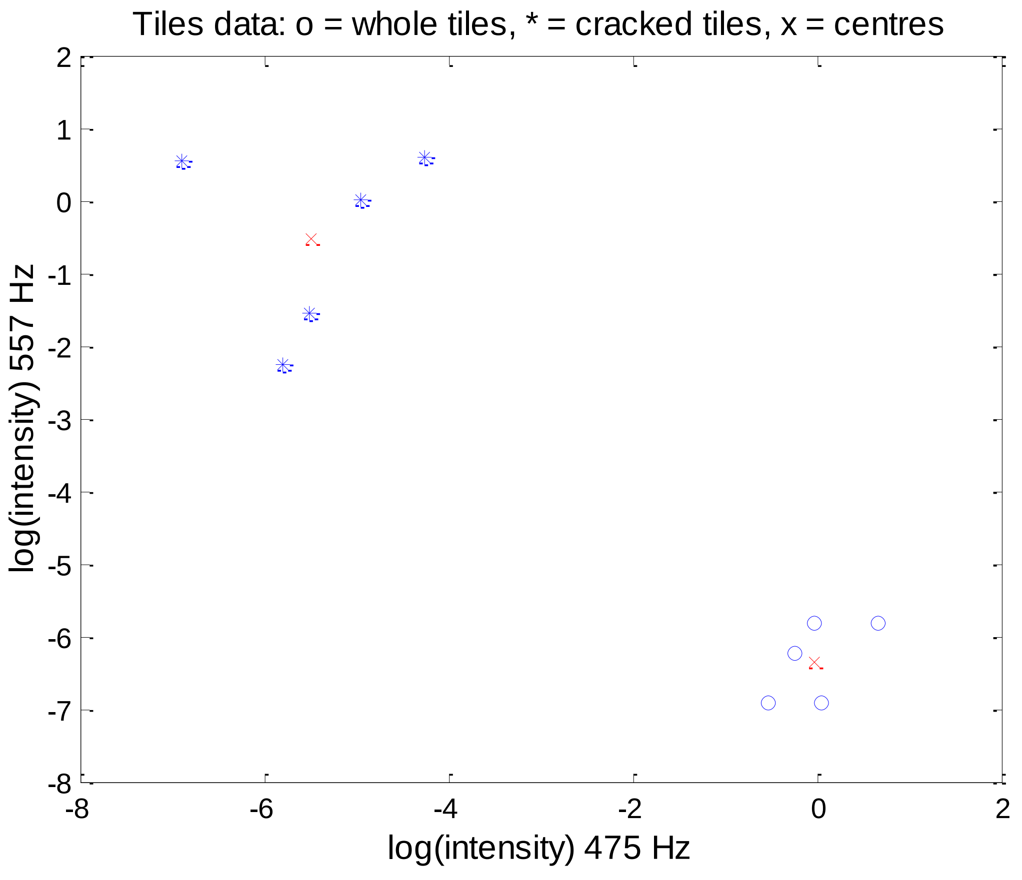
Iteration 2

2. k-means clustering



Iteration 3

2. k-means clustering



Iteration 4 (then stop, because no visible change)
Each data point belongs to the cluster defined by the nearest centre

2. k-means clustering

475Hz	557Hz	Result =
0.958	0.003	1
1.043	0.001	1
1.907	0.003	1
0.780	0.002	1
0.579	0.001	1
0.003	0.105	2
0.001	1.748	2
0.014	1.839	2
0.007	1.021	2
0.004	0.214	2

군집화 결과 :

1. The last five data points (rows) belong to the first cluster
2. The first five data points (rows) belong to the second cluster

2. k-means clustering

- 거리계산

$p = (p_1, p_2, p_3, \ldots, p_n)$, $q = (q_1, q_2, q_3, \ldots, q_n)$

Euclidean distance

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

2. k-means clustering

- Python code

kmeans_example.py

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

X = np.array([[1, 2], [4, 3], [2, 5],
              [8, 5], [10, 6], [9, 4]])

kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
```

```
In [313]: X
Out[313]:
array([[ 1,  2],
       [ 2,  3],
       [ 2,  5],
       [ 8,  5],
       [10,  6],
       [ 9,  4]])
```

KMeans

n_clusters   : 클러스터의 개수
random_state : seed for reproducability

2. k-means clustering

```
# cluster label
kmeans.labels_
# bind data & cluster label
np.hstack((X, kmeans.labels_.reshape(-1, 1)))

# center of clusters
kmeans.cluster_centers_

# predict new data
kmeans.predict([[0, 0], [12, 3]])
```

```
In [309]: kmeans.labels_
Out[309]: array([1, 1, 1, 0, 0, 0])

In [310]: np.hstack((X, kmeans.labels_.reshape(-1, 1)
Out[310]:
array([[ 1,  2,  1],
       [ 2,  3,  1],
       [ 2,  5,  1],
       [ 8,  5,  0],
       [10,  6,  0],
       [ 9,  4,  0]])
```

```
In [311]: kmeans.cluster_centers_
Out[311]:
array([[9.        , 5.        ],
       [1.66666667, 3.33333333]])

In [312]: kmeans.predict([[0, 0], [12, 3]])
Out[312]: array([1, 0])
```

3. KNN classifier

- 분류(classification)

No	running hour	working hour	Category
1	0.27	0.65	Patient
2	0.34	0.68	patient
3	0.46	0.95	patient
4	0.37	0.75	patient
5	0.48	0.75	patient
6	0.36	0.86	patient
7	0.51	0.98	patient
8	0.43	0.91	patient
9	0.28	0.78	patient
10	0.46	0.86	patient
11	0.74	0.51	normal
12	0.67	0.46	normal
13	0.56	0.43	normal
14	0.67	0.34	normal
15	0.81	0.56	normal
16	0.81	0.43	normal
17	0.76	0.35	normal
18	0.65	0.42	normal
19	0.78	0.23	normal
20	0.88	0.26	normal

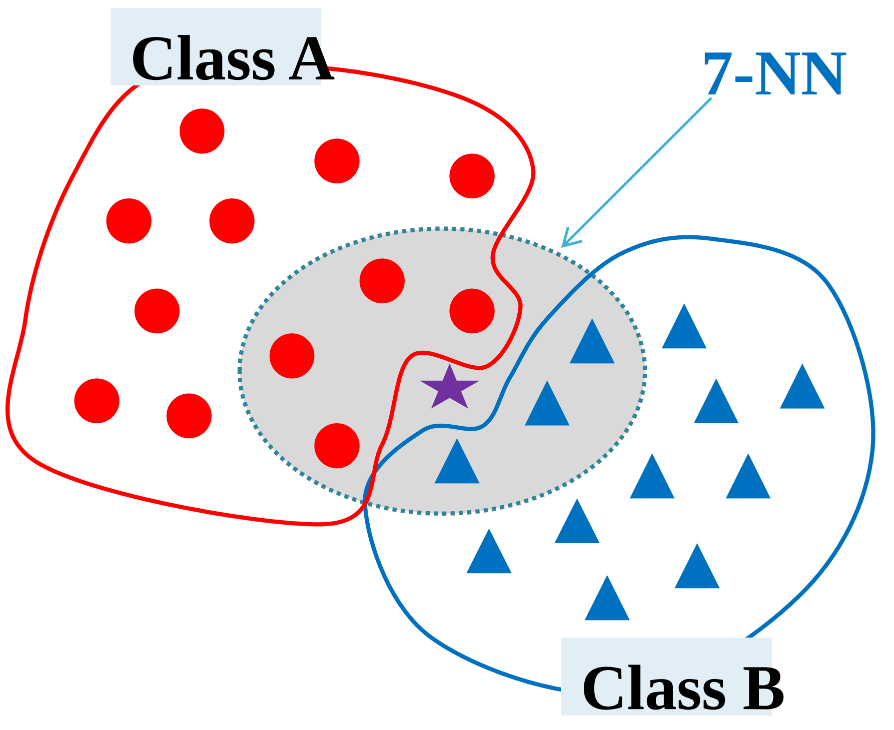
Given Classified Data

Patient or Normal ?

running hour	working hour
0.45	0.61

3. KNN classifier

- Idea of KNN
  - Find K nearest neighbor for new point (★)
  - Decide new point belongs to major class (class A)
    - # of neighbor of Class A > # of neighbor of Class B

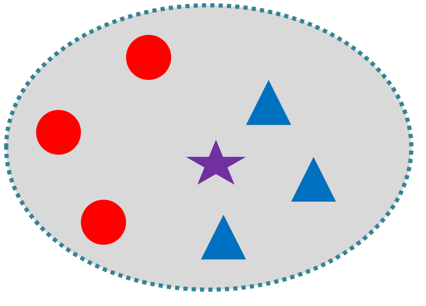


3. KNN classifier

- Algorithm
  - Calculate distance between new point and every point of given classes
  - Choose K nearest points by the distance
  - Choose major class from K points

    (the class is for the new point)

6-NN



???

3. KNN classifier

- How to calculate the distance between two element ?
  - Using Euclidean distance

$$\mathbf{p} = (p_1, p_2, \ldots, p_n)$$
$$\mathbf{q} = (q_1, q_2, \ldots, q_n)$$

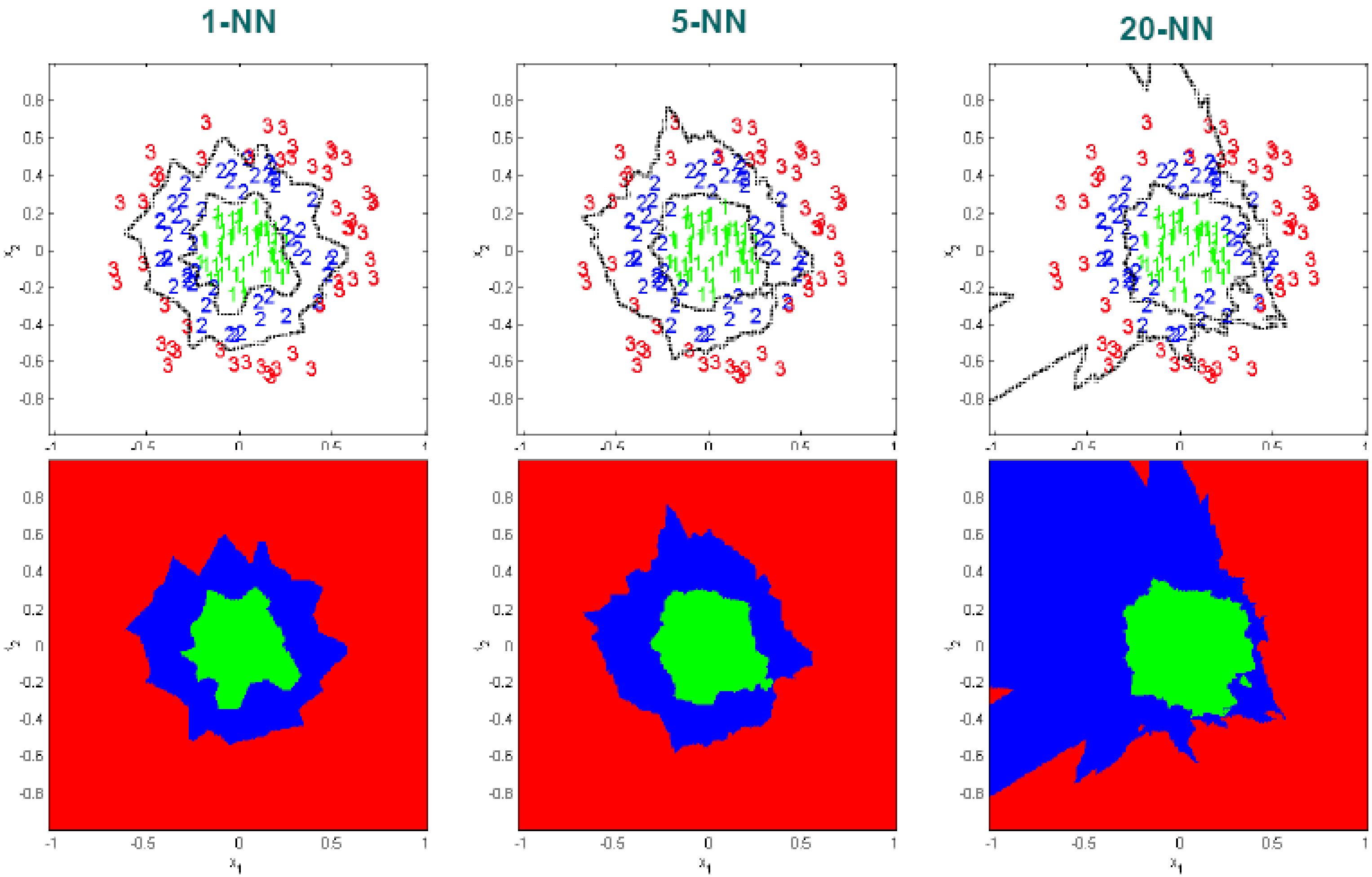
$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

3. KNN classifier

- K 를 얼마로 하는 것이 좋은가
  - 크게 할 때와 작게 할 때 각각 장단점이 있다
  - 데이터 수가 N 이라고 할 때 K < sqrt(N) 을 권장

3. KNN classifier

- 1NN vs kNN



3. KNN classifier

- 장점
  - 통계적 가정 불필요 (비모수적 방법)
  - 단순하다
  - 성능이 좋다
  - 모델을 훈련(학습)하는 시간이 필요 없다

- 단점
  - 데이터가 커질수록 많은 메모리 필요, 처리시간(분류시간) 증가

3. KNN classifier

- Python code

knn_basic.py

```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the iris dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)
print(iris_X.shape)   # (150, 4)

# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(iris_X, iris_y, test_size=0.3,\
                     random_state=1234)
```

3. KNN classifier

```
# Define learning model
model =  KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(train_X, train_y)

# Make predictions using the testing set
pred_y = model.predict(test_X)
print(pred_y)

# model evaluation: accuracy #############

acc = accuracy_score(test_y, pred_y)
print('Accuracy : {0:3f}'.format(acc))
```

```
Accuracy : 0.977778
```

3. KNN classifier

- `KNeighborsClassifier`

Hyper parameter (조율모수)

  - `n_neighbors : 이웃의 개수 (default: 5)`

3. KNN classifier

- Dataset scaling
  - Kmeans, knn 과 같은 거리기반 학습방법을 적용할 때는 scaling 필요
  - Scaling: 변수들의 값의 범위를 일정하게 맞추는 과정
  - 이유: 거리 계산에서 scale 이 큰 변수가 작은 변수보다 더 영향을 미치기 때문에 변수별 영향력을 동일하게 해야 한다.
  - Ex) 키(150~200), 시력(0.1~2.0)

(155, 1.0)   (150, 0.1)

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

  - python

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)                    # X : input data
X_scaled = scaler.transform(X)
```

3. KNN classifier

- Dataset scaling

```
In [215]: iris_X
Out[215]:
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
```

```
In [220]: scaler.transform(iris_X)
Out[220]:
array([[-9.00681170e-01,  1.01900435e+00, -1.34022653e+00,
        -1.31544430e+00],
       [-1.14301691e+00, -1.31979479e-01, -1.34022653e+00,
        -1.31544430e+00],
       [-1.38535265e+00,  3.28414053e-01, -1.39706395e+00,
        -1.31544430e+00],
       [-1.50652052e+00,  9.82172869e-02, -1.28338910e+00,
        -1.31544430e+00],
       [-1.02184904e+00,  1.24920112e+00, -1.34022653e+00,
        -1.31544430e+00],
       [-5.37177559e-01,  1.93979142e+00, -1.16971425e+00,
        -1.05217993e+00],
       [-1.50652052e+00,  7.88807586e-01, -1.34022653e+00,
        -1.18381211e+00],
```

scikit-learn에서는 다음과 같은 스케일링 클래스를 제공한다.

- `StandardScaler(X)`: 평균이 0과 표준편차가 1이 되도록 변환.
- `RobustScaler(X)`: 중앙값(median)이 0, IQR(interquartile range)이 1이 되도록 변환.
- `MinMaxScaler(X)`: 최대값이 각각 1, 최소값이 0이 되도록 변환
- `MaxAbsScaler(X)`: 0을 기준으로 절대값이 가장 큰 수가 1또는 -1이 되도록 변환

4. Performance metric

- Performance metric
  - Performance evaluation of learning model (classification)

For binary classification model only

  - Sensitivity (recall)
  - Specificity
  - precision
  - F1 score
  - ROC, AUC

For all classification model

  - Accuracy

4. Performance metric

- Binary classification metric

positive (1) : 양성
negative (0) : 음성

Fact (실제값)

Predict (예측값)

	Fact is positive	Fact is negative
Predict as positive	TP	FP
Predict as negative	FN	TN

fact	predict
1	1
0	1
0	0
1	1
0	1
1	0

TP : true positive FP : false positive
FN : false negative TN : true negative

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

(정확도)

4. Performance metric

	Fact is True	Fact is False
Predict as True	TP	FP
Predict as False	FN	TN

- Binary Classification Error

Sensitivity = TP/(TP+FN)
(민감도, recall(재현율))

Specificity = TN/(TN+FP)
(특이도)

- Sensitivity
  - Fraction of all Class1 (True) that we correctly predicted at Class 1
  - *How good are we at finding what we are looking for*
- Specificity
  - Fraction of all Class 2 (False) called Class 2
  - *How many of the Class 2 do we filter out of our Class 1 predictions*

4. Performance metric

	Fact is True	Fact is False
Predict as True	TP	FP
Predict as False	FN	TN

- Binary Classification Error

Precision = TP/(TP+FP)

(정밀도)

환자(Positive), 정상인(Negative)

Sensitivity : 환자를 환자라고 예측한 비율
Specificity : 정상인을 정상인이라고 예측한 비율
Precision : 환자라고 예측한 것 중에서 실제 환자의 비율

어떤 평가기준이라도 값이 클수록 좋다!

4. Performance metric

- Binary classification metric
  - F1 score : harmonic mean of sensitivity and specificity
    - sensitivity 와 specificity 의 불균형에 대해 감점이 이루어짐

$$F1\ score = \frac{2 \times sensitivity \times specificity}{sensitivity + specificity}$$

sensitivity	specificity	F1 score
1	1	1
1	0	0
0.8	0.8	0.8
0.8	0.5	0.62

4. Performance metric

- How to calculate sensitivity, specificity,.. for multi-class model ?

class A, class B, class C

For class A:
positive : class A, negative: class B,C

For class B:
positive : class B, negative: class A,C

For class C:
positive : class C, negative: class A,B

4. Performance metric

- Python metric
  - https://scikit-learn.org/stable/modules/model_evaluation.html

Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	for imbalanced datasets.
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'jaccard' etc.	`metrics.jaccard_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
'roc_auc_ovr'	`metrics.roc_auc_score`	
'roc_auc_ovo'	`metrics.roc_auc_score`	
'roc_auc_ovr_weighted'	`metrics.roc_auc_score`	
'roc_auc_ovo_weighted'	`metrics.roc_auc_score`	

4. Performance metric

Clustering	
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score
'adjusted_rand_score'	metrics.adjusted_rand_score
'completeness_score'	metrics.completeness_score
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score
'homogeneity_score'	metrics.homogeneity_score
'mutual_info_score'	metrics.mutual_info_score
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score
'v_measure_score'	metrics.v_measure_score
Regression	
'explained_variance'	metrics.explained_variance_score
'max_error'	metrics.max_error
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_root_mean_squared_error'	metrics.mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
'r2'	metrics.r2_score
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance

4. Performance metric

- example

```
from sklearn.metrics import accuracy_score

test_y = [2, 0, 2, 2, 0, 1]
pred_y = [0, 0, 2, 2, 0, 2]

acc = accuracy_score(test_y, pred_y)
print(acc)
```

```
In [250]: print(acc)
0.6666666666666666
```

4. Performance metric

- Confusion matrix

```
from sklearn.metrics import confusion_matrix
test_y = [2, 0, 2, 2, 0, 1]
pred_y = [0, 0, 2, 2, 0, 2]
confusion_matrix(test_y, pred_y)
```

```
In [224]: confusion_matrix(test_y, pred_y)
Out[224]:
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
```

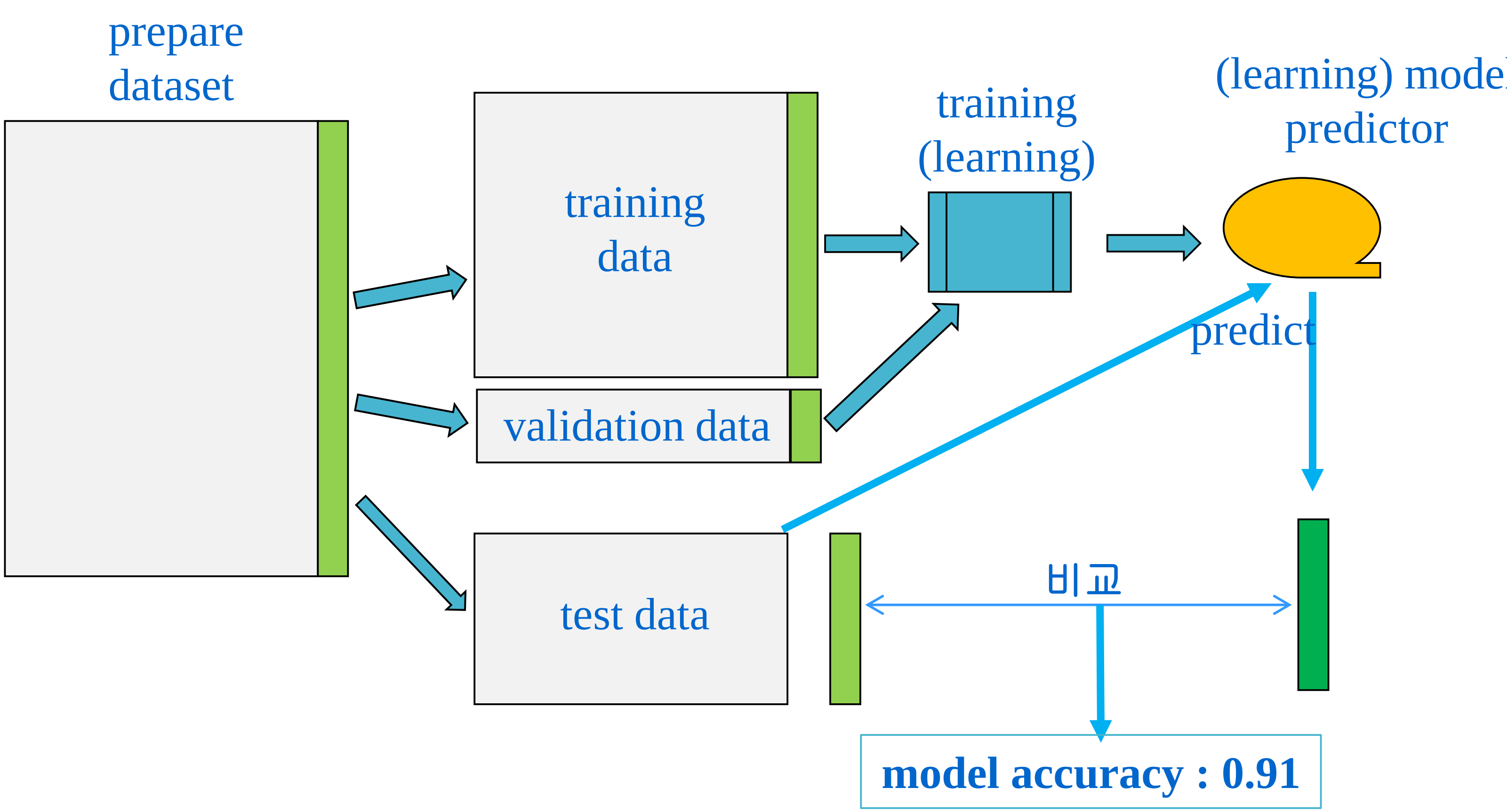
$$\begin{array}{cc} & \text{pred\_y} \\ & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \text{test\_y} \begin{array}{c} 0 \\ 1 \\ 2 \end{array} & \begin{array}{c} [2, 0, 0] \\ [0, 0, 1] \\ [1, 0, 2] \end{array} \end{array}$$

```
# binary classification
test_y = [1, 0, 0, 1, 0, 1]
pred_y = [0, 0, 0, 1, 0, 1]
tn, fp, fn, tp = confusion_matrix(test_y, pred_y).ravel()
(tn, fp, fn, tp)
```

```
In [243]: (tn, fp, fn, tp)
Out[243]: (3, 0, 1, 2)
```

5. K-fold Cross Validation

- motivation



Is it correct ?

5. K-fold Cross Validation

- Only one classification experiment is enough ?

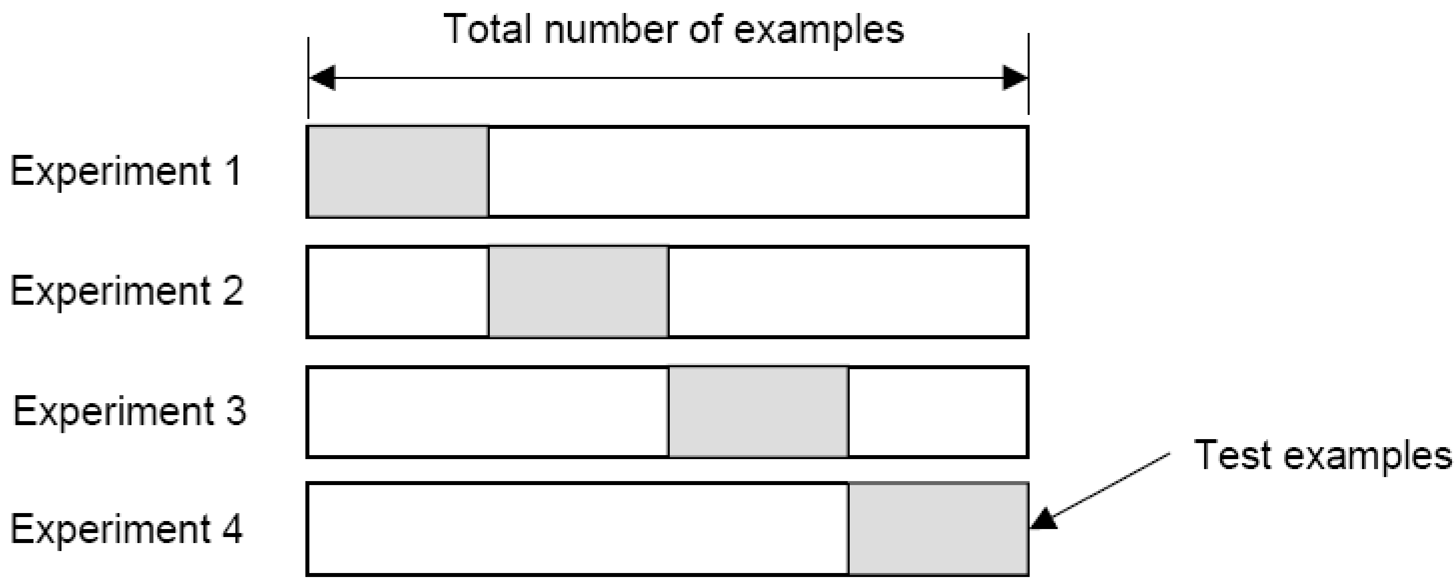
Training data	Test data

  - Classification accuracy = 0.87 (???)

- 위의 예에서 Test 데이터셋을 다르게 만들면 accuracy 가 달라질 것이다
- Test 데이터셋이 어떻게 구성되었는가에 따라 accuracy 가 원래 성능보다 높거나 낮게 나올 수도 있다.
- 그렇다면 어떻게 해야 분류 모델 또는 분류 알고리즘의 성능을 보다 정확히 알 수 있을까?

5. K-fold Cross Validation

- Create a K-fold partition of the dataset
  - For each of K experiments, use K-1 folds for training and the remaining one for testing (일반적으로 k=10 을 많이 사용)



  - 모델의 정확도는 각 fold 의 정확도들의 평균으로 계산

$$Acc = \frac{1}{K}\sum_{i=1}^{K} Acc_i$$

5. K-fold Cross Validation

- Python function

05.knn_kfold.py

```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np

# Load the iris dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)

# Define fold (5 fold)
kf = KFold(n_splits=5, random_state=123, shuffle=True)

# Define learning model
model =  KNeighborsClassifier(n_neighbors=3)

acc = np.zeros(5)      # 5 fold
i = 0                  # fold no
```

```
for train_index, test_index in kf.split(iris_X):
    print("fold:", i)

    train_X, test_X = iris_X[train_index], iris_X[test_index]
    train_y, test_y =  iris_y[train_index], iris_y[test_index]

    model.fit(train_X, train_y)

    pred_y = model.predict(test_X)

    # model evaluation: accuracy #############
    acc[i] = accuracy_score(test_y, pred_y)
    print('Accuracy : {0:3f}'.format(acc[i]))
    i += 1


print("5 fold :", acc)
print("mean accuracy :", np.mean(acc))
```

5. K-fold Cross Validation

```
fold: 0
Accuracy : 0.966667
fold: 1
Accuracy : 0.966667
fold: 2
Accuracy : 0.966667
fold: 3
Accuracy : 0.966667
fold: 4
Accuracy : 0.933333
5 fold : [0.96666667 0.96666667 0.96666667 0.96666667 0.93333333]
mean accuracy : 0.96
```

5. K-fold Cross Validation

- K-fold Cross Validation (simple way)

05.knn_cross_val_score.py

```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np

# Load the iris dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)

# Define learning model
model =  KNeighborsClassifier(n_neighbors=3)

# Define fold (model, train, target, cross validation)
scores = cross_val_score(model, iris_X, iris_y, cv=5)

print("fold acc", scores)
print("mean acc", np.mean(scores))
```

5. K-fold Cross Validation

```
In [360]: print("fold acc", scores)
fold acc [0.96666667 0.96666667 0.93333333 0.96666667 1.         ]

In [361]: print("mean acc", np.mean(scores))
mean acc 0.9666666666666668
```

Note. K-fold cross validation 의 용도

- K-fold cross validation이 원하는 모델을 도출하지는 않음
- 주어진 데이터셋으로 모델 개발시 미래의 정확도를 추정
- 최종 모델 개발을 위한 hyper parameter 튜닝에 사용
- 전처리시 feature selection 에 사용

- K-fold cross validation 에 의해 최적의 hyper parameter 값을 확정하면 전체 데이터를 활용하여 최종 모델을 완성함