Machine learning 05

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

Contents

- Unsupervised learning

Unsupervised learning

- Supervised or not
  - unsupervised learning
    - no label on the training data
    - the system must learn without any help

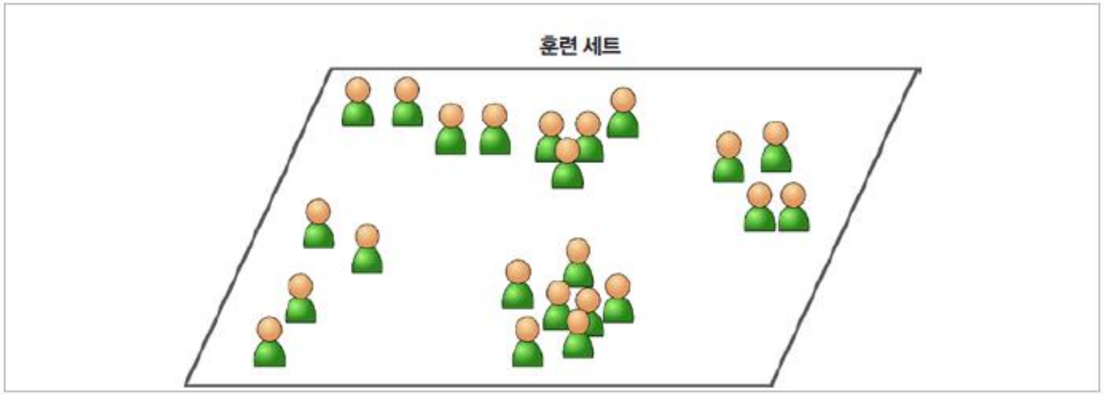


그림 1-7 비지도 학습에서 레이블 없는 훈련 세트

Unsupervised learning

- Application examples
  - clustering
  - outlier detection
  - density estimation

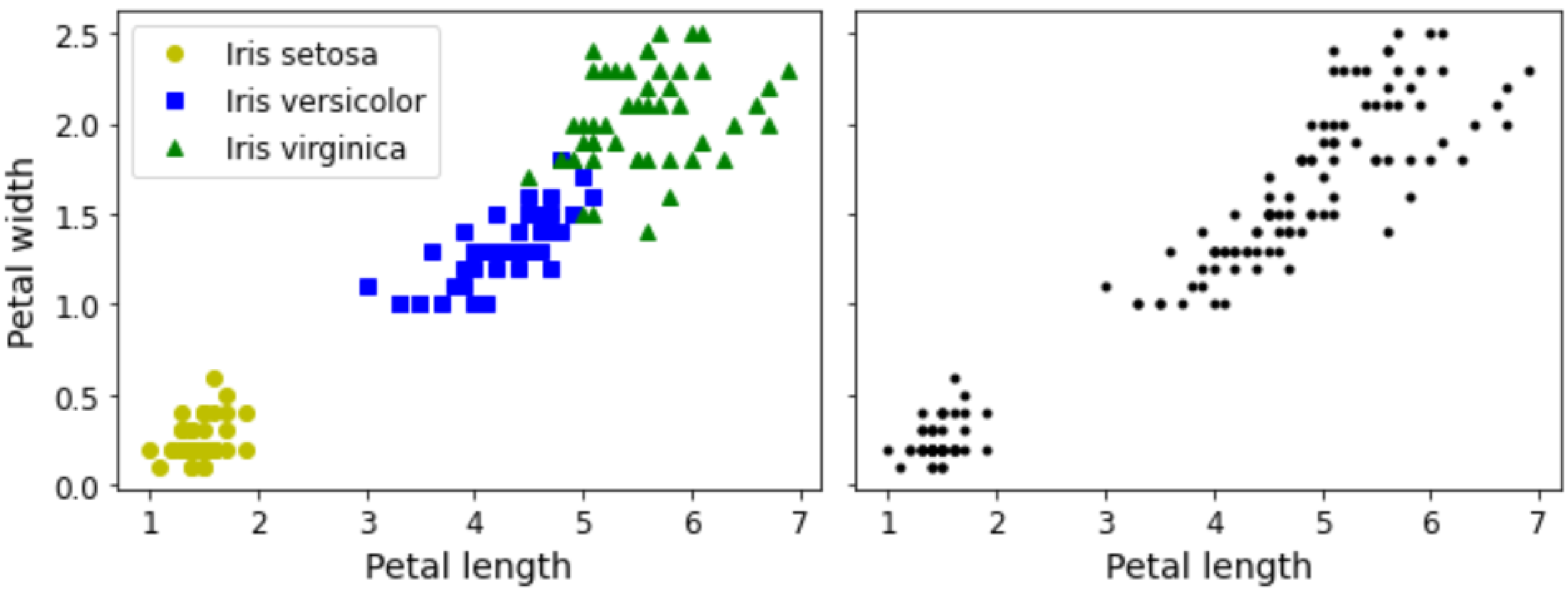
Gyeongsang National University 경상국립대학교

Clustering

- Definition
  - the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters)

Clustering

- Classification vs clustering



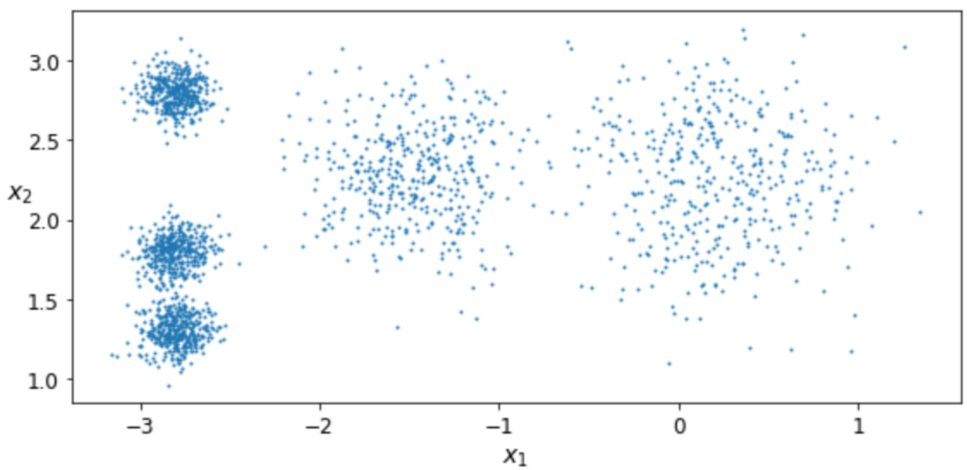
Clustering

- Applications
  - customer categorization
  - dimension reduction technique
  - outlier detection
  - semi-supervised learning
  - image segmentation

GNU Gyeongsang National University 경상국립대학교

K-means clustering

- Lloyd-Forgy algorithm
  - simple algorithm that quickly and efficiently generate clusters from unlabeled datasets in a few iterations



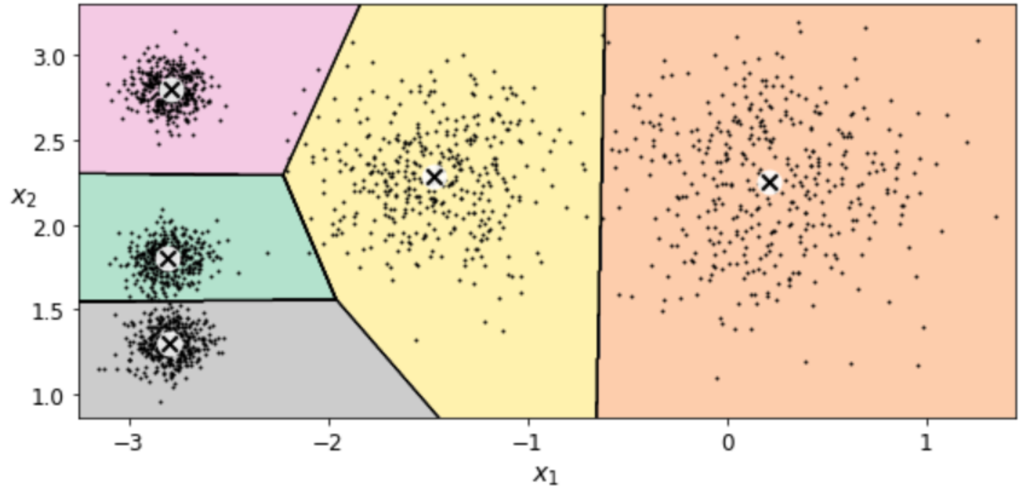
Gyeongsang National University 경상국립대학교

K-means clustering

- Example in scikit-learn
  - Voronoi tessellation

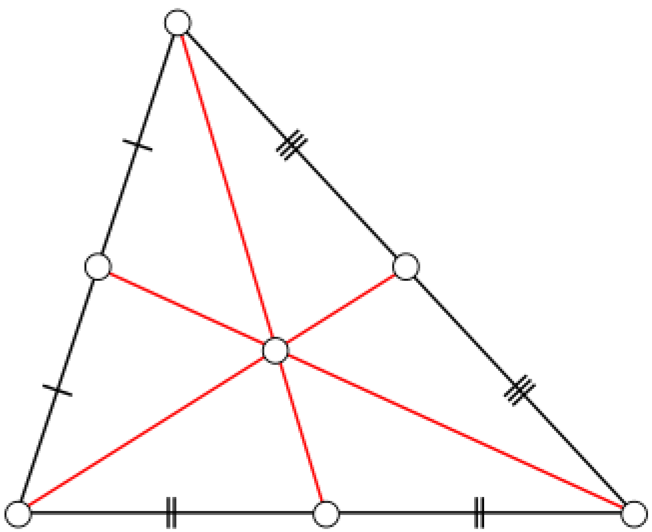
```
from sklearn.cluster import KMeans
```

```
k = 5
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X)
```



K-means clustering

- Centroid
  - the arithmetic mean position of all the points in the figure

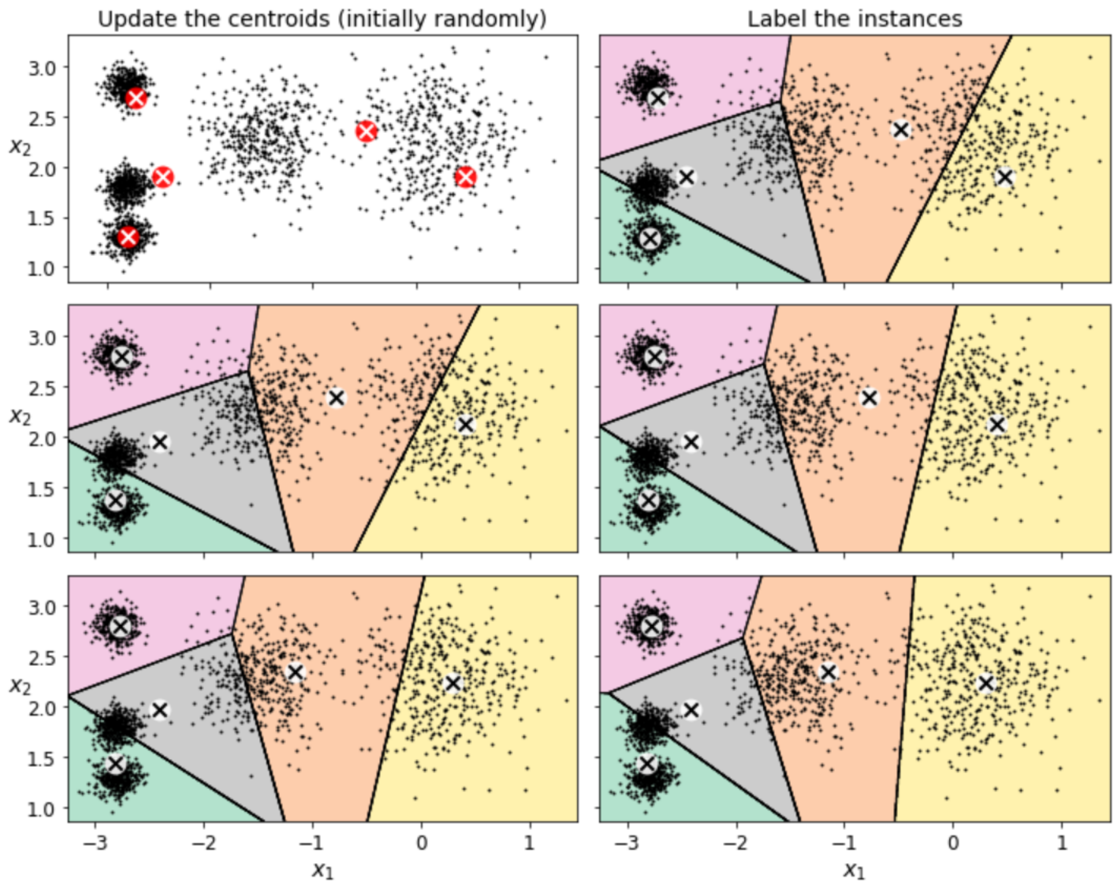


K-means clustering

- Procedure
    - randomly choose k centroids
    - calculate distance from k centroids
    - update centroids
    - calculate distance from k centroids

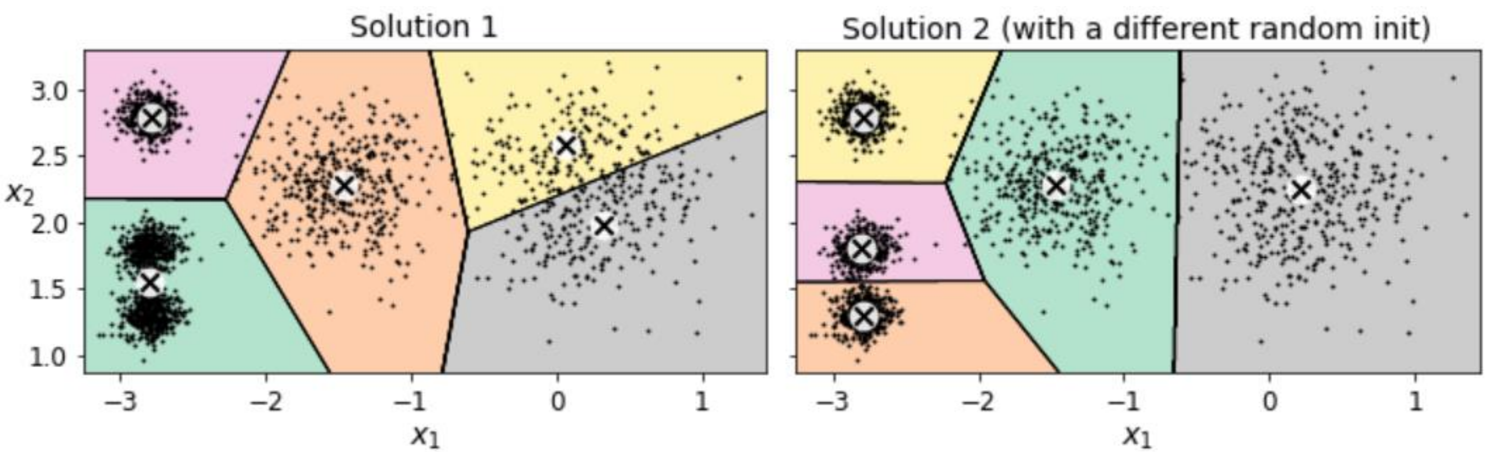
K-means clustering

- Procedure – visualization



K-means clustering

- Uniqueness
  - different results depending on the starting centroids



K-means clustering

- How to measure optimality?
  - unsupervised has no labels
  - inertia
    - distance between centroid and samples

```python
kmeans.inertia_
```

211.5985372581684

```python
X_dist = kmeans.transform(X)
np.sum(X_dist[np.arange(len(X_dist)), kmeans.labels_]**2)
```

211.59853725816856

Gyeongsang National University 경상국립대학교

K-means clustering

- Make multiple clustering model

```
kmeans_rnd_init1.inertia_
```

219.43539442771402

```
kmeans_rnd_init2.inertia_
```

211.5985372581684

```
kmeans_rnd_10_inits = KMeans(n_clusters=5, init="random", n_init=10,
                             algorithm="full", random_state=2)
kmeans_rnd_10_inits.fit(X)
```

KMeans(algorithm='full', init='random', n_clusters=5, random_state=2)

GNU Gyeongsang National University 경상국립대학교

K-means clustering

- Centroid initialization
  - k-means++ clustering
    - find centroid which is far from other centroids

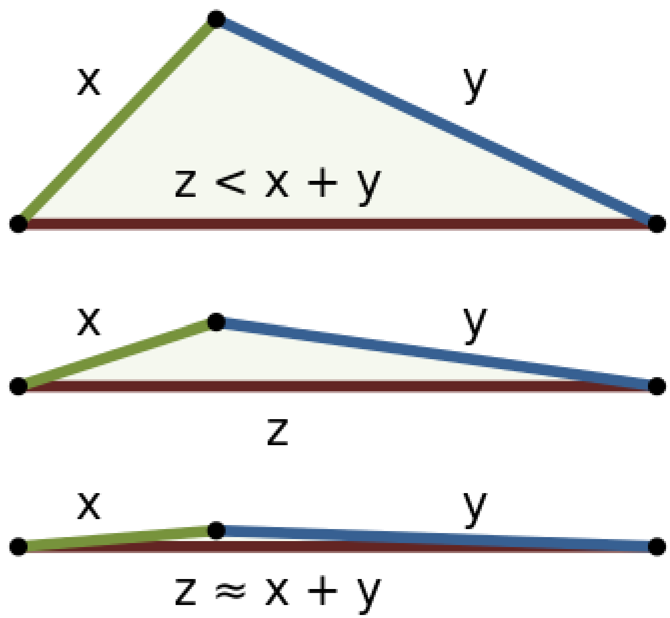
```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1, random_state=42)
kmeans.fit(X)
kmeans.inertia_
```

```
211.59853725816836
```

Gyeongsang National University 경상국립대학교

K-means clustering

- Improvement in time complexity
    - using triangle inequality



Gyeongsang National University 경상국립대학교

K-means clustering

- Improvement in time complexity – implementation

```
%timeit -n 50 KMeans(algorithm="elkan", random_state=42).fit(X)
```

```
1.41 s ± 25.6 ms per loop (mean ± std. dev. of 7 runs, 50 loops each)
```

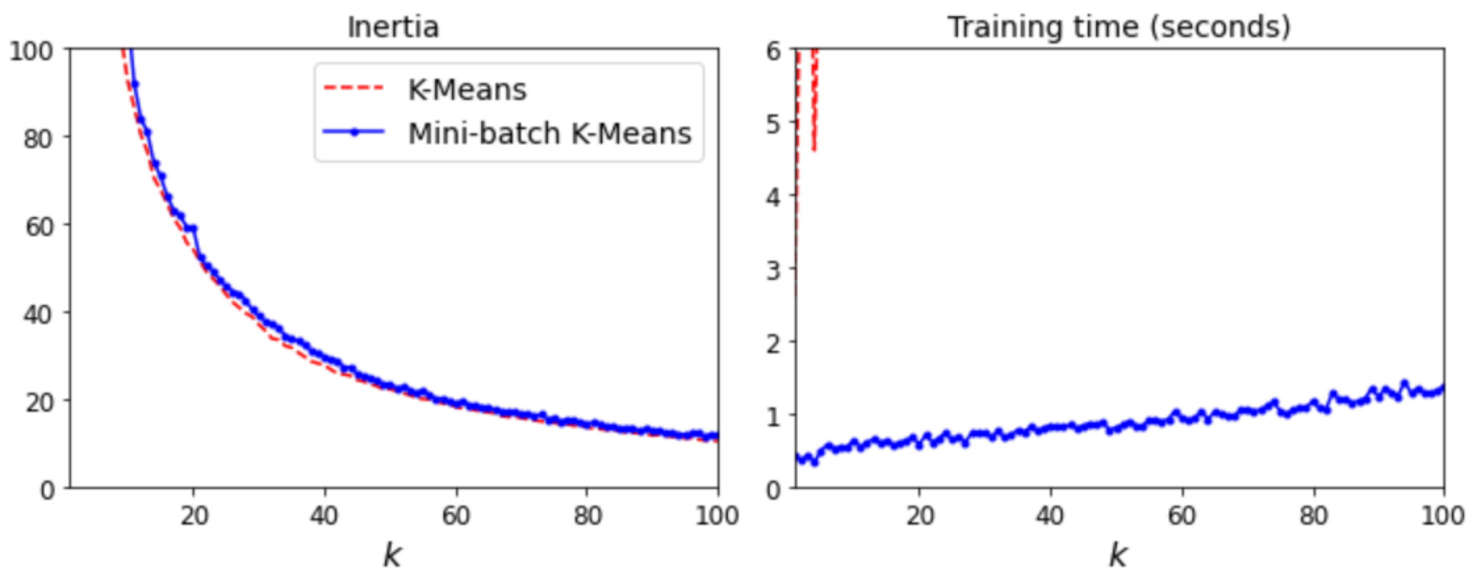
```
%timeit -n 50 KMeans(algorithm="full", random_state=42).fit(X)
```

```
1.46 s ± 23.1 ms per loop (mean ± std. dev. of 7 runs, 50 loops each)
```

GNU
Gyeongsang
National
University
경상국립대학교

K-means clustering

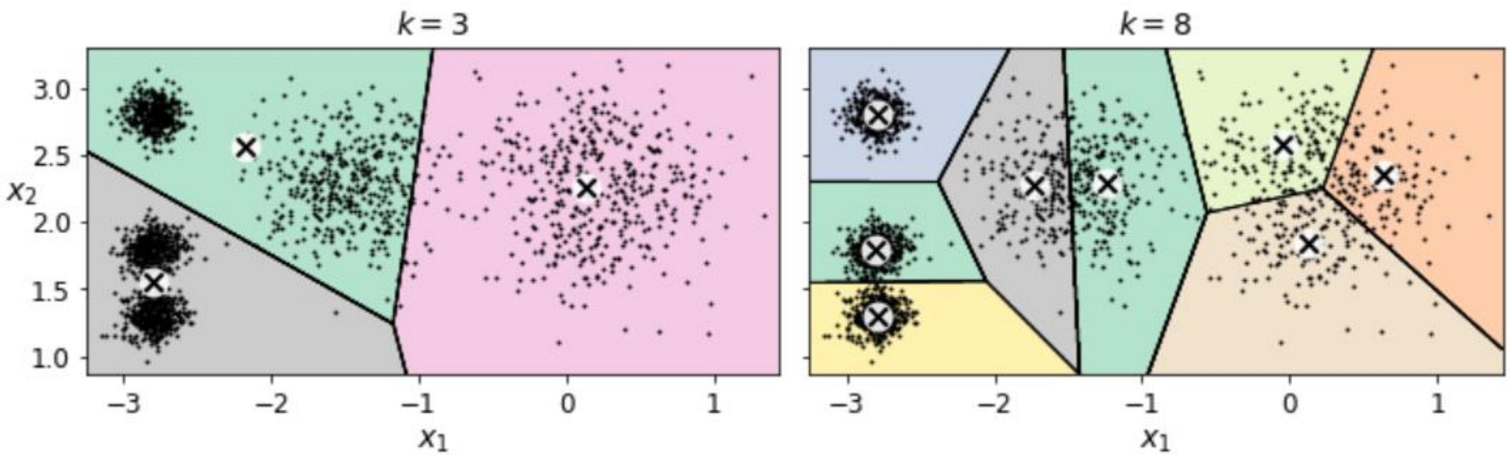
- Minibatch k-means clustering
  - training through mini-batch, not all data



Gyeongsang National University 경상국립대학교

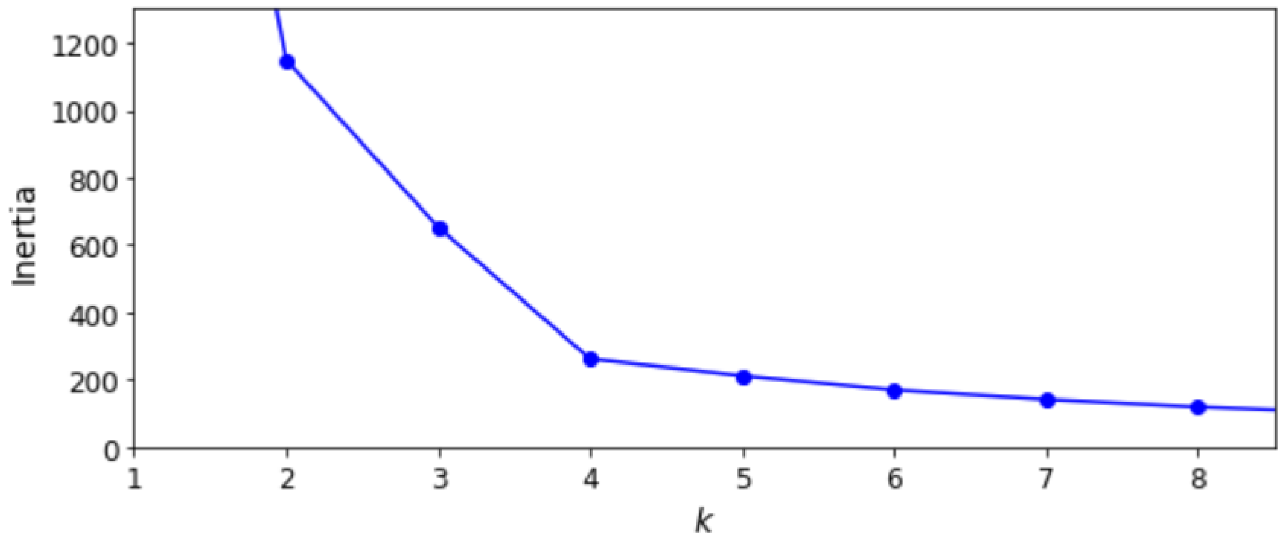
K-means clustering

- Impact of k value
  - finding optimal k is important



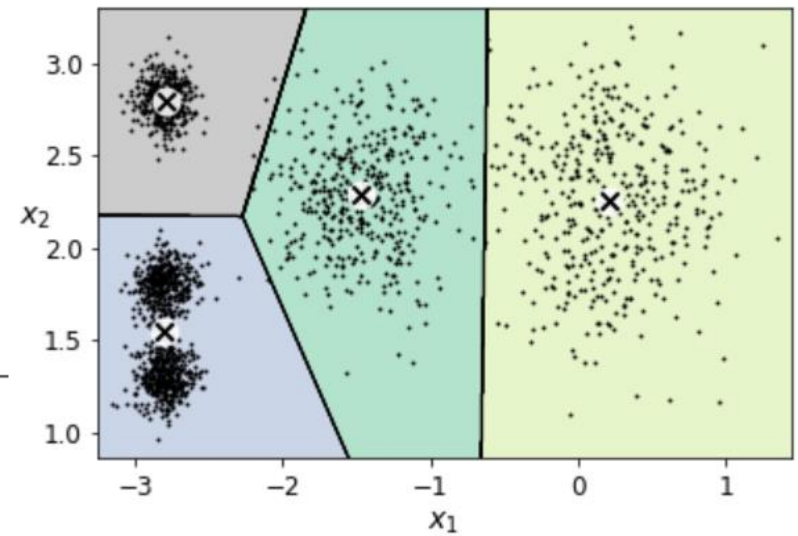
K-means clustering

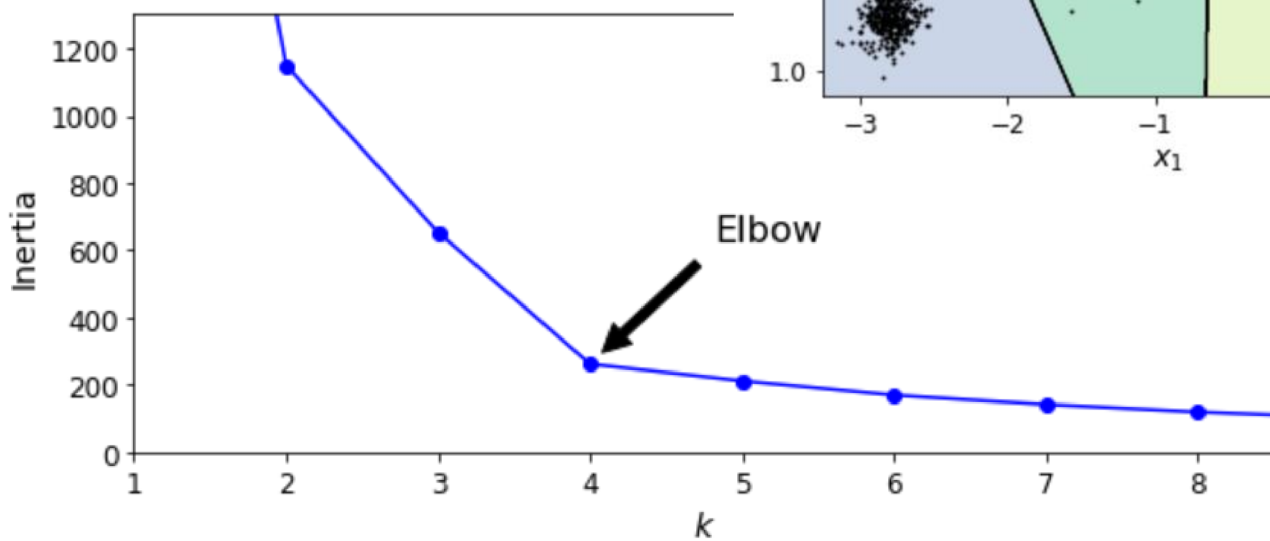
- Inertia according to k
  - when k increases, average distance is getting smaller



K-means clustering

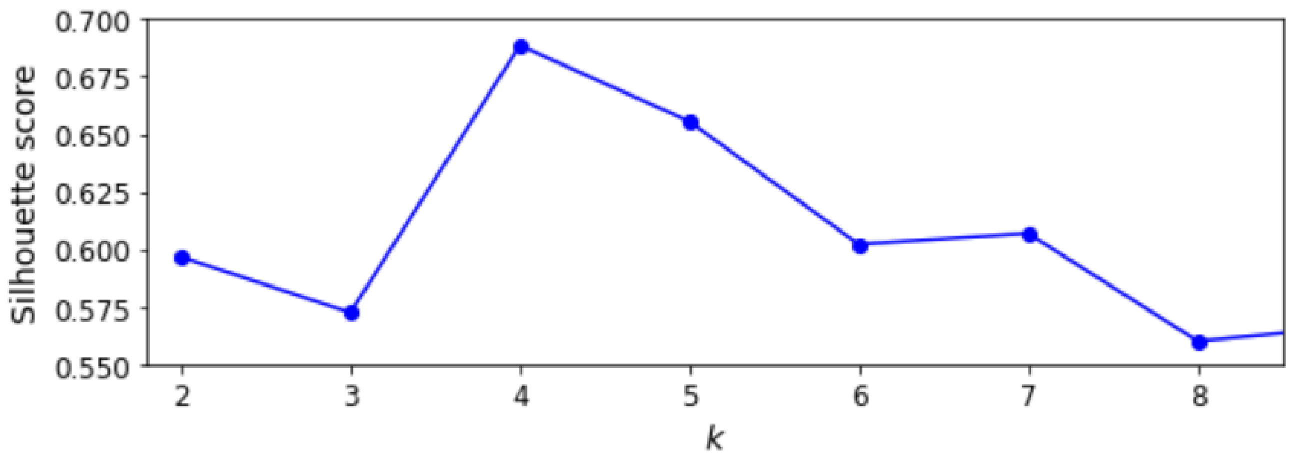
- Inertia according to k
  - finding elbow point





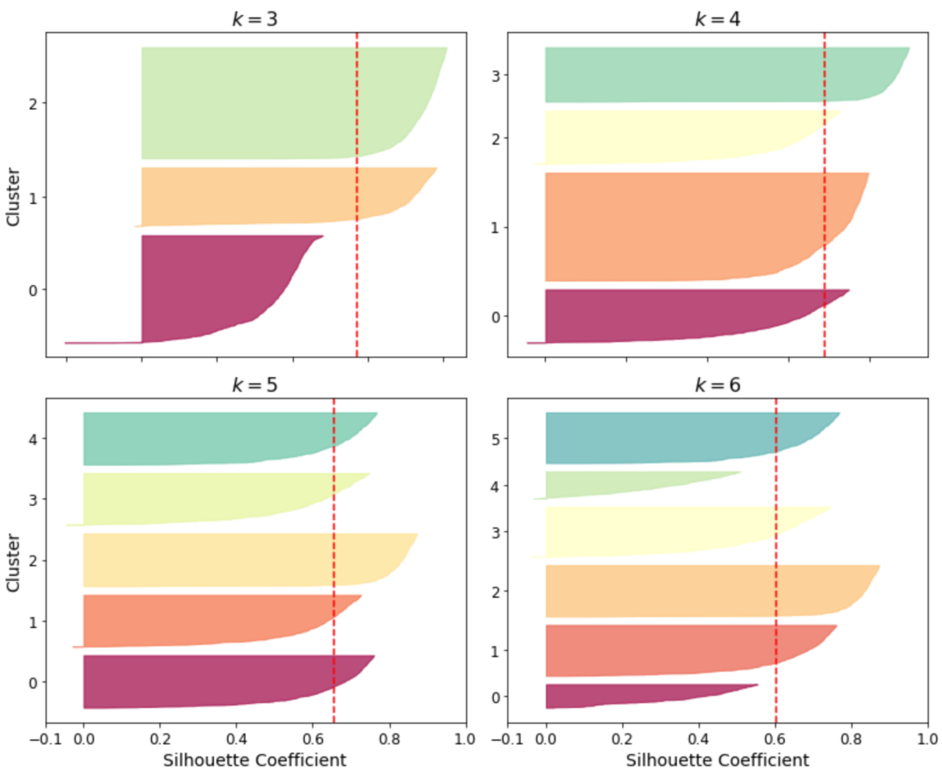
K-means clustering

- Silhouette score
  - average distance inside the cluster



K-means clustering

- Silhouette diagram
  - height
    - the number of samples
  - length
    - silhouette coefficient

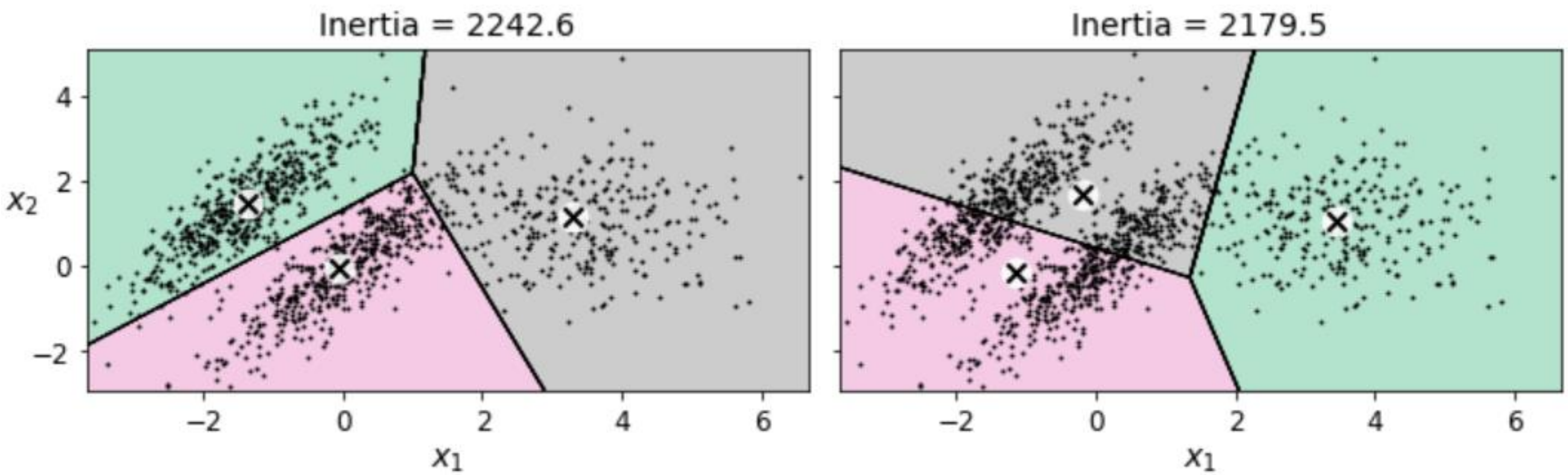


K-means clustering

- Trivia about k-means clustering
  - (+) fast
  - (+) scalable
  - (-) not unique (cannot find global optimum)
  - (-) vulnerable to data properties

K-means clustering

- Limitation of k-means clustering



Applications of clustering

- Segmentation
  - image segmentation
    - division of images into segments
  - semantic segmentation
    - every object belonging to the same type are allocated to the same segment
  - color segmentation
    - allocate pixels which have similar color to the same segment

Applications of clustering

- Example of a color segmentation

Applications of clustering

- Preprocessing based on clustering
  - clustering before supervised learning
    - 8 x 8 mono-color 1,797 images
      - just apply logistic regression
      - pipeline using k-means clustering then logistic regression

Applications of clustering

- Preprocessing based on clustering
  - logistic regression

```
from sklearn.linear_model import LogisticRegression
```

```
log_reg = LogisticRegression(multi_class="ovr", solver="lbfgs", max_iter=5000, random_state=42)
log_reg.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=5000, multi_class='ovr', random_state=42)
```

```
log_reg_score = log_reg.score(X_test, y_test)
log_reg_score
```

```
0.9688888888888889
```

Gyeongsang National University 경상국립대학교

Applications of clustering

- Preprocessing based on clustering
  - k-means clustering → logistic regression

```python
from sklearn.pipeline import Pipeline
```

```python
pipeline = Pipeline([
    ("kmeans", KMeans(n_clusters=50, random_state=42)),
    ("log_reg", LogisticRegression(multi_class="ovr", solver="lbfgs", max_iter=5000, random_state=42)),
])
pipeline.fit(X_train, y_train)
```

```
Pipeline(steps=[('kmeans', KMeans(n_clusters=50, random_state=42)),
                ('log_reg',
                 LogisticRegression(max_iter=5000, multi_class='ovr',
                                    random_state=42))])
```

```python
pipeline_score = pipeline.score(X_test, y_test)
pipeline_score
```

```
0.9777777777777777
```

Applications of clustering

- Preprocessing based on clustering
  - apply grid search to pipeline process

```
param_grid = dict(kmeans__n_clusters=range(2, 100))
grid_clf = GridSearchCV(pipeline, param_grid, cv=3, verbose=2)
grid_clf.fit(X_train, y_train)
```

```
grid_clf.best_params_
```

```
{'kmeans__n_clusters': 88}
```

```
grid_clf.score(X_test, y_test)
```

```
0.9822222222222222
```

Gyeongsang National University 경상국립대학교

Applications of clustering

- Semi-supervised learning
    - small size of labelled samples

```
n_labeled = 50
```

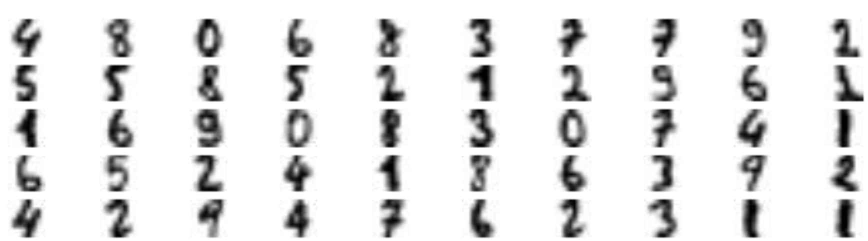
```
log_reg = LogisticRegression(multi_class="ovr", solver="lbfgs", random_state=42)
log_reg.fit(X_train[:n_labeled], y_train[:n_labeled])
log_reg.score(X_test, y_test)
```

```
0.8333333333333334
```

Applications of clustering

- Semi-supervised learning
    - clustering can make representative image
    - make labels by programmer himself/herself

```
kmeans = KMeans(n_clusters=k, random_state=42)
X_digits_dist = kmeans.fit_transform(X_train)
representative_digit_idx = np.argmin(X_digits_dist, axis=0)
X_representative_digits = X_train[representative_digit_idx]
```



Gyeongsang National University 경상국립대학교

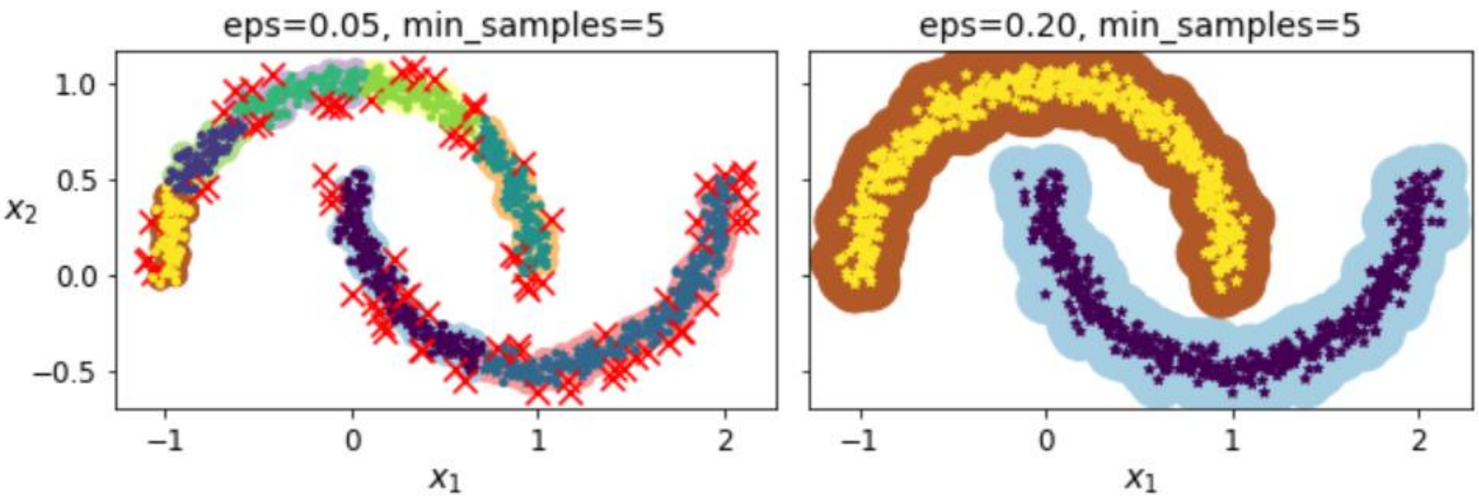
Applications of clustering

- DBSCAN
  - Find the points in the ε neighborhood of every point and identify the core points with more than minPts neighbors
  - Find the connected components of core points on the neighbor graph, ignoring all non-core points
  - Assign each non-core point to a nearby cluster if the cluster is an ε neighbor, otherwise assign it to noise

GNU Gyeongsang National University 경상국립대학교

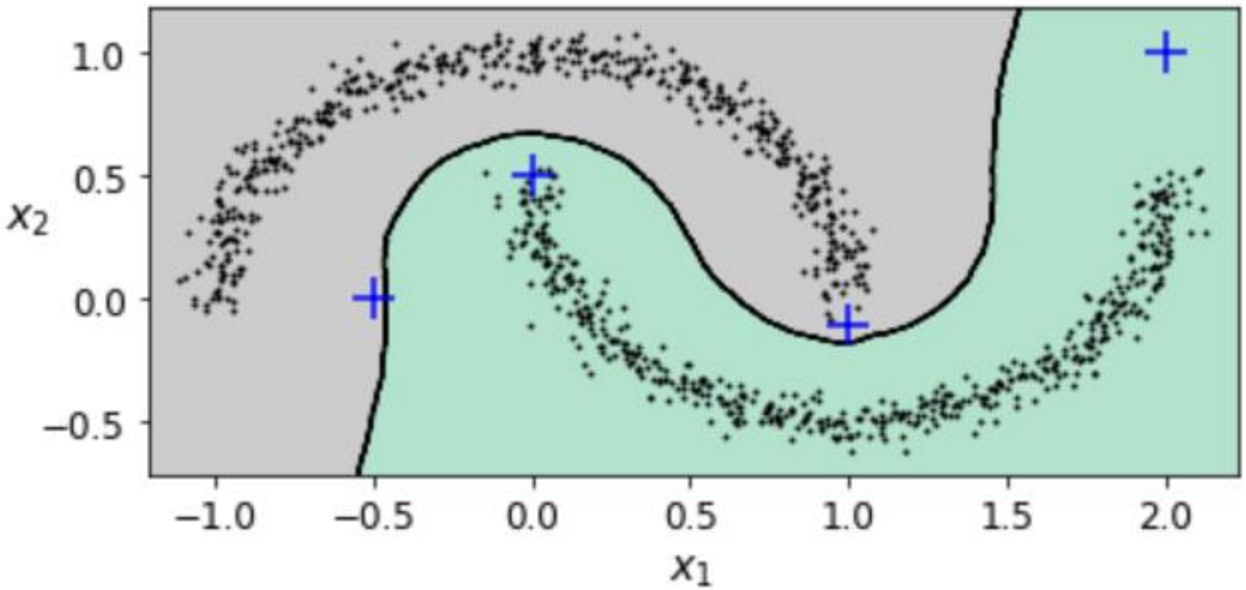
Applications of clustering

- DBSCAN
  - implementation



Applications of clustering

- DBSCAN
  - classification using clustering



Applications of clustering

- Other clustering algorithms
  - agglomerative clustering
  - BIRCH
  - mean-shift
  - affinity propagation
  - spectral clustering

Gyeongsang National University 경상국립대학교

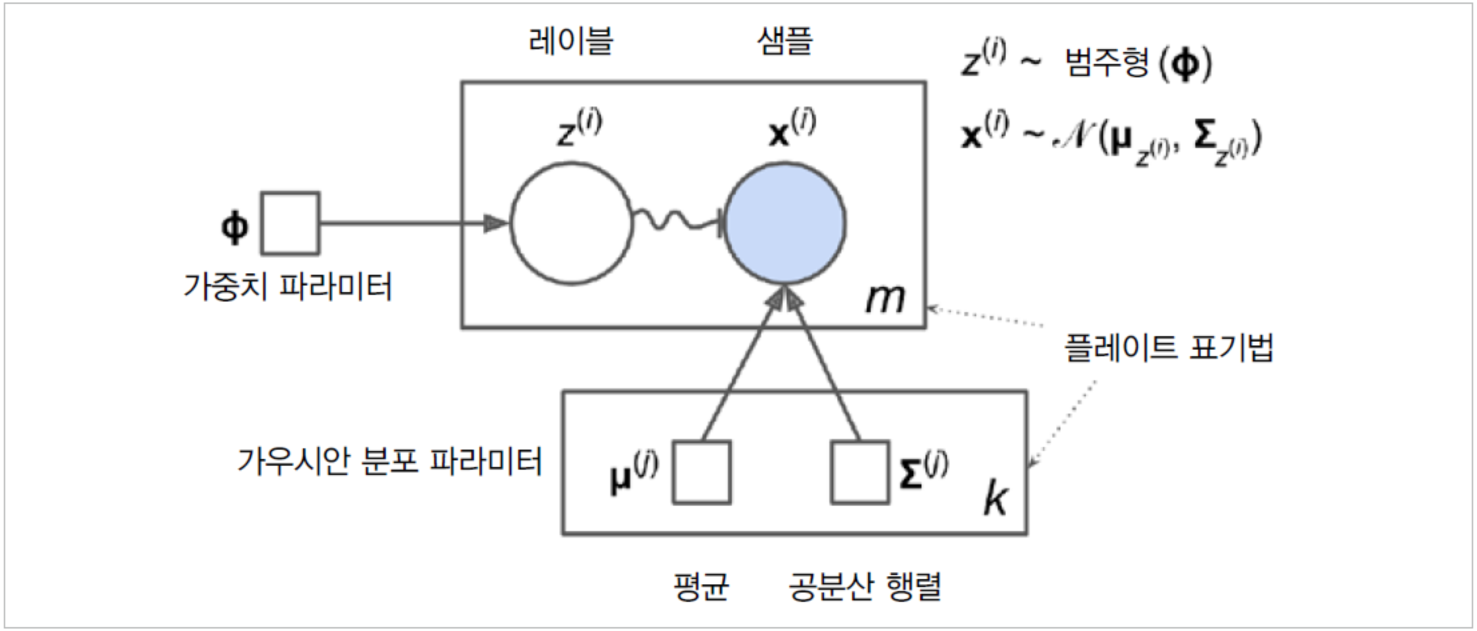
Gaussian mixture model

- Basic idea
  - assume that samples are generated from mixture of Gaussian distribution

Gyeongsang National University 경상국립대학교

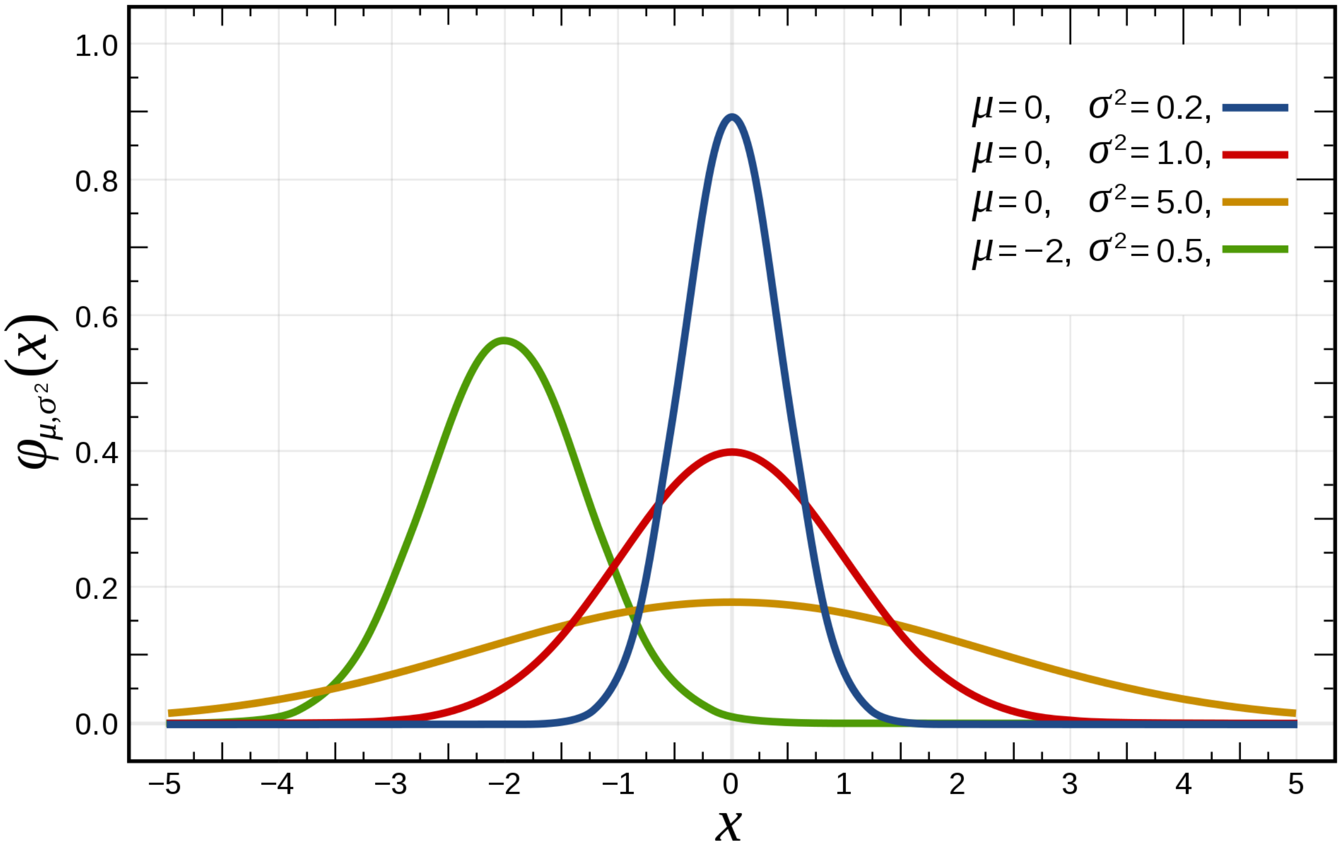
Gaussian mixture model

- Graph diagram for Gaussian mixture



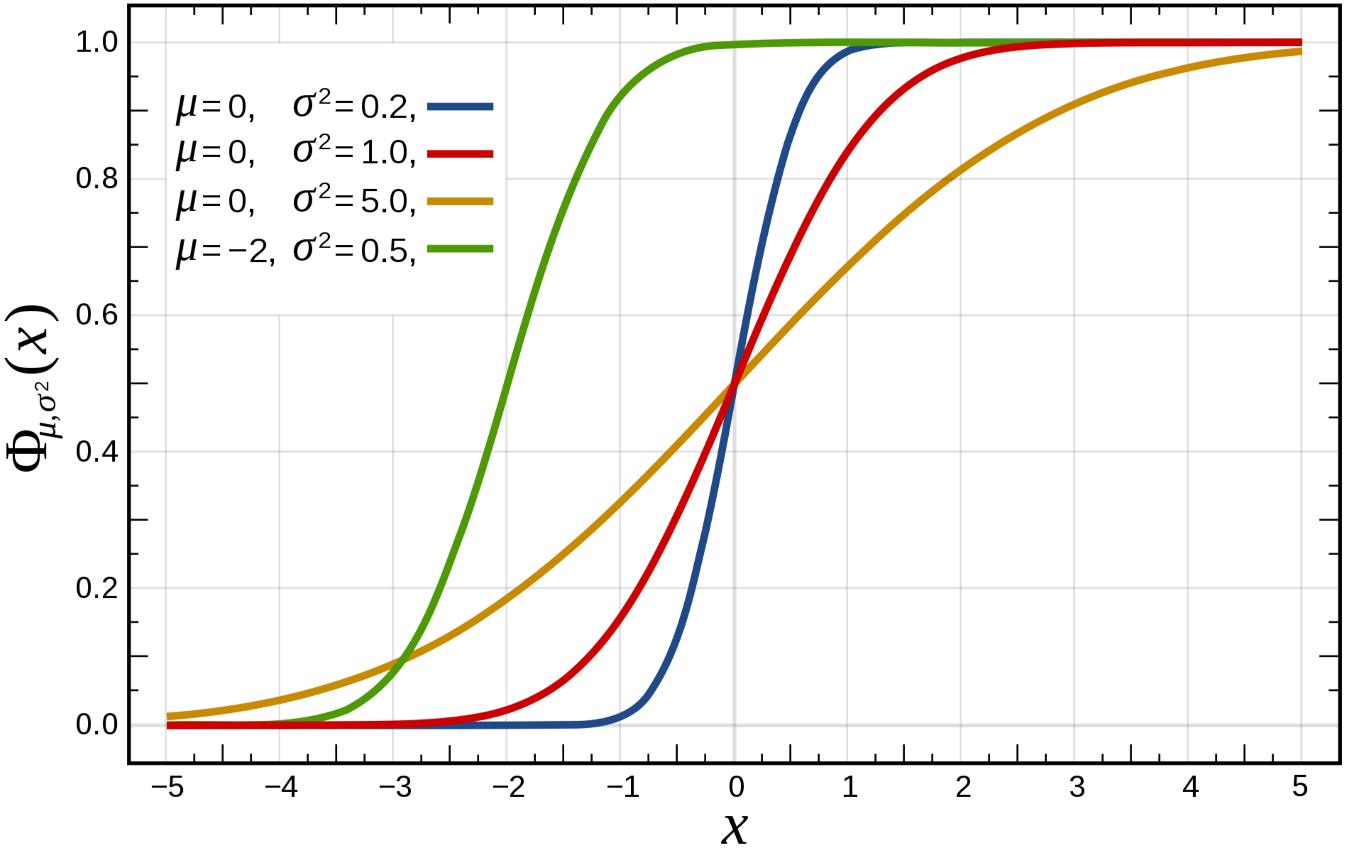
Gaussian mixture model

- Gaussian distribution



Gaussian mixture model

- Gaussian distribution



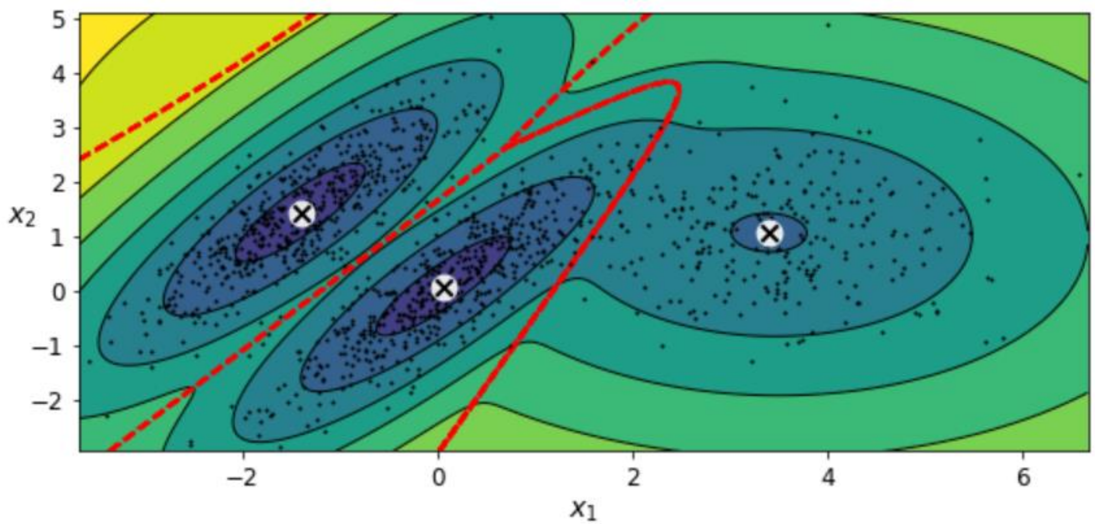
Gaussian mixture model

- Central limit theorem
  - the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable—whose distribution converges to a normal distribution as the number of samples increases

Gyeongsang National University 경상국립대학교

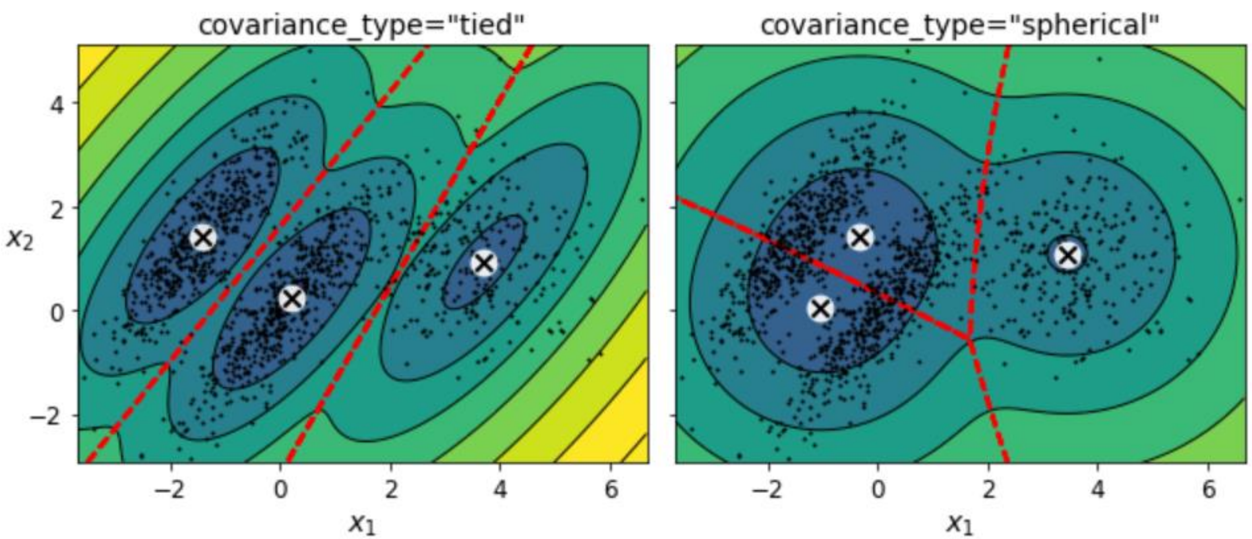
Gaussian mixture model

- Process
  - Expectation-maximization
    - (expectation) allocate samples to a cluster
    - (maximization) update parameters of the cluster



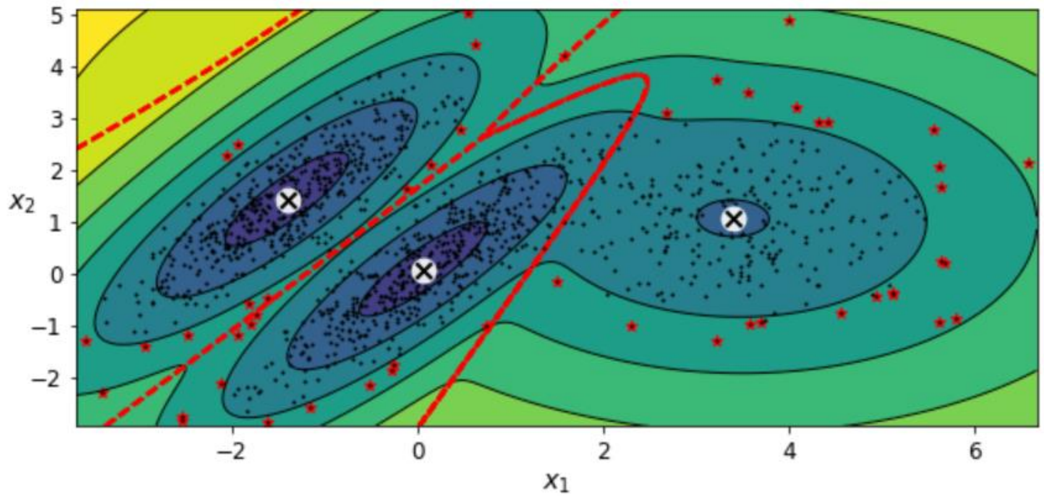
Gaussian mixture model

- Limitation to cluster parameters
  - the number of clusters
  - the type of covariance



Gaussian mixture model

- Applications
  - anomaly detection
    - consider samples in low density areas as outliers



Gaussian mixture model

- Selecting the number of clusters
  - Bayesian information criterion (BIC)
  - Akaike information criterion (AIC)

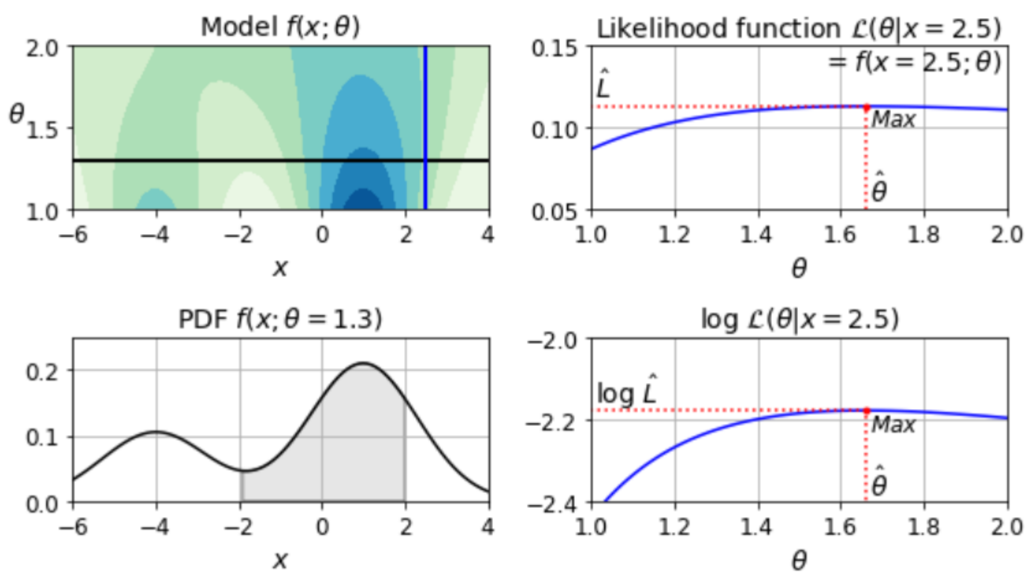
$$BIC = \log(m)p - 2\log(\hat{L})$$

$$AIC = 2p - 2\log(\hat{L})$$

- m: the number of samples
- p: the number of parameters
- $\hat{L}$: maximum value of likelihood function

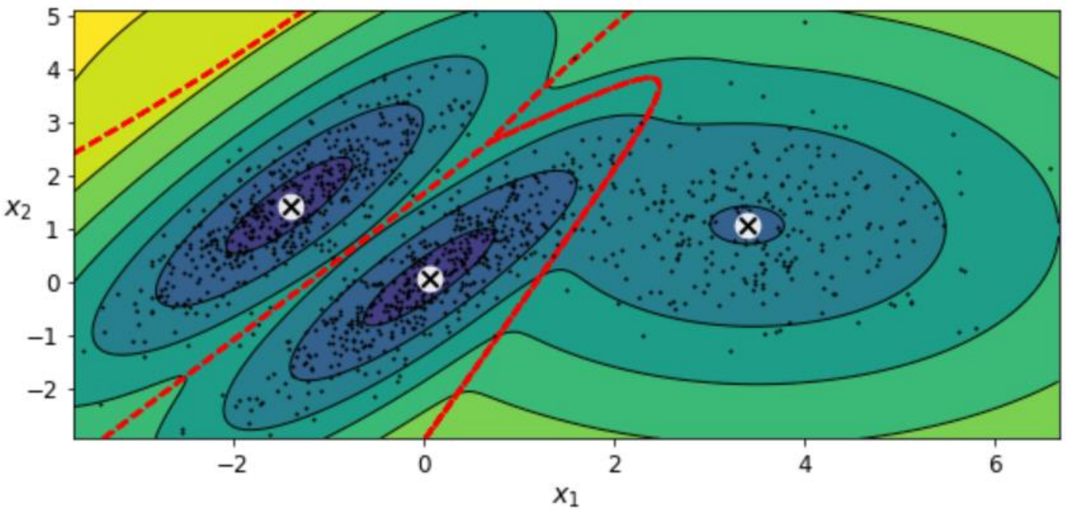
Gaussian mixture model

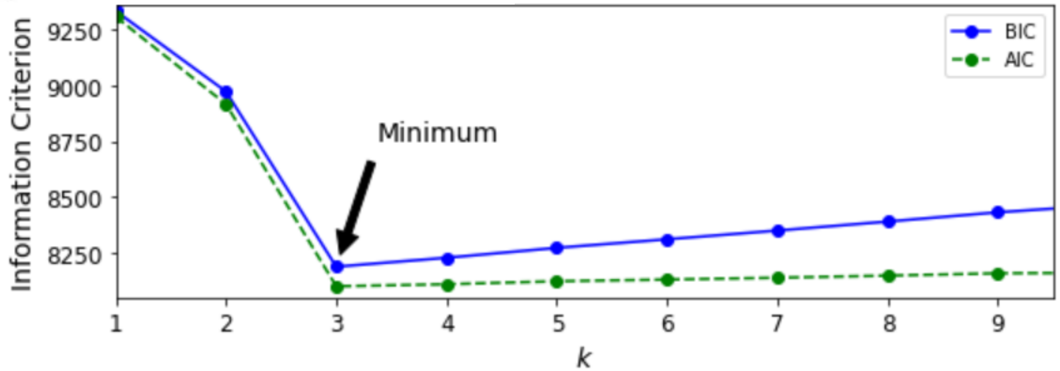
- Likelihood function
  - output x, parameter θ



Gaussian mixture model

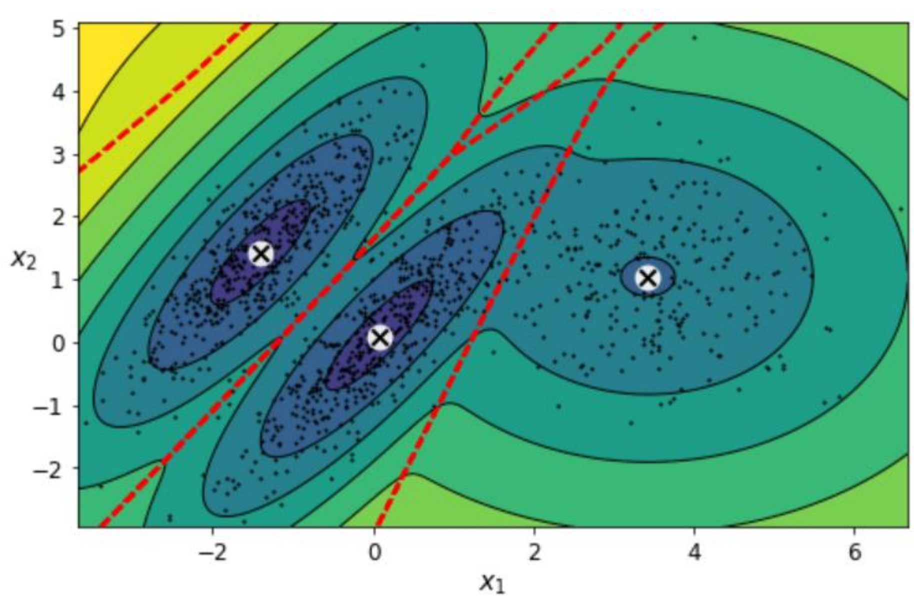
- Plotting BIC and AIC





Gaussian mixture model

- Bayesian Gaussian mixture
  - without manually finding the optimal number of clusters, count unnecessary cluster weight as zero



Gaussian mixture model

- Other algorithms for anomaly detection
  - PCA
    - using reconstruction error
  - fast minimum covariance determinant (Fast-MCD)
    - assume that samples are derived from single Gaussian distribution

Gaussian mixture model

- Other algorithms for anomaly detection
  - isolation forest
    - ensemble of random decision tree, outlier will be isolated
  - local outlier factor
    - comparison with the density of samples
  - one-class SVM
    - consider one-class kernel SVM classifier

Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함