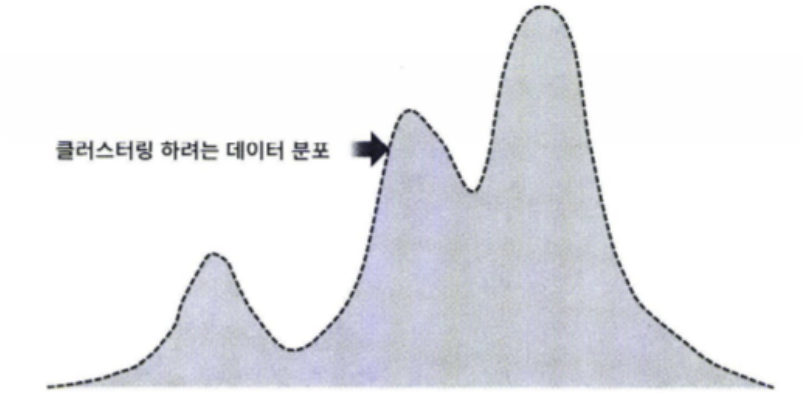
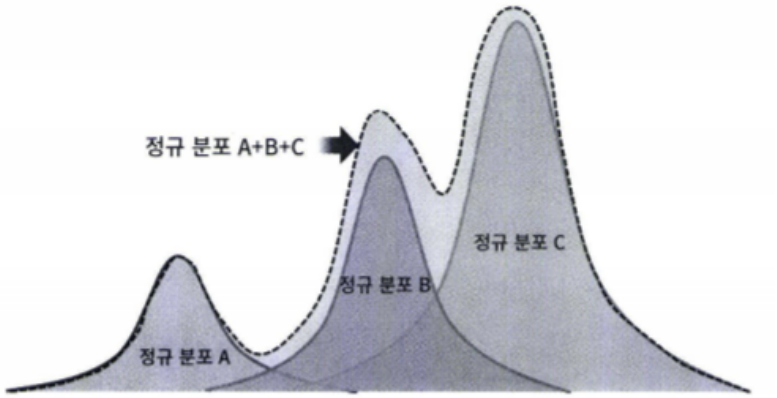
비지도학습

GMM

- 데이터를 여러 개의 가우시안 분포가 섞인 것으로 간주
- 섞인 데이터 분포에서 개별 가우시안 분포를 추출함으로써 군집화를 수행





GMM의 모수추정

1. 전체 분포에서 개별 정규분포를 찾음(개별 정규 분포의 평균과 분산을 찾음)
2. 각 데이터가 어떤 정규 분포에 해당되는지의 확률

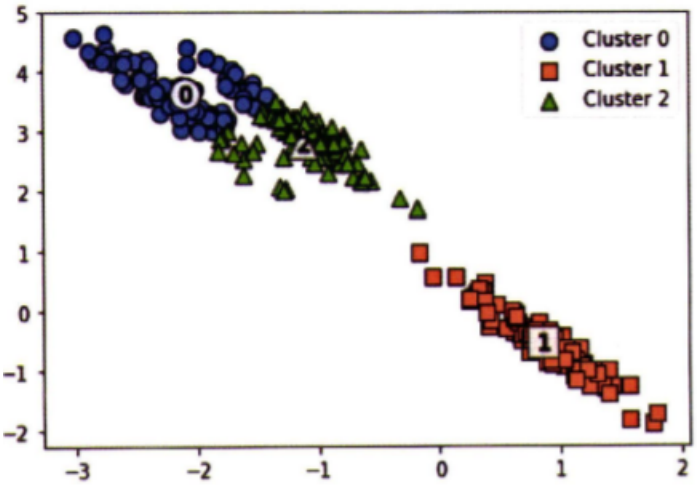
Expectation - Maximization 알고리즘 사용

- Expectation 단계와 Maximization 단계를 번갈아 수행
  - Expectation : 각각의 원소가 어느 집단에 속할지 결정
  - Maximization : 원소들이 각 집단에 속함에 따른 새로운 분포를 생성

1. 데이터가 존재
2. 임의의 평균과 분산이 배정된 분포를 k개 만듦
3. 사전, 사후 확률 공식을 통해 각 원소가 각 분포에 각각 속할 확률을 구함
4. 각각의 원소는 속할 확률이 더 높은 분포에 소속되고, 각 군집을 업데이트
5. 각 군집의 중심점의 평균 및 분산을 재산정, 새로운 분포를 생성
6. 다시 3~5반복

K-Means VS GMM

- 공통점
  1. 두 군집방식 모두 집단을 평균으로 구분한다는 점에서 유사
  2. 모든 데이터 포인트는 가장 가까운 평균의 집단에 소속됨
- 차이점
  - K-Means
    - 데이터 세트가 타원이 아닌 원형일 수록 효율이 높아짐
      - 평균 거리 기반으로 군집화를 수행하기 때문에 다른 군집이 같은 거리상 원형으로 형성되면서 하나의 군집은 길쭉한 방향으로 데이터가 밀집해있을 경우 최적의 군집화가 이루어지지 않음



  - GMM
    - 보다 유연하게 다양한 데이터 세트에 잘 적용될 수 있음
    - 수행시간이 K-Means보다 오래걸림

실습

In [1]:

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline

iris = load_iris()
feature_names = ["SL","SW","PL","PW"]

# 좀 더 편리한 데이터 handling을 위해 DF으로 변환
irisDF = pd.DataFrame(data = iris.data, columns = feature_names)
irisDF["target"] = iris.target
```

In [4]:

```
from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components = 3, random_state = 0).fit(iris.data) # 3개로 군집화
gmm_cluster_labels = gmm.predict(iris.data)

# 군집화 결과를 irisDF의 "gmm_cluster" 칼럼 명으로 저장
irisDF["gmm_cluster"] = gmm_cluster_labels

# target 값에 따라 gmm_cluster 값이 어떻게 매칭됐는지 확인
iris_result = irisDF.groupby(["target"])["gmm_cluster"].value_counts()
print(iris_result)
```

```
target  gmm_cluster
0       0              50
1       1              45
        2               5
2       2              50
Name: gmm_cluster, dtype: int64
```

In [6]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 3, init="k-means++", max_iter = 300, random_state = 0).fit(iris.data)
kmeans_cluster_labels = kmeans.predict(iris.data)
irisDF["kmeans_cluster"] = kmeans_cluster_labels
print(irisDF.groupby(["target"])["kmeans_cluster"].value_counts())
```

```
target  kmeans_cluster
0       1                50
1       2                48
        0                 2
2       0                36
        2                14
Name: kmeans_cluster, dtype: int64
```