

[11주차] 정답 없는 데이터와 마주하기: 비지도 학습

안녕하세요! 11주차 수업에 오신 것을 환영합니다.

지금까지 우리는 '정답(y)'이 주어진 상태에서 모델을 학습시키는

지도 학습(Supervised Learning) (e.g., 선형/로지스틱 회귀, 결정 트리)을 다루었습니다.

오늘은 머신러닝의 또 다른 큰 축인 비지도 학습(Unsupervised Learning)을 배웁니다.

비지도 학습은 정답(y)이 없는 데이터를 주어졌을 때,

데이터 내에 내재된 숨겨진 구조나 패턴을 찾아내는 방식입니다.

오늘은 비지도 학습의 두 가지 대표적인 기법,

1. K-Means 군집화(Clustering): 데이터의 숨겨진 그룹을 찾는 기법

2. 주성분 분석(PCA): 데이터의 차원을 축소하여 시각화하는 기법

에 대해 학습하고 scikit-learn으로 직접 구현해 보겠습니다.

모듈 1: K-Means 군집화(Clustering)

군집화(Clustering)란?

• 정답 데이터가 없는 데이터들을 서로 비슷한 그룹으로 묶는 작업입니다.

• 비유: 여러분에게 빨강, 파랑, 노랑 양말이 마구 섞인 양말 서랍이 주어졌을 때,
각 양말에 '빨강'이라는 정답 라벨이 없어도, 여러분은 자연스럽게
색깔과 모양이 비슷한 것끼리 세 개의 무더기로 나눌 수 있습니다. 이것이 군집화입니다.

K-Means 알고리즘의 직관적 이해:

• K-Means는 K개의 그룹(Cluster)을 찾는 가장 유명한 알고리즘이다.

• 작동 원리(비유): '조별 과제 조장 청하기':

1. 초기화

K명의 조장(Centroid)을 무작위로 고상 아무 데나 배치합니다.

2. 할당(Assign)

모든 학생(데이터)이 자신에게 가장 가까운 조장과 조를 구성합니다.

3. 업데이트(Update)

각 조장들은 자신에게 모인 조원들의 정중상(평균 위치)으로 이동합니다.

■ [반복]

이제 조장의 위치가 바뀌었으므로, 다시 (2)의 방식으로 조를 재구성합니다.

■ [수렴]

조장들의 위치가 바뀌지 않을 때 까지(인정화) 2~3단계를 반복합니다.

K-Means 실습 1: 가상 데이터로 원리 확인

scikit-learn을 이용해 K-Means 알고리즘을 직접 적용해 보겠습니다.

먼저, make_blobs라는 함수를 사용해 군집화 연습용 가상 데이터를 생성합니다.

In [27]:

```
# 필요한 라이브러리 import
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs # 군집화 연습용 데이터 생성기
from sklearn.cluster import KMeans # K-Means 모델
from sklearn.decomposition import PCA # 차원 축소
from sklearn.preprocessing import StandardScaler # 데이터 스케일링
from sklearn.datasets import load_iris # Iris 데이터셋

# 1. 데이터 준비 (정답 없는 데이터 생성)
# n_samples=300: 300개의 샘플 생성
# centers=3: 3개의 그룹(중심)을 가지도록 생성
# cluster_std=2.0: 그룹의 퍼진 정도
# random_state=42: 재현성을 위한 시드 고정
# X는 (300, 2) shape의 2D 데이터, y_true는 실제 정답(0,1,2)
X, y_true = make_blobs(n_samples=300, centers=3,
                       cluster_std=2.0, random_state=42)

# 2. 데이터 탐색 (시각화)
# 비지도 학습으로, 정답(y_true)을 모르고 가정하고
# 모든 데이터를 회색(c='gray')으로 시작합니다.
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c='gray', s=50, alpha=0.7)
plt.title("Unlabeled Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid(True)
plt.show()

print("우리의 목표: 이 회색 점들을 3개의 그룹으로 분류하는 것입니다.")
```

우리의 목표: 이 회색 점들을 3개의 그룹으로 분류하는 것입니다.

K-Means 실습 2: 모델 학습 및 결과 확인

이제 KMeans 모델을 생성하고 학습시킵니다.

핵심 파라미터는 n_clusters (K값)입니다.

In [28]:

```
# 1. K-Means 모델 생성
# n_clusters=3: 우리가 찾으려는 그룹의 개수(K)를 3으로 지정
# random_state=42: 조장의 초기 무작위 위치를 고정
model_kmeans = KMeans(n_clusters=3, random_state=42)

# 2. 모델 학습 (fit)
# 비지도 학습으로, 정답(y) 없이 X 데이터만 입력합니다!
# 이 과정에서 내부적으로 '할당'과 '업데이트' 단계가 반복됩니다.
model_kmeans.fit(X)

# 3. 학습 결과 확인
# model.labels_: 모델이 각 데이터에 할당한 그룹 번호(0, 1, 2)
y_pred = model_kmeans.labels_
# model.cluster_centers_: 학습이 완료된 3개 조장(Centroid)의 최종 위치
centroids = model_kmeans.cluster_centers_

print("K-Means 학습 완료!")
print(f"예측된 그룹 라벨(앞 10개): {y_pred[:10]}")
print(f" 중심점 위치: \n{centroids}")
```

K-Means 학습 완료!

예측된 그룹 라벨(앞 10개): [1 0 2 1 2 0 2 0 0]

최종 중심점 위치:

[-2.9 0.76773 9.07285344]

[-6.88811639 -7.0878587]

[4.85432791 2.04801886]

학습된 모델이 데이터를 어떻게 그룹화했는지,

model.labels_를 이용해 시각화해 봅시다.

In [29]:

```
# K-Means 군집화 결과 시각화
plt.figure()
# 데이터를 산점도로 그리되, 색깔(c)을 모델이 예측한 그룹(y_pred)으로 지정
plt.scatter(X[:, 0], X[:, 1], c=y_pred, cmap='viridis', alpha=0.7)

# 최종 중심점(Centroid)을 빨간색 'x' 마커로 겹쳐 그리기
plt.scatter(centroids[:, 0], centroids[:, 1],
            c='red', s=200, marker='x', label='Centroids')

plt.title("K-Means Clustering Result (K=3)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.grid(True)
plt.show()

print("정답(레이블) 없이도 데이터의 분포만으로 그룹을 성공적으로 찾아냈습니다.")
```

정답(레이블) 없이도 데이터의 분포만으로 그룹을 성공적으로 찾아냈습니다.

모듈 2: 최적의 K값 찾기(엘보우 메소드)

방금 우리는 정답이 3개인 것을 알고 n_clusters=3으로 설정했습니다.

하지만 실제 비지도 학습 상황에서는 K값이 몇 개가 적절한지 모릅니다.

K값을 어떻게 선택할까요?

• 관성(Inertia): 각 데이터 포인트가 자신이 속한 그룹의 중심(Centroid)으로부터 얼마나 멀어져 있는지를 나타내는 값입니다. (제곱 거리의 총합)

▪ Inertia 값이 작을수록 데이터들이 중심에 잘 뭉쳐있다는 의미입니다.

• Inertia의 특성:

▪ K=1 (그룹 1개): 모든 데이터가 하나의 중심을 공유하므로 Inertia가 최대.

▪ K=n (데이터 개수만큼): 모든 데이터가 자기 자신이 중심으로 Inertia는 0.

(하지만 의미 없음)

엘보우 메소드(Elbow Method):

```
• K=1부터 10까지 늘려가면서, 각 K값에 대한 Inertia를 계산하여 그래프로 그립니다.
• 그래프가 급격히 줄어드는 지점(팔꿈치, Elbow)이 나타나는데,
  이 지점이 K를 늘릴 때 얻는 효율(Inertia 감소)이 급격히 줄어드는 지점이므로,
  최적의 K값 후보로 간주할 수 있습니다.
print("K값에 따른 Inertia 계산 완료!")
```

K값에 따른 Inertia 계산 완료!

```
Elbow Method
Inertia
K
[20000, 17500, 15000, 12500, 10000, 7500, 5000, 2500, 0]
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[20000, 17500, 15000, 12500, 10000, 7500, 5000, 2500, 0]
```

결과 해석:

- K=1에서 2로, 2에서 3으로 갈 때는 Inertia가 급격히 감소합니다.

- K=3에서 4로 갈 때부터는 감소폭이 원만해집니다.

=> 따라서, 이 그래프의 '팔꿈치(Elbow)' 지점은 K=3이며,

최적의 클러스터 개수는 3개라고 판단할 수 있습니다.

In [30]:

```
# K-Means 군집화 결과 시각화
plt.figure()
# 데이터를 산점도로 그리되, 색깔(c)을 모델이 예측한 그룹(y_pred)으로 지정
plt.scatter(X[:, 0], X[:, 1], c=y_pred, cmap='viridis', alpha=0.7)

# 최종 중심점(Centroid)을 빨간색 'x' 마커로 겹쳐 그리기
plt.scatter(centroids[:, 0], centroids[:, 1],
            c='red', s=200, marker='x', label='Centroids')

plt.title("K-Means Clustering Result (K=3)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.grid(True)
plt.show()

print("정답(레이블) 없이도 데이터의 분포만으로 그룹을 성공적으로 찾아냈습니다.")
```

정답(레이블) 없이도 데이터의 분포만으로 그룹을 성공적으로 찾아냈습니다.

모듈 3: 차원 축소와 PCA

이제 비지도 학습의 두 번째 주제인 차원 축소(Dimensionality Reduction)로 넘어갑니다.

차원이란?

• 데이터가 가진 특징(Feature)의 개수입니다.

• 붓꽃(Iris) 데이터: 4개의 특징 (sepal length/width, petal length/width)

-> 4차원 데이터

• 차원을 축소할까요?

1. 시각화: 인자는 3차원까지만 인지할 수 있습니다.

• 4차원, 100차원 데이터는 시각화가 불가능합니다.

• 데이터를 차원이나 3차원으로 압축하면, 데이터의 본질을 눈으로 확인할 수 있습니다.

2. 차원의 저차원 (Curse of Dimensionality): 차원이 너무 높아지면 특징이 너무 많아집니다.

데이터가 존재하는 공간은 매우 넓어지는 데 비해 데이터는 희박해져(sparse)

오히려 모델의 성능이 저하될 수 있습니다. (e.g., 불필요한 노이즈 특징)

3. 차원의 저차원 (Dimensionality Reduction): 차원을 축소하는 과정입니다.

• 차원 축소의 목표: 차원을 축소하면서도 데이터의 정보를最大限度으로 유지하는 것입니다.

• 차원 축소의 종류: 차원 축소 방법

• 차원 축소