5주차

Chapter 6. 비지도 학습

▼ 6-1 군집 알고리즘

군집 알고리즘은 타깃이 없는 데이터에서 비슷한 데이터끼리 묶어주는 비지도 학습의 대표적인 기법

사과, 바나나, 파인애플 흑백 사진 군집화

1. 데이터 준비와 구조 확인

- 데이터 로드: 사과, 바나나, 파인애플 흑백 이미지 데이터는 넘파이 `npy` 파일로 제공
- 데이터 구조: `fruits` 배열은 `(300, 100, 100)` 크기, 이는 300개의 샘플, 각각 100x100 크기의 이미지가 포함된 구조

2. 이미지 시각화

- 이미지 출력: 첫 번째 이미지를 `plt.imshow()`로 출력하면 사과 이미지가 나타남
- 색상 반전: `cmap='gray_r'` 설정을 통해 밝은 부분이 0에 가깝고, 어두운 부분이 255에 가깝게 표시됨

3. 데이터 분할 및 분석

- 데이터 분할: 사과, 파인애플, 바나나 데이터는 각각 `100개씩` 슬라이싱하여 별도의 배열로 저장
- 픽셀 평균값 분석: 각 과일의 이미지 평균값을 계산해 히스토그램으로 시각화함
- 바나나의 평균 픽셀값이 40 이하로 낮은 반면, 사과와 파인애플은 90~100 사이에 분포

4. 픽셀별 평균값 비교

- 픽셀 위치별 비교: 각 과일의 픽셀 평균값을 막대그래프로 표현해 위치별 차이를 확인
- 대표 이미지 시각화: 각 과일의 평균값 이미지를 `100x100` 배열로 재구성하여 출력. 각 과일의 대표 이미지와 가까운 사진을 고름

5. 평균값과 가까운 사진 찾기

- 절댓값 오차 계산: 각 샘플과 대표 이미지의 절댓값 오차 평균을 계산하여 apple_mean과 오차가 작은 사과 이미지를 선별
- 가장 비슷한 100개 사진: 오차가 작은 순으로 100개 이미지를 10x10 격자에 출력

▼ 6-2 k-평균

k-평균 군집 알고리즘은 데이터 샘플을 k개의 클러스터로 분류하는 비지도 학습 알고리즘

클러스터의 중심은 "센트로이드"라 불리며, 알고리즘은 반복적으로 센트로이드를 찾아내며 최적의 군집화를 수행

k-평균 알고리즘 단계

1. 무작위로 k개의 클러스터 중심을 초기화
2. 각 샘플에서 가장 가까운 클러스터 중심을 찾아 해당 클러스터에 할당
3. 클러스터에 속한 샘플들의 평균을 사용해 클러스터 중심을 업데이트
4. 클러스터 중심의 위치 변화가 없을 때까지 2-3단계를 반복

KMeans 클래스 사용법

1. 데이터 준비: 데이터를 다운로드하고, 3차원 배열을 2차원 배열로 변환

```
fruits = np.load('fruits_300.npy')
fruits_2d = fruits.reshape(-1, 100*100)
```

2. 모델 훈련: sklearn의 `KMeans` 클래스를 사용하여 n_clusters=3으로 지정한 뒤 fit() 메서드를 호출해 모델을 훈련

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=3, random_state=42)
km.fit(fruits_2d)
```

3. 결과 확인: `labels_` 속성을 통해 각 샘플의 클러스터 레이블을 확인

```
print(km.labels_)
```

군집화 결과 시각화

- 클러스터별로 데이터를 시각화하여 군집이 어떤 과일 사진으로 구성되는지 확인

```
draw_fruits(fruits[km.labels_==0])
```

클러스터 중심 (센트로이드)

`KMeans`는 최종 클러스터 중심을 `cluster_centers_` 속성에 저장, 이미지 형태로 시각화, 2차원 배열을 100x100 크기의 2차원 배열로 변환

```
draw_fruits(km.cluster_centers_.reshape(-1, 100, 100), r
atio=3)
```

클러스터 개수 (k) 선택

적절한 클러스터 개수를 찾기 위해 엘보우 방법을 사용

- 클러스터 개수를 변화시키며 이너셔(inertia)를 계산하고, 이를 그래프로 그려 꺾이는 지점을 찾기
- 이 지점에서 클러스터 개수 증가의 효과가 감소하기 때문에 최적의 k값을 결정하는 기준

```
inertia = []
for k in range(2, 7):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(fruits_2d)
    inertia.append(km.inertia_)

plt.plot(range(2, 7), inertia)
plt.xlabel('k')
plt.ylabel('inertia')
plt.show()
```

▼ 6-3 주성분 분석

주성분 분석(PCA)을 이용한 차원 축소 : 데이터의 특성 수(차원)를 줄여 저장 공간 절약 및 모델 성능 향상을 도모하는 비지도 학습 기법

1. 차원 축소

- 고차원 데이터에서 일부 특성을 선택해 데이터 크기를 줄여, 모델 학습과 예측 성능을 높임
- PCA는 데이터를 설명하는 주요 특성(주성분)을 찾아 차원을 줄이고 손실을 최소화하여 원본 데이터를 어느 정도 복원 가능

2. PCA 소개

- PCA는 데이터의 분산이 큰 방향을 찾아 주성분으로 정의
- 각 주성분 벡터는 원본 데이터의 특성 수와 동일한 길이를 가지며, 데이터의 주요한 분산 정보를 담고 있어 차원 축소 후에도 원본 데이터의 주요 특성을 잘 표현함

3. PCA 적용

- `n_components=50`을 설정하여 주성분 50개를 추출한 후 데이터를 10000차원에서 50차원으로 축소
- 차원 축소 후, `inverse_transform()` 메서드를 사용해 데이터를 10000차원으로 재구성하여 원본 이미지와 비교했을 때 품질 손실이 거의 없음

4. 설명된 분산

- 각 주성분이 원본 데이터의 분산을 얼마나 잘 설명하는지를 나타내는 값
- 50개의 주성분으로 원본 데이터의 92% 이상의 분산을 설명
- 시각화하여 주성분 수를 줄일 때 대부분의 분산이 상위 몇 개의 주성분에 의해 설명됨을 확인

5. 차원 축소 후 학습 성능

- 원본 데이터와 차원 축소된 데이터를 로지스틱 회귀 모델에 사용해 비교
  - 원본 데이터 (10000차원): 정확도 99.7%, 훈련 시간 약 0.98초
  - 차원 축소 데이터 (50차원): 정확도 100%, 훈련 시간 약 0.04초
- 축소된 데이터는 학습 시간을 크게 단축, 차원이 줄어도 정확도는 유지

6. 클러스터링

- 2차원으로 줄인 데이터에 K-평균 클러스터링을 적용하여 3개의 클러스터를 찾았고, 시각화한 결과에서 클러스터들이 잘 구분되는 것을 확인