

# 8장 차원 축소

# 주요 내용

- 차원의 저주
- 차원 축소를 위한 접근법
  - 사영
  - 다양체 학습
- 사영 기법 알고리즘
  - PCA(주성분 분석)
  - 임의 사영
- 다양체 학습 알고리즘
  - LLE(국소적 선형 임베딩)

# 기본 아이디어

- 차원의 저주: 샘플의 특성이 너무 많으면 학습이 매우 어려워짐.
- 차원 축소: 특성 수를 (크게) 줄여서 학습 불가능한 문제를 학습 가능한 문제로 만드는 기법
- 차원 축소로 인한 정보손실을 어느 정도 감안하면서 훈련 속도와 성능을 최대로 유지하는 것이 목표

# 활용 예제

- MNIST 데이터셋
  - 사진의 중앙에만 집중해도 숫자 인식에 별 문제 없음.
  - 주성분 분석(PCA) 기법을 이용하여 784개 픽셀 대신 154개만 대상으로 충분히 학습 가능
- 데이터 시각화
  - 군집 같은 시각적인 패턴을 감지하여 데이터에 대한 통찰 얻을 수 있음.
  - 차원을 2, 3차원으로 줄이면 그래프 시각화 가능

## 8.1. 차원의 저주

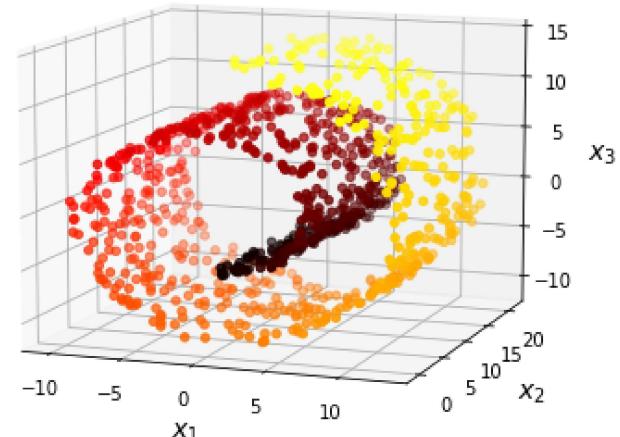
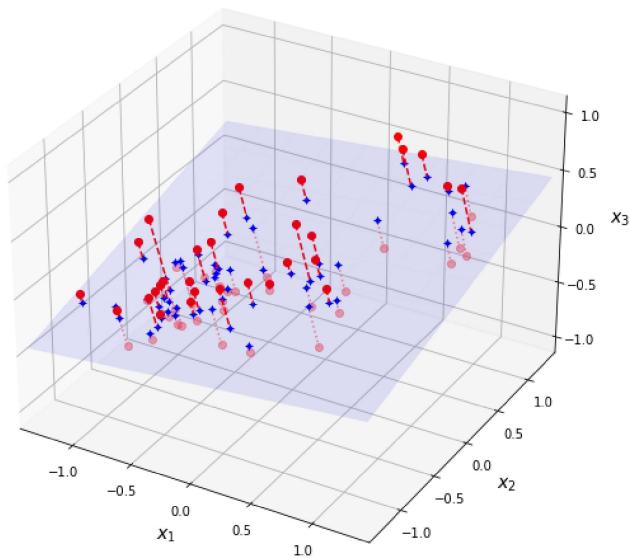
## 차원의 저주

- 벡터의 차원에 해당하는 특성 수가 커질 수록 두 샘플 사이의 거리가 매우 커져서 과대적합 위험도 커짐.
- 이유: 새로운 데이터 샘플이 주어졌을 때 훈련셋에 포함된 샘플들과의 거리가 일반적으로 매우 멀어서 기존 값들을 이용한 추정(예측)이 어렵기 때문.
- 해결책: 샘플 수 늘리기. 하지만 고차원의 경우 충분히 많은 샘플 수를 준비하는 일이 매우 어렵거나 사실상 불가능.

## 8.2. 차원 축소 기법

## 기본 아이디어

- 모든 훈련 샘플이 고차원 공간의 일부인 저차원 부분공간에 가깝게 놓여 있는 경우가 일반적으로 발생
- 예제: 아래 두 이미지 모두 3차원(3D) 공간상의 데이터셋을 보여줌. 하지만 데이터셋 자체는 2차원 형식을 따름.

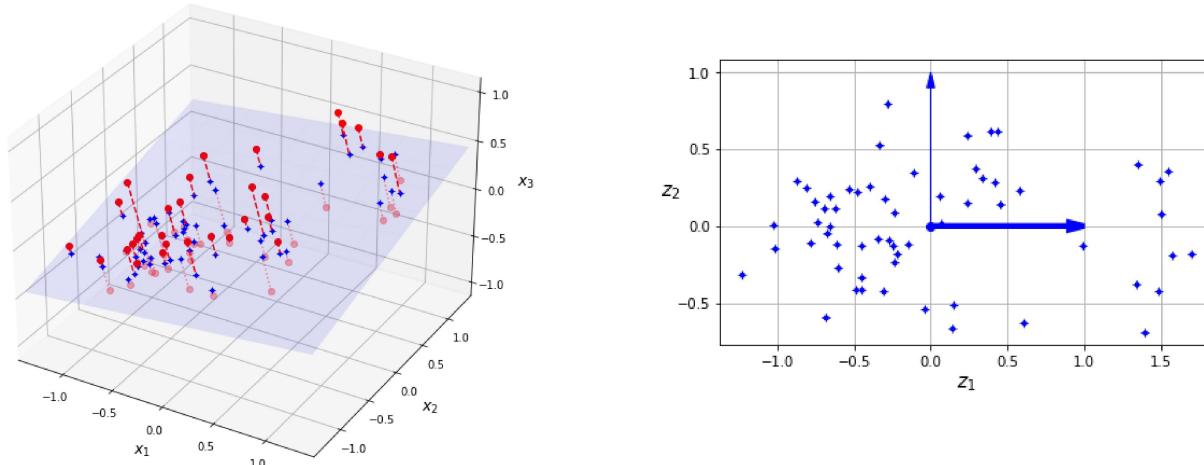


## 대표적인 차원 축소 기법

- 사영 projection
- 다양체 학습 manifold learning

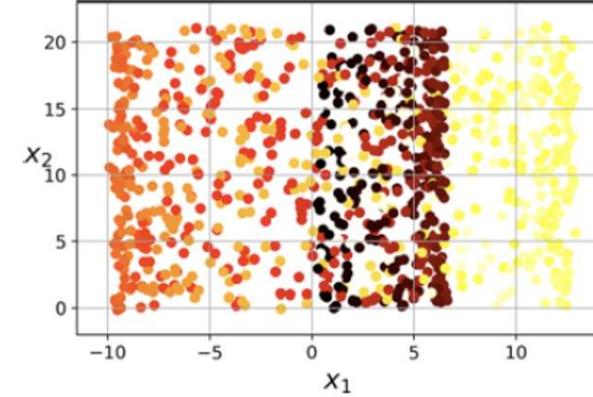
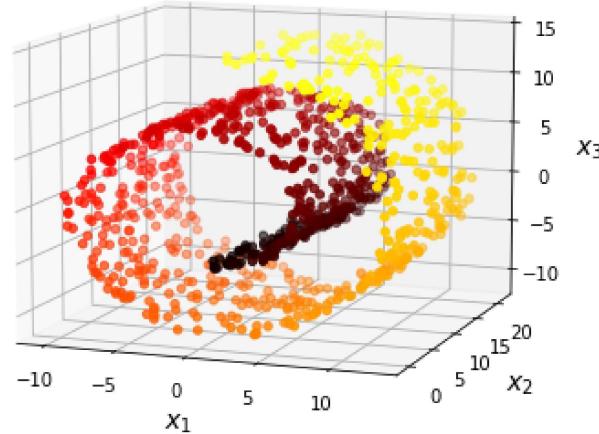
# 사영 기법

- $n$ 차원 공간에 존재하는 데이터세를 낮은  $d$ 차원 공간으로 사영하기.
- 예제: 아래 왼쪽 3차원에 존재하는 적절한 2차원 평면으로 사영하면 적절한 2차원 상의 이미지를 얻게됨. 오른쪽 2차원 이미지에 사용된 축  $z_1$ 과  $z_2$ 를 적절하게 찾는 게 주요 과제임.



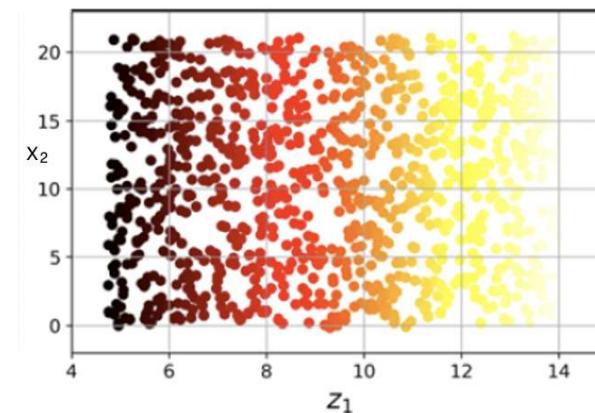
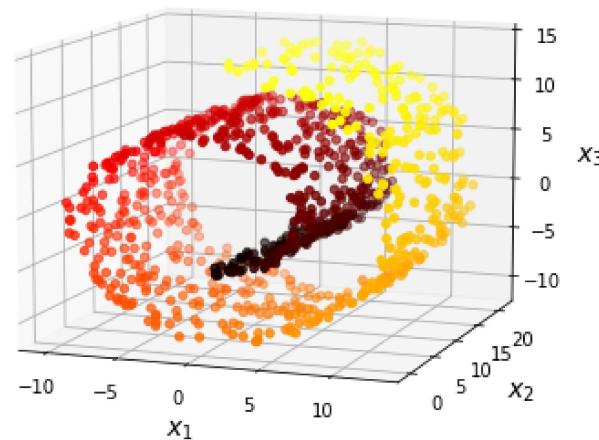
## 부적절한 사영

- 사영이 경우에 따라 보다 복잡한 결과를 낼 수 있음.
- 롤케이크를  $x_1$ 과  $x_2$  축으로 사영하면 샘플 구분이 보다 어려워짐.



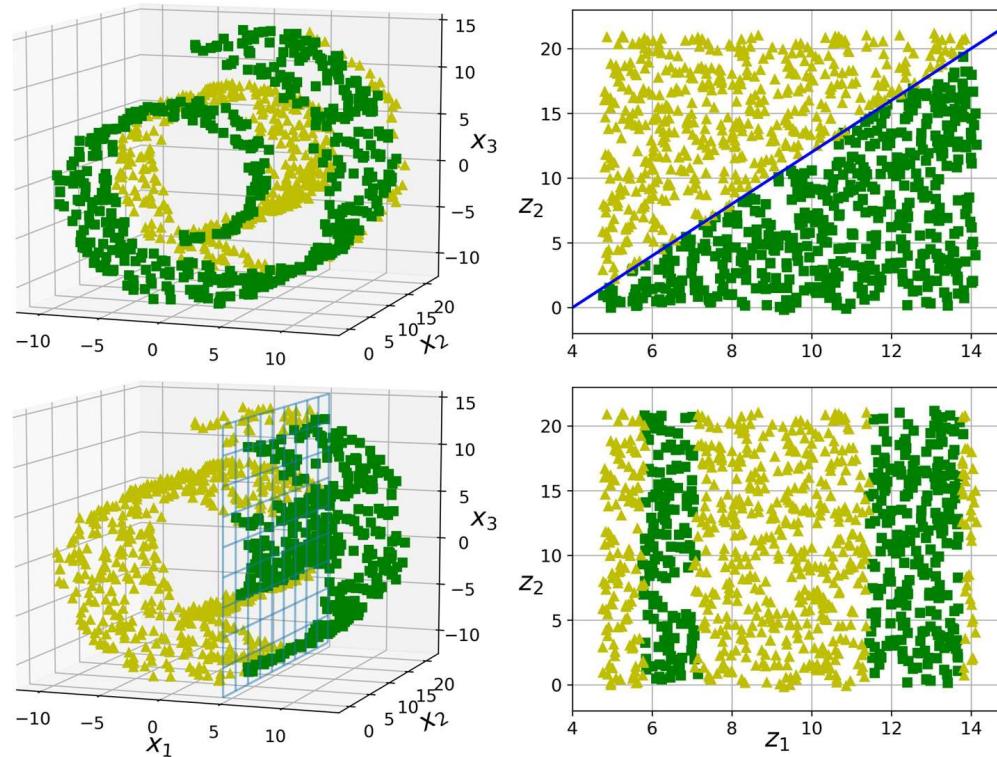
## 다양체 학습

- 롤케이크의 경우 사영 보다는 돌돌 말린 것을 펼치면 보다 적절한 2차원 이미지를 얻게 됨.



# 다양체 가설

- 대부분의 고차원 데이터셋이 더 낮은 차원의 다양체에 가깝다는 가설
- 저차원의 다양체 공간으로 차원 축소를 진행하면 보다 간단한 다양체가 된다는 가설과 함께 사용되지만 일반적으로 사실 아님. (아래 그림 참조)



### 8.3. PCA(주성분 분석)

## 아이디어

- 훈련 데이터에 가장 가까운 초평면(hyperplane)에 데이터셋을 사영하는 기법
- 주성분 분석(PCA, Principal Component Analysis)이 핵심.
- 분산 보존 개념과 주성분 개념이 중요함.

## 초평면

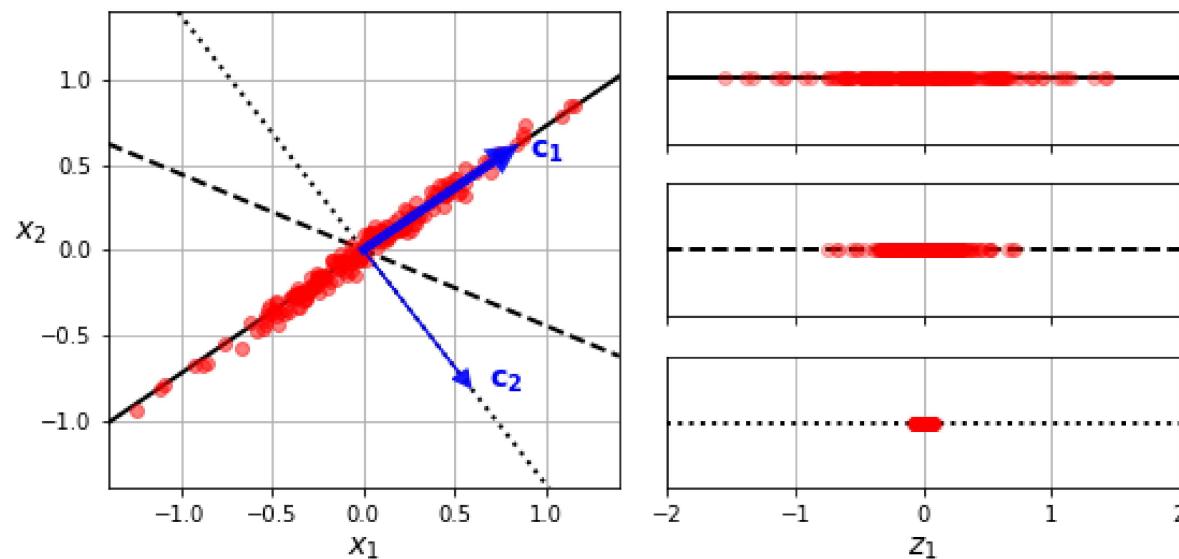
- 초평면 hyperplane: 3차원 이상의 고차원에 존재하며 아래의 방정식을 만족시키는 벡터들의 집합.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + c = 0$$

- 설명
  - $n=1$  인 경우: 1차원 공간에 존재하는 점
  - $n=2$  인 경우: 2차원 공간에 존재하는 직선
  - $n=3$  인 경우: 3차원 공간에 존재하는 평면
  - $n \geq 4$  인 경우:  $n$ 차원 공간에 존재하는 초평면

## 분산 보존

- 분산 보존: 저차원으로 사영할 때 데이터셋의 분산이 최대한 유지되도록 축을 지정 해야 함.
- 예제: 아래 그림에서  $c_1$  벡터가 위치한 실선 축으로 사영하는 경우가 분산을 최대한 보존함.  $c_1$ 에 수직이면서 분산을 최대로 보존하는 축은  $c_2$ .



## 주성분

- 첫째 주성분: 분산을 최대한 보존하는 축
- 둘째 주성분: 첫째 주성분과 수직을 이루면서 첫째 주성분이 담당하지 않는 분산을 최대한 보존하는 축
- 셋째 주성분: 첫째, 둘째 주성분과 수직을 이루면서 첫째, 둘째 주성분이 담당하지 않는 분산을 최대한 보존하는 축
- ...

## 특잇값 분해(SVD)

- 데이터셋의 주성분은 특잇값 분해(SVD) 기법을 이용하면 쉽게 계산 가능.
- 찾아진 초평면으로의 사영 또한 쉽게 계산됨.
- 데이터셋이 크거나 특성이 많으면 계산이 매우 오래 걸릴 수 있음.

## 사이킷런의 PCA 모델

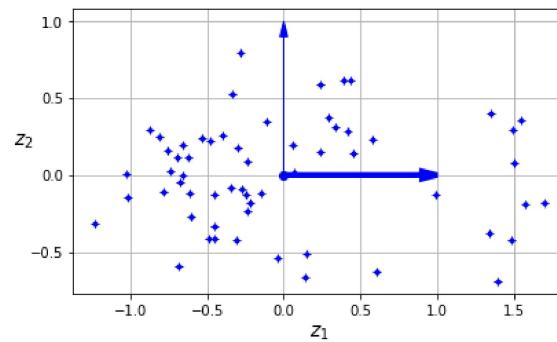
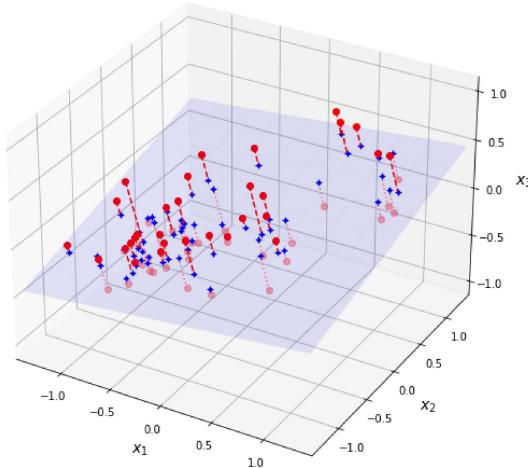
- 사이킷런의 PCA 모델: SVD 기법 활용
- 예제: 데이터셋의 차원을 2로 줄이기

```
from sklearn.decomposition import PCA  
  
pca = PCA(n_components = 2)  
X2D = pca.fit_transform(X)
```

# 설명 분산 비율

- `explained_variance_ratio_` 속성 변수: 각 주성분에 대한 원 데이터셋의 분산 비율 저장
- 예제: 아래 사영 그림에서 설명된 3차원 데이터셋의 경우.
  - $z_1$  축: 75.8%
  - $z_2$  축: 15.2%

```
>>> pca.explained_variance_ratio_
array([0.7578477 , 0.15186921])
```

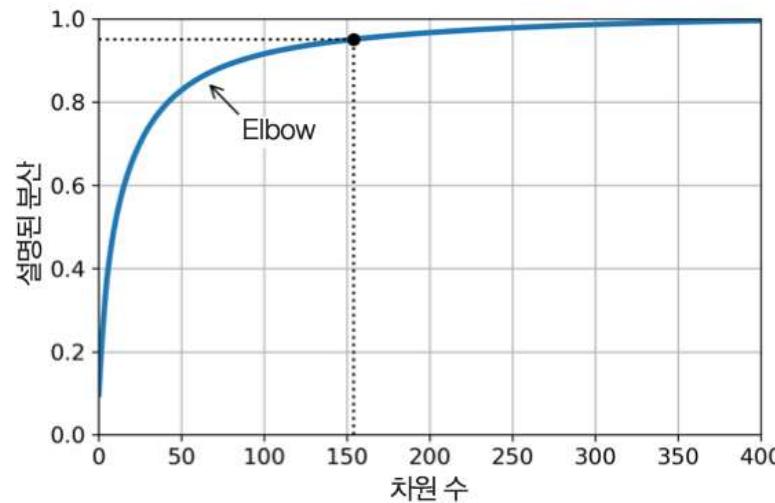


## 적절한 차원

- 적절한 차원: 밝혀진 분산 비율의 합이 95% 정도 되도록 하는 주성분들로 구성
- 데이터 시각화 목적인 경우: 2개 또는 3개

## 설명 분산 비율 활용

- 샘플의 차원과 설명 분산 비율의 합 사이의 그래프 활용
- 설명 분산의 비율의 합의 증가가 완만하게 변하는 지점(elbow)에 주시할 것.
- MNIST 데이터셋의 주성분 분석을 통해 95% 정도의 분산을 유지하려면 154개 정도의 주성분만 사용해도 됨.



## PCA 활용 예제: 파일 압축

- 파일 압축 용도로 PCA를 활용할 수 있음.
- 차원 축소 결과:
  - 784차원을 154 차원으로 줄임.
  - 유실된 정보: 5%
  - 크기: 원본 데이터셋 크기의 20%
- 원본과의 비교: 정보손실 크지 않음 확인 가능

Original	Compressed
4 2 9 3 1	4 2 9 3 1
5 7 1 4 3	5 7 1 4 3
7 9 1 0 8	7 9 1 0 8
0 9 9 1 4	0 9 9 1 4
5 1 7 6 1	5 1 7 6 1

## 랜덤 PCA

- 주성분 선택을 위해 사용되는 SVD 알고리즘을 확률적으로 작동하도록 만드는 기법
- 보다 빠르게 지정된 개수의 주성분에 대한 근삿값을 찾아줌.

## 점진적 PCA

- 훈련세트를 미니배치로 나눈 후 IPCA(incremental PCA)에 하나씩 주입 가능
- 온라인 학습에 적용 가능
- `partial_fit()` 활용에 주의할 것.

```
from sklearn.decomposition import IncrementalPCA  
  
n_batches = 100  
inc_pca = IncrementalPCA(n_components=154)  
for X_batch in np.array_split(X_train, n_batches):  
    inc_pca.partial_fit(X_batch)  
  
X_reduced = inc_pca.transform(X_train)
```

## 넘파이의 memmap() 클래스 활용

- 바이너리 파일로 저장된 (매우 큰) 데이터셋을 마치 메모리에 들어있는 것처럼 취급 할 수 있는 도구 제공
- 이를 이용하여 미니배치/온라인 학습 가능

```
# memmap 생성
filename = "my_mnist.mmap"
X mmap = np.memmap(filename, dtype='float32', mode='write',
shape=X_train.shape)
X mmap[:] = X_train
X mmap.flush()

# memmap 활용
X mmap = np.memmap(filename, dtype="float32", mode="readonly").reshape(-1,
784)

batch_size = X mmap.shape[0] // n_batches
inc_pca = IncrementalPCA(n_components=154, batch_size=batch_size)
inc_pca.fit(X mmap)
```

## 8.4. 임의 사영

## 존슨-린덴슈트라우스 정리

- 고차원의 데이터를 적절한 크기의 저차원으로 임의로 사영하더라도 데이터셋의 정보를 많이 잃어버리지 않음을 보장
- 아래 부등식을 만족하는  $d$ 를 사영 공간의 차원으로 지정.
  - $m$ : 훈련셋 크기
  - $\varepsilon$ : 허용된 정보손실 정도

$$d \geq \frac{4 \log(m)}{\frac{1}{2}\varepsilon^2 - \frac{1}{3}\varepsilon^3}$$

## 사이킷런의 임의 사영 모델 1

- GaussianRandomProjection 모델

```
gaussian_rnd_proj = GaussianRandomProjection(eps=0.1, random_state=42)
X_reduced = gaussian_rnd_proj.fit_transform(X)
```

## 사이킷런의 임의 사영 모델 2

- SparseRandomProjection 모델
  - 희소 행렬sparse matrix을 사용하는 GaussianRandomProjection 모델
  - 빠르고 메모리 효율적
  - 대용량 데이터셋이 주어진 경우 유용

```
gaussian_rnd_proj = SparseRandomProjection(eps=0.1, random_state=42)
X_reduced = gaussian_rnd_proj.fit_transform(X)
```

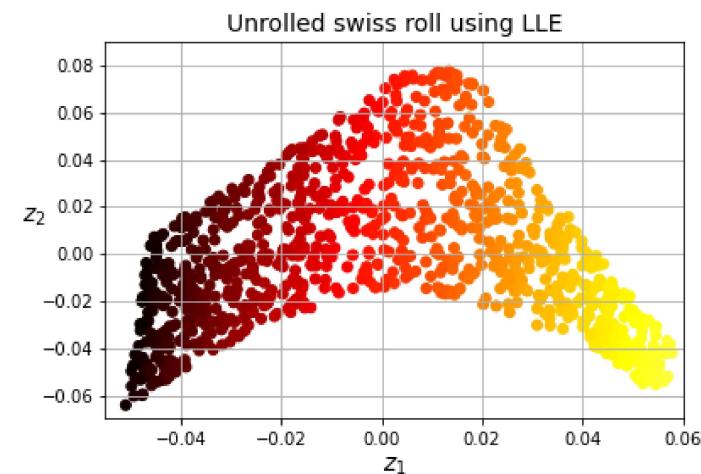
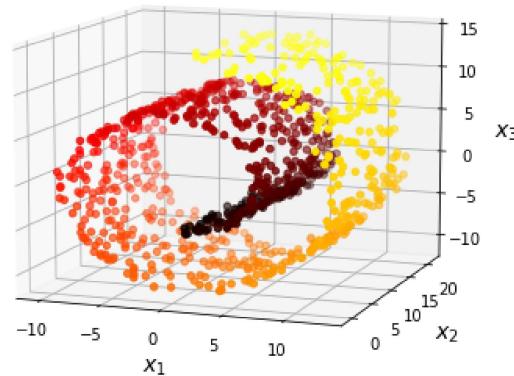
## 8.5. LLE(국소적 선형 임베딩)

## 기본 아이디어

- 대표적인 다양체 학습 기법
- 롤케이크 데이터셋의 경우처럼 전체적으로 비선형인 다양체이지만 국소적으로는 데이터가 선형적으로 연관되어 있음.
- 국소적 관계가 가장 잘 보존되는 훈련 세트의 저차원 표현 찾을 수 있음.
- 사영이 아닌 다양체 학습에 의존

## 예제: 롤케이크

```
X_swiss, t = make_swiss_roll(n_samples=1000, noise=0.2, random_state=42)
lle = LocallyLinearEmbedding(n_components=2, n_neighbors=10, random_state=42)
X_unrolled = lle.fit_transform(X_swiss)
```



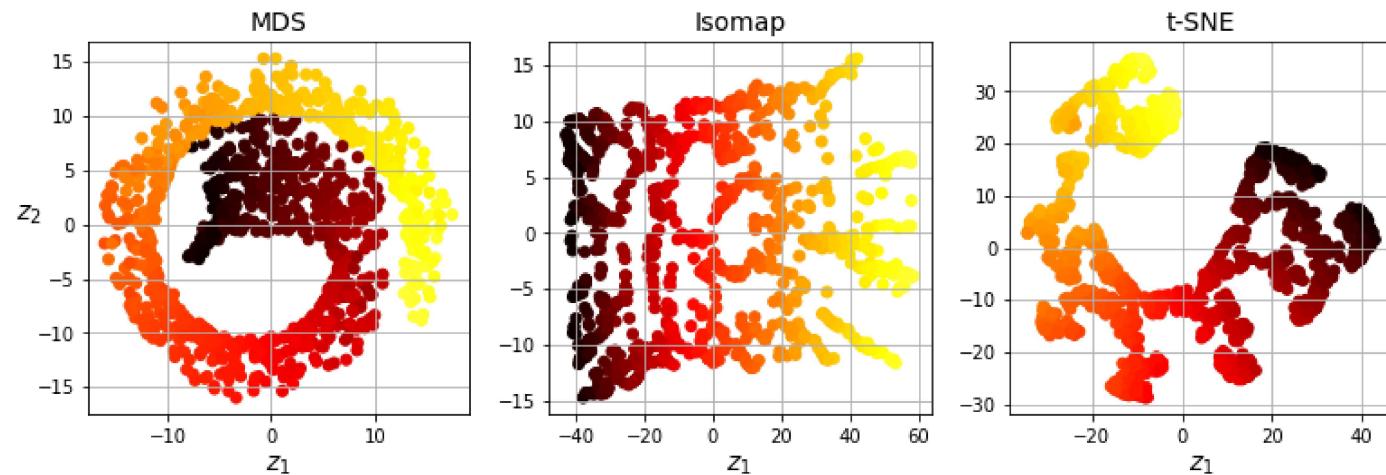
## 8.6. 기타 차원 축소 기법

## 사이킷런에서 지원하는 모델

- 다차원 스케일링 Multidimensional Scaling(MDS)
- Isomap
- t-SNE(t-Distributed Stochastic Neighbor Embedding)
- 선형 판별 분석 Linear Discriminant Analysis(LDA)
- 커널 PCA

## 예제

- 롤케이크를 각각 MDS, Isomap, t-SNE 방식으로 2차원으로 변환한 결과



## 예제

- 른케이크를 다양한 커널을 이용하여 커널 PCA로 2차원 데이터셋으로 변환한 결과

