

# Feature and Feature Vector

## Feature

- 샘플을 잘 설명하는 특징

e.g) 사람 (좋은가)

continuous: 나이, 구 ...

categorical: 성별, 직업 ...

### 1. 브트징

- 특징으로부터 두 thing의 유사도를 알 수 있다

· 영첨난 메모리 필요  
→ ex) 주민등록번호

특징을 통해 특정 샘플을 위치화 할 수 있다.

### Minst dataset에서 feature

해당 이미지가 어떤 숫자를 표현하는지

- 전통적인 머신러닝

사람이 어떤 한 특징이 주요한지 가정을 세워

그걸 바탕으로 전처리



input 으로 model 학습

장점: 사람이 해석하기 쉬움

단점: 사람이 생각치 못한 가능성이 존재할 가능성

- 현재 DL

feature vector

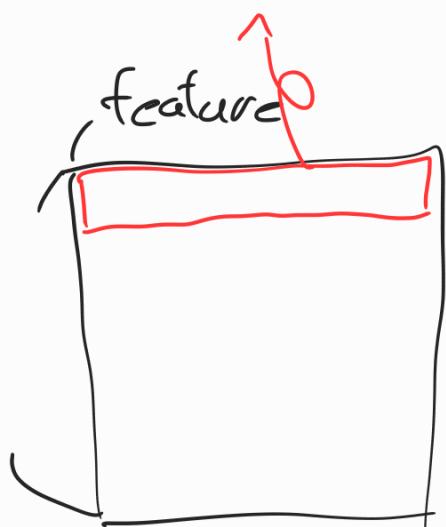
Raw data  $\rightarrow$  초소한의 전처리

<input Model 학습

장점: 구현이 어렵지 않음, 여러 특징 활용

단점: 사람이 해석하기 어렵음

data



## Feature Vector

- 각 특징들을 모아서 하나의 Vector로 만들 것
- 각 차원(열)은 어떠한 속성에 대한 level을 대응  
· level이 비슷할수록 비슷한 생김
- Feature Vector를 통해 샘플 사이의 유사도 계산할 수 있음

Continuous value → 비슷한 값 비슷한 의미

Categorical value → 비슷한 값 다른 의미

one hot encoding

→ Sparse Vector (대부분의 차원이 0)

Integer 表示

1개의 1개 n개의 0으로 이루어진 차원 벡터

$x_1 x_2 x_3 x_4$

1 0 1 0 0

- 두 샘플 사이의 유사도를 구할 수 없다

Embedding vectors

ex) word Embedding vectors

차원 축소 및 dense vector로 표현

↳ 대부분의 차원이 어려한 값을 갖음

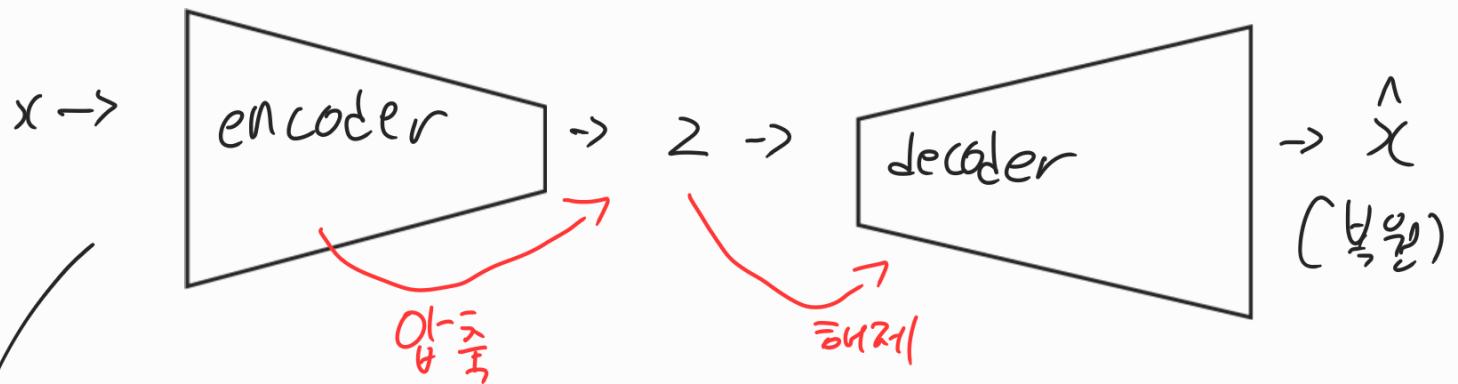
샘플이 유사한 면 거리가 가까움

Sparse vector 벡터간 유사도 계산이 어렵다

그래서

Dense vector로 표현

# Auto encoder



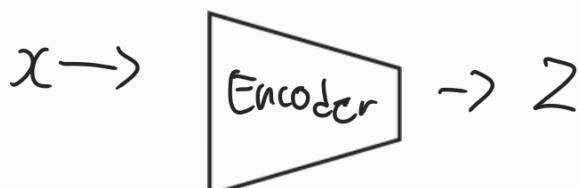
정보를 최대한 보존하여면서 ↑ 수행

특징을 추출하는 방법을 자동으로 학습

## Encoder

압축

- 필요 없는 정보(특징) 버림
- 입력에 비해 작은 차원의 output으로 구성



$z$  = 입력에 대한 feature vector  
(입력에 비해 Dense vector)

## Decoder

$z$ 를 바탕으로 입력  $x$ 와 비슷하게 복원

## Autoencoder (AE)

- 압축과 해제를 반복하여 특징을 추출을 자동으로 학습  
↳ 정보의 필요성을 구분

$$x \rightarrow z \rightarrow \hat{x}$$

$z$ 는 feature vector

인코더에 통과시키는 것은 feature vector or叫做  
Embedding 과정

# Auto Encoder (AE)

## Model

class Autoencoder

Sequential

Linear

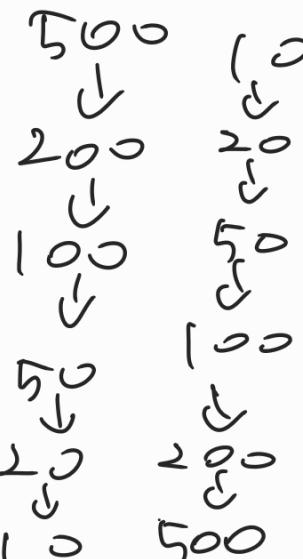
ReLU

Batch Normal

마지막 Layer 는

Activation func, Batch normal 사용 X

encoder    decoder



```
def forward(self, x):    X.Shape = [batch-size, 784]
    z = encoder(x)    Z.Shape = [  ,   , batch-size?]
    y = decoder(z)    Y.Shape = [  ,   , 784?]
    return y
```

## Data

- Matplotlib data load
- data shuffle (randperm, index-select)
- train, val data split

## train

- Model load
- optimizer, 손실함수 정의
- 전시동에 사용한 trainer ring
- train data, label 쌍

## test (predict)

with torch.no\_grad()

random index 선택  
[batch\_size, 784]

↓  
[?, 784]

↓  
[1, 784]

Show-image

# Latent-space

color map

plt.figure

↳ 100개의 data를 all에 예측한

↳ 예상되는 y값

→ ↑ 2개의 all

## Generation

if bf\_size == 2

-1 ~ 1

이 파트 구현하면서

torch.stack

한 번더 살펴보기

torch.FloatTensor

torch.clamp

torch.cat