Autoencoder

UOS Session 3

Study plan

<환경을 분석할 줄 아는 사람이 되자!>

Session 2. Machine learning (9/23)

Session 3. Logistic regression (10/7)

+ Naver 이활석님 강의자료
+ The Keras Blog
(Building Autoencoders in keras)
+ 그 외 다양한 웹 자료

Study plan

<환경을 분석할 줄 아는 사람이 되자!>

Session 2. Machine learning (9/23)

~~Session 3. Logistic regression (10/7)~~

→ **Autoencoder + project based learning**

+ Naver 이활석님 강의자료
+ The Keras Blog
(Building Autoencoders in keras)
+ 그 외 다양한 웹 자료

Contents

1. Autoencoder

2. Project based learning

   : Data crawling & database

1. Autoencoder

Autoencoder : definition

Stacking Autoencoder

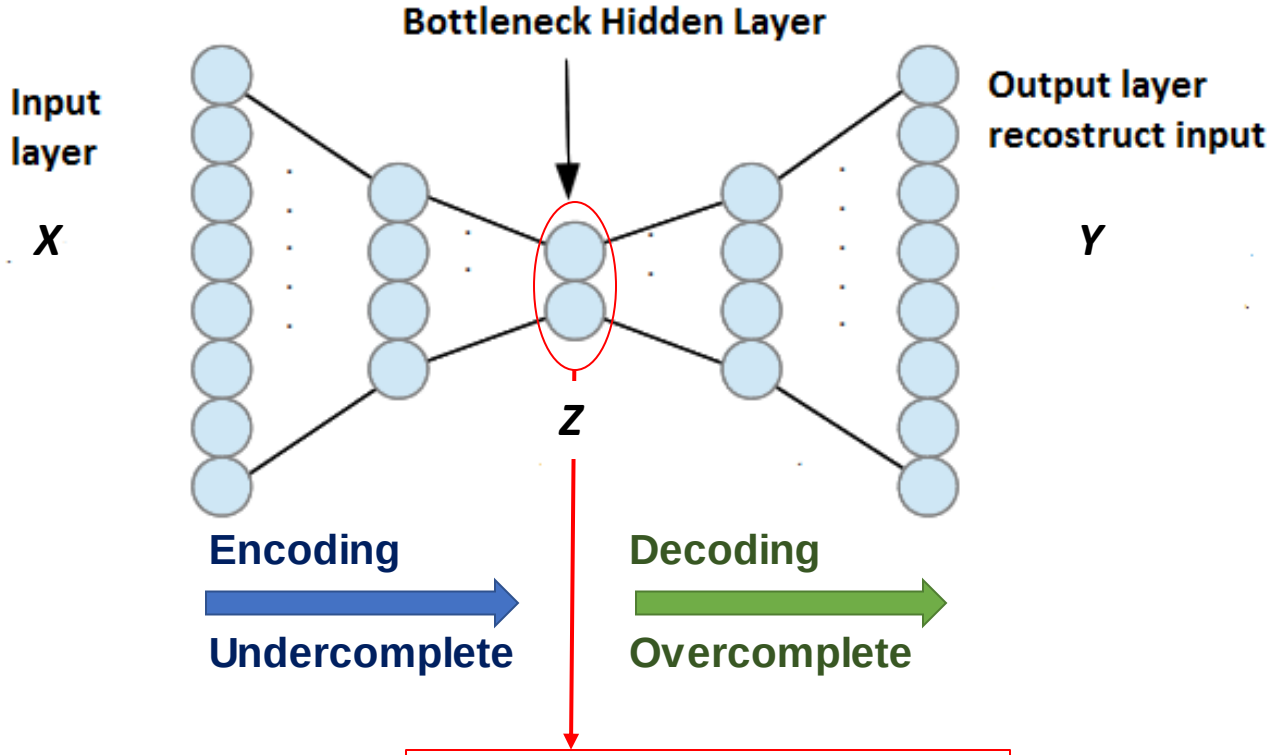
Denoising Autoencoder

Manifold learning

Autoencoder

= Auto-associators
= Diabolo networks
= Sandglass-shaped net

- Encoder = 인지 네트워크 (Recognition network)
  Input → Latent variable
- Decoder = 생성 네트워크 (Generative network)
  Latent variable → Output
- Input dimension = Output dimension
  → Reconstruction (재구성)이라고도 함
- Encoding → Decoding → Reconstructed input
  과정을 통해 input data의 중요한 feature를 학습하는 모델

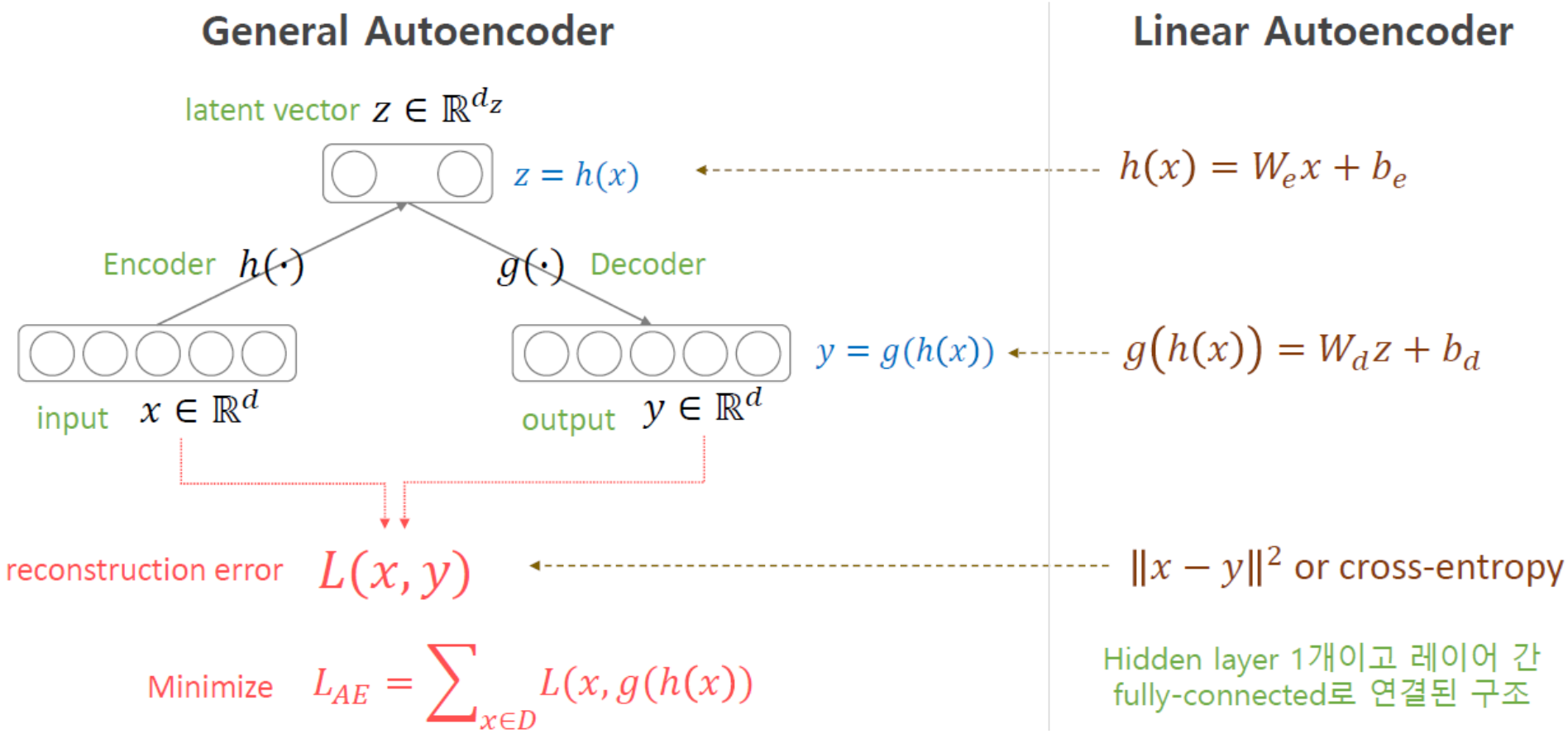


- Code
- **Latent variable**
- Feature
- Hidden representation

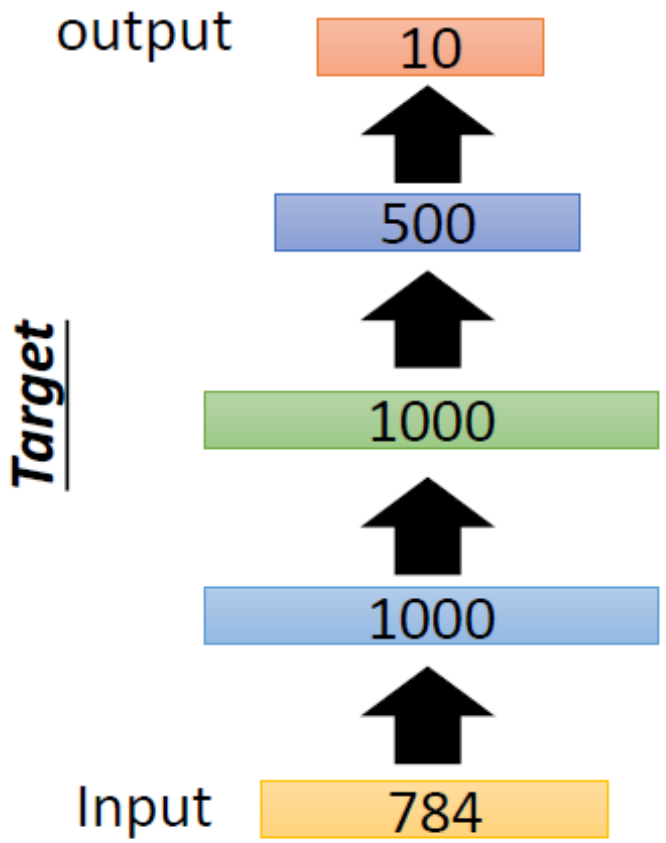
Autoencoder : characteristics

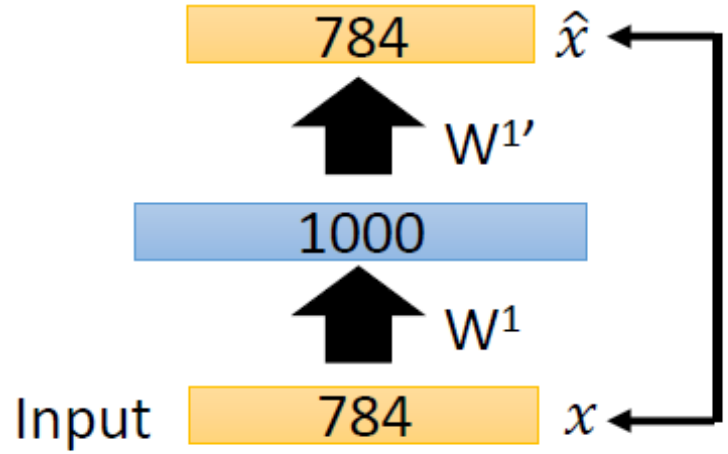
- Unsupervised learning → Supervised learning

비교사 학습문제를 교사학습문제로 바꾸어 해결 → Self-supervised learning 이라고도 함

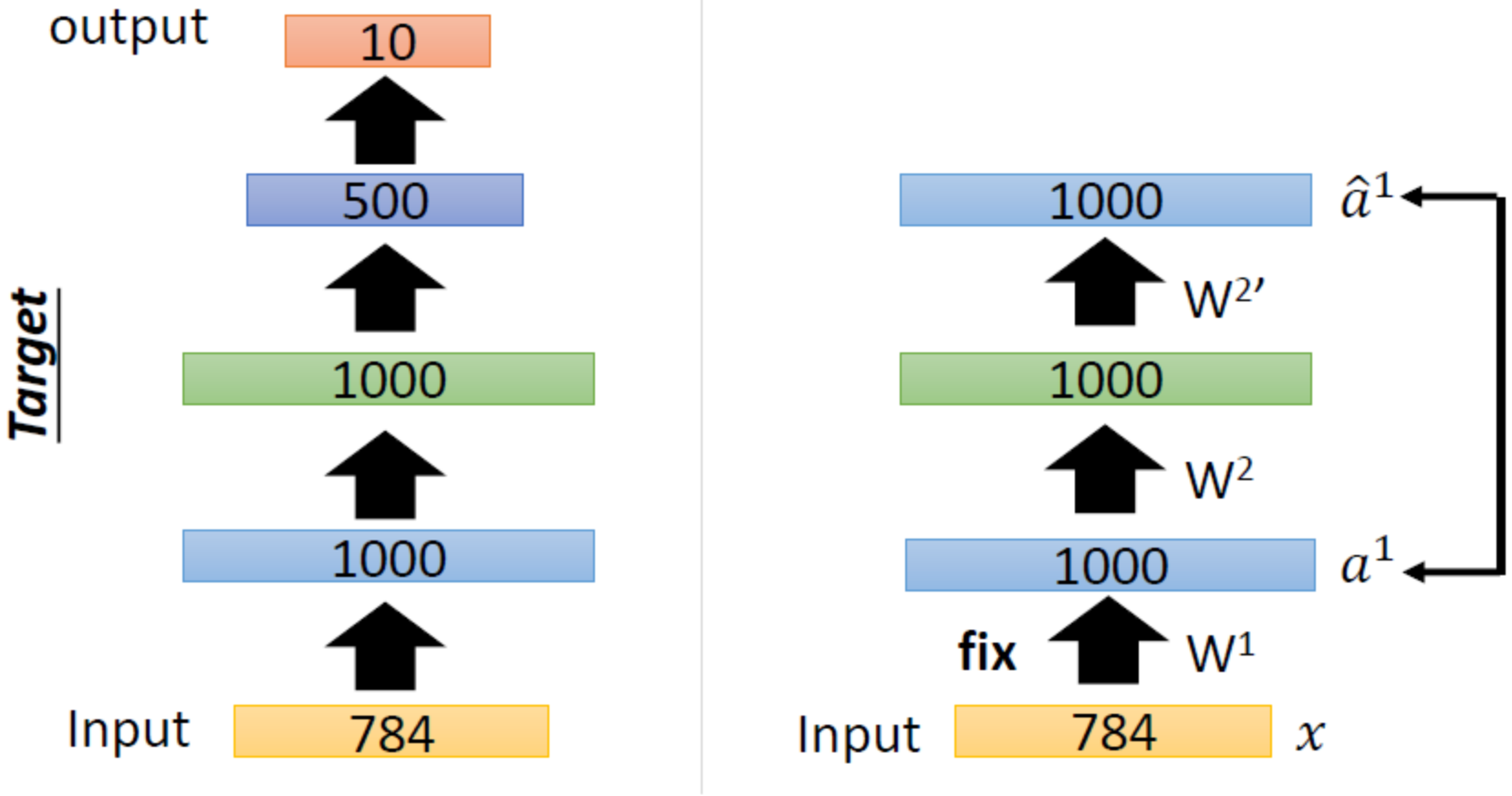


Stacking Autoencoder

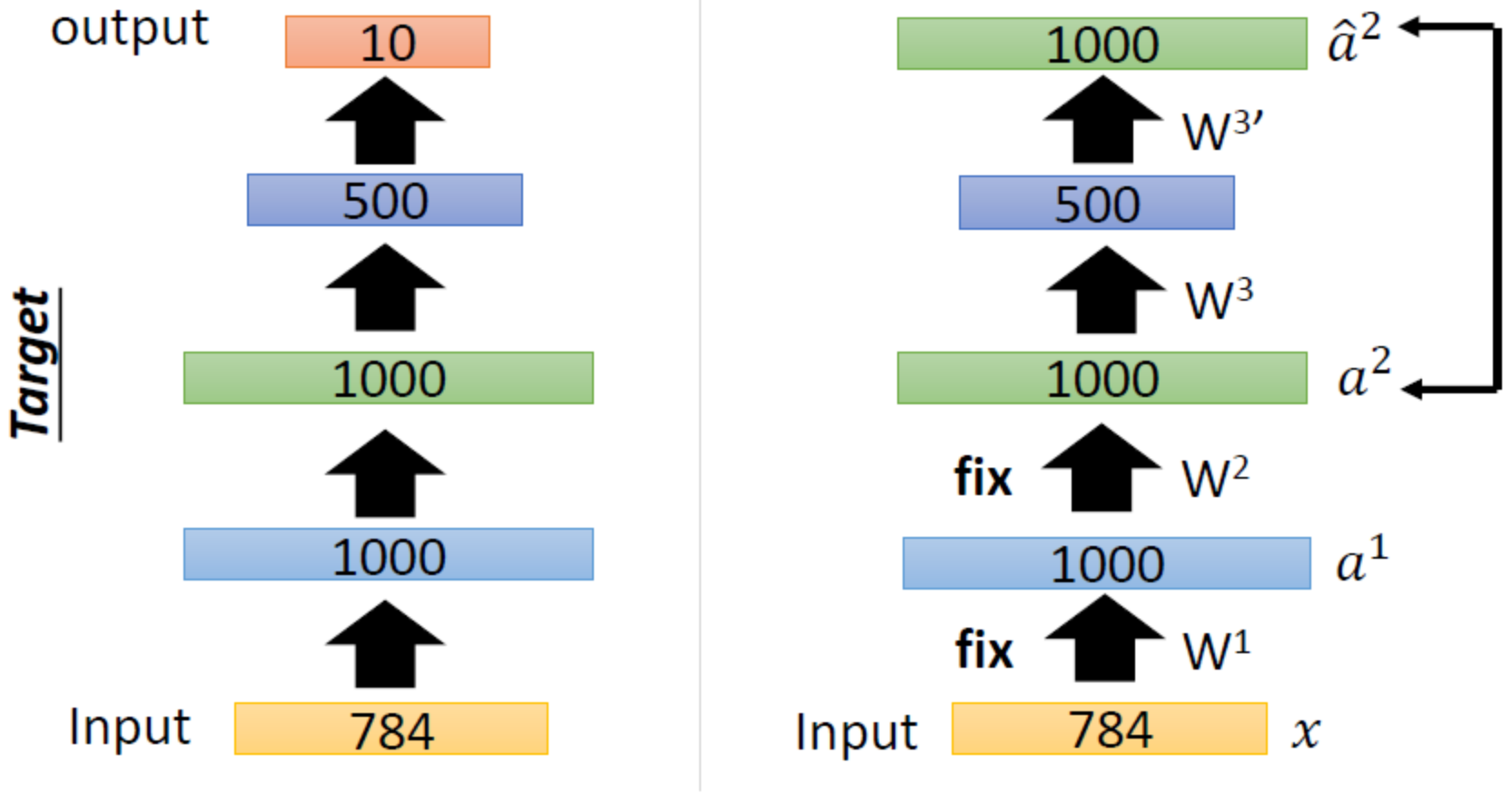




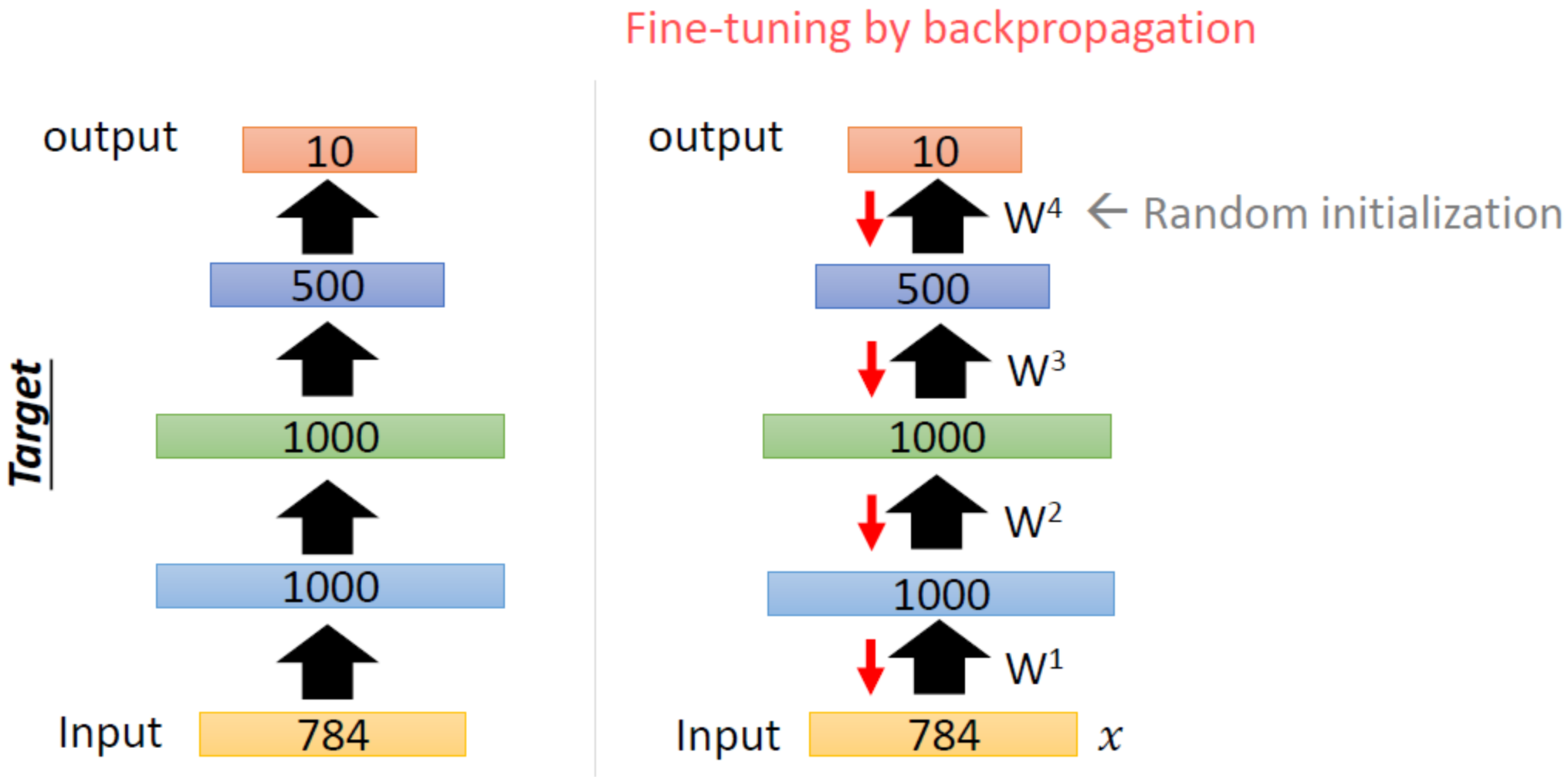
Stacking Autoencoder



Stacking Autoencoder

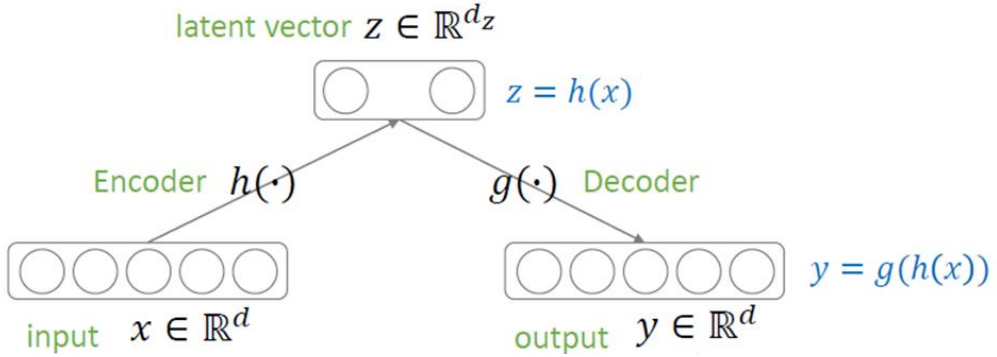


Stacking Autoencoder

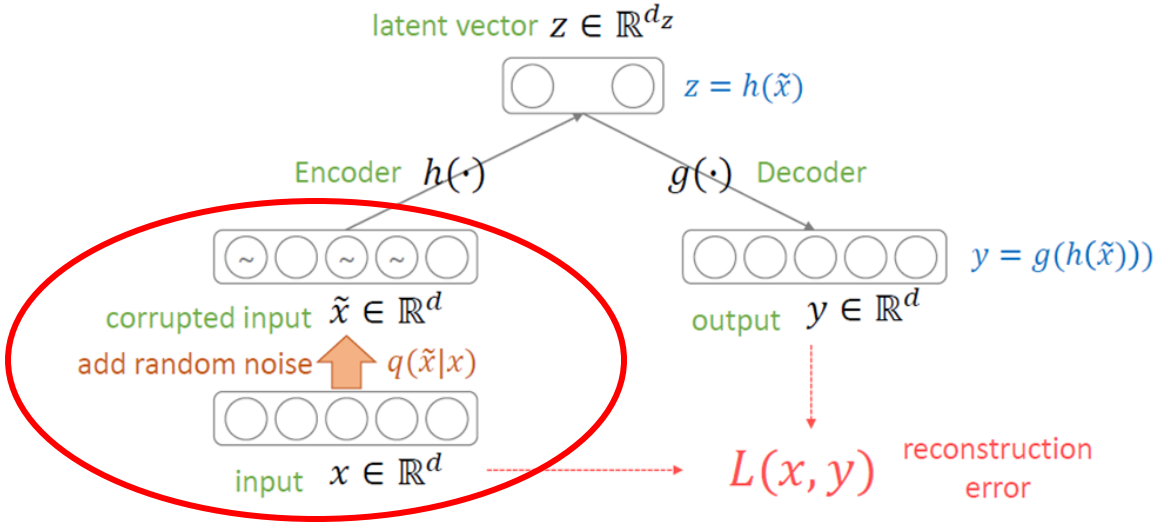


Denoising Autoencoder

- **General Autoencoder**



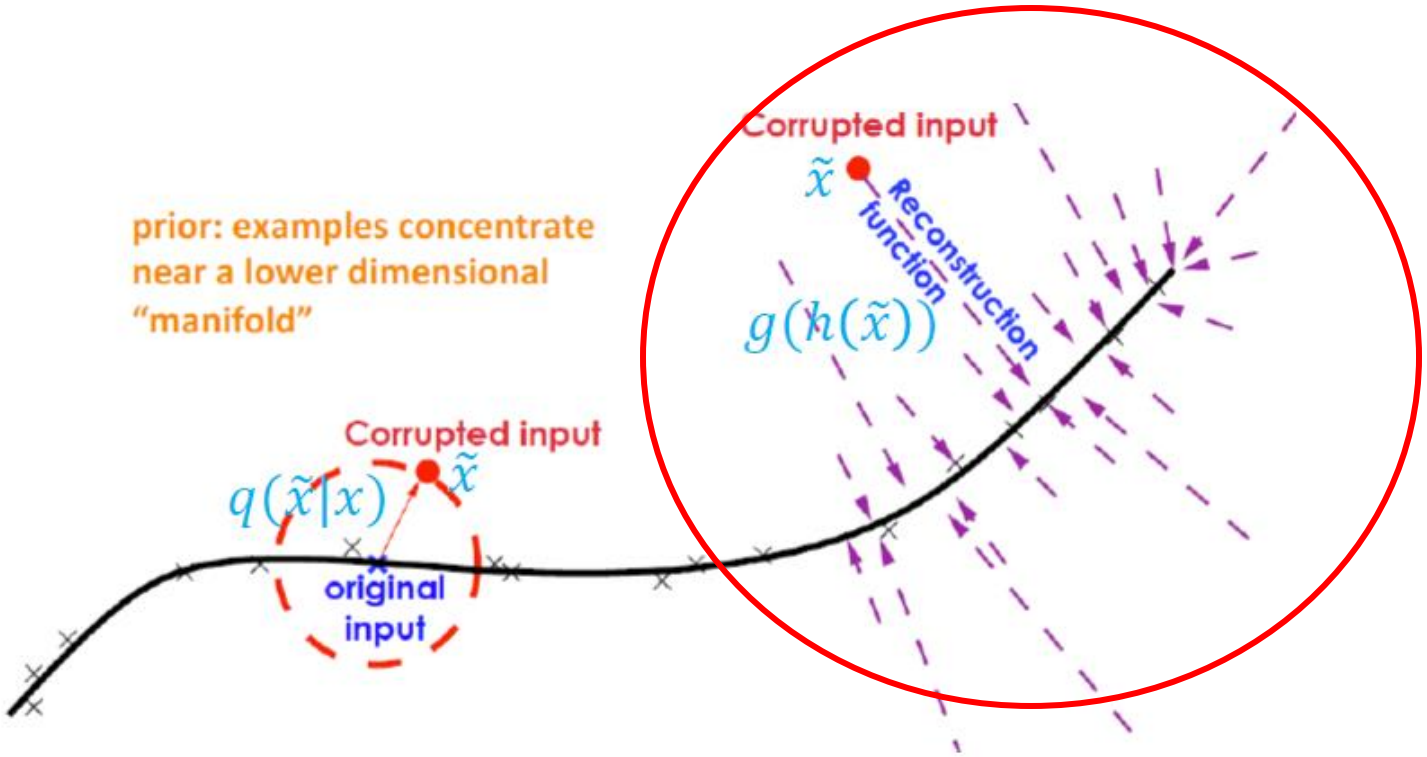
- **Denoising Autoencoder**



Minimize $L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$

- **Corrupted input → input 값의 작은 변동에 대한 저항성을 갖게됨**
- **대체적으로 더 나은 성능의 performance를 보이게 됨 (Deep neural net pre-training에서)**

Denoising Autoencoder



- Project the corrupted input back onto the manifold

Manifold learning

Data compression

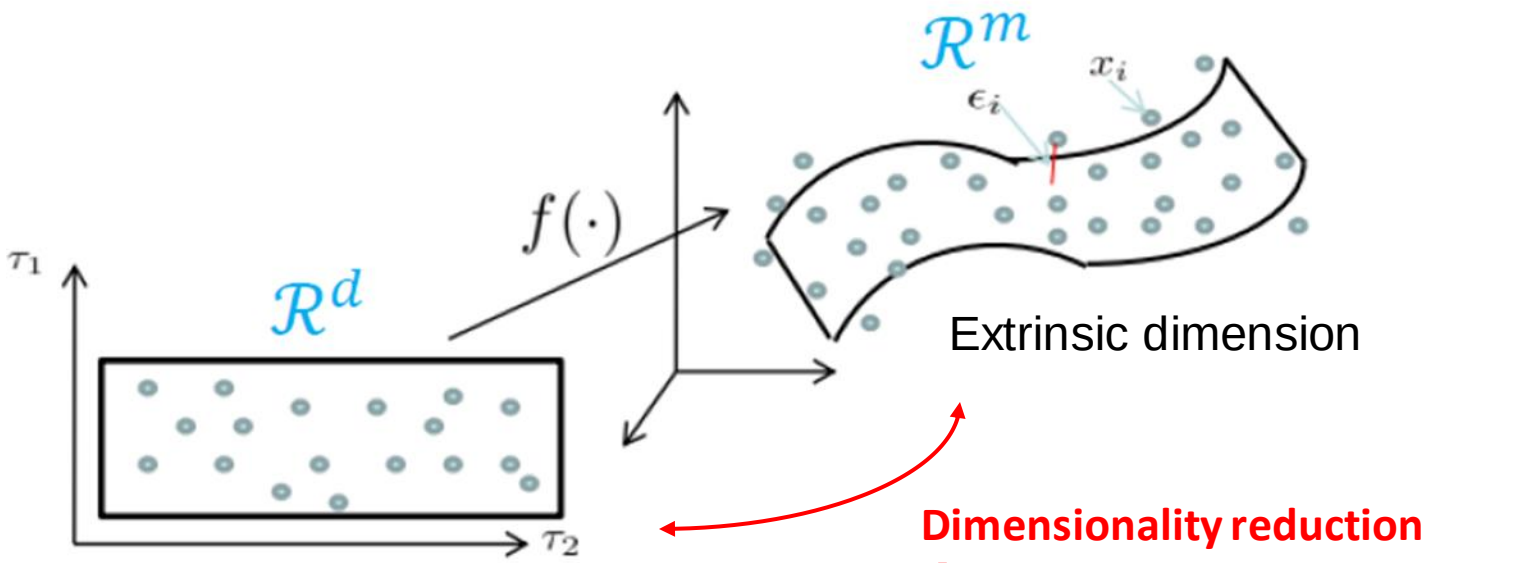
Data visualization

Curse of dimensionality

Reasonable distance metrics

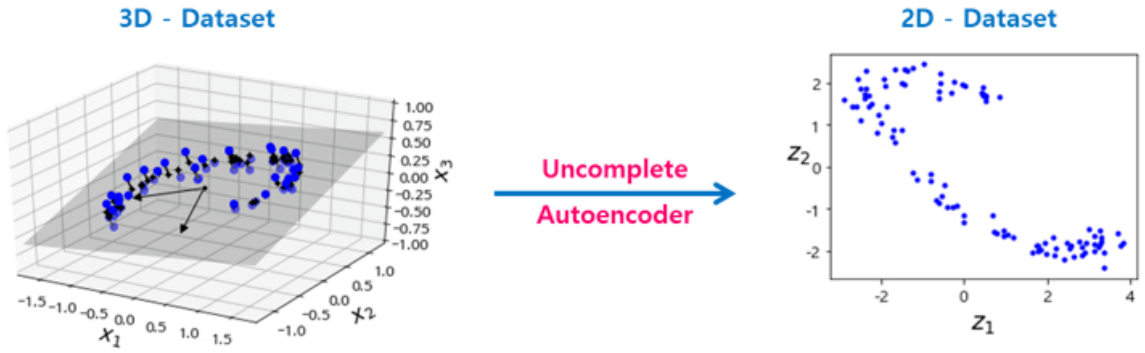
Data compression

- **Definition**
  - R^m (x_i는 noise를 포함) dimension projection to R^d projection
  - Manifold hypothesis : $p(\tau_i)$ 는 연속적, 균등 분배, 적은 noise 포함

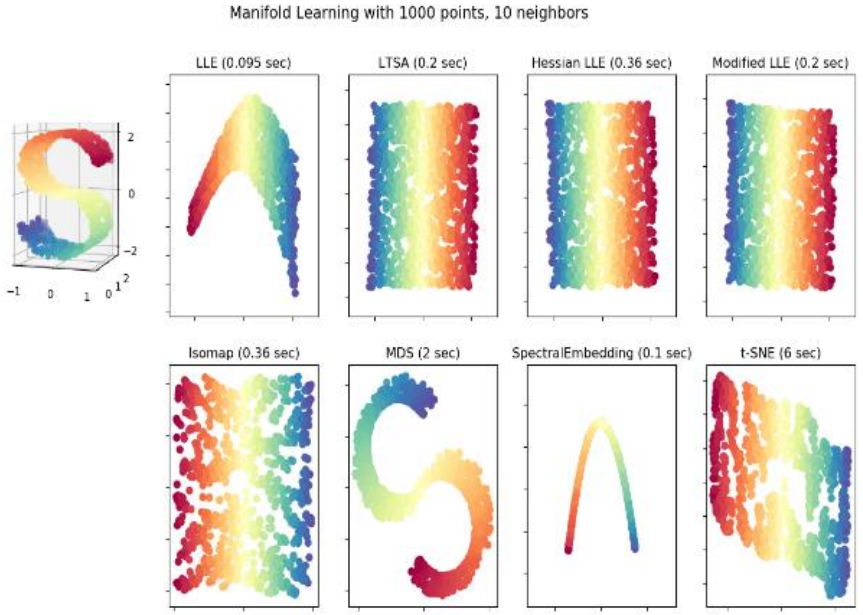


Data compression & visualization

- R^m 차원의 data를 R^2 혹은 R^3 차원으로 축소해 data간의 관계를 시각화 가능



https://excelsior-cjh.tistory.com/187



scikit-learn.org/stable/modules/manifold.html

Curse of dimensionality

- **차원의 저주**
  - 데이터의 차원 증가

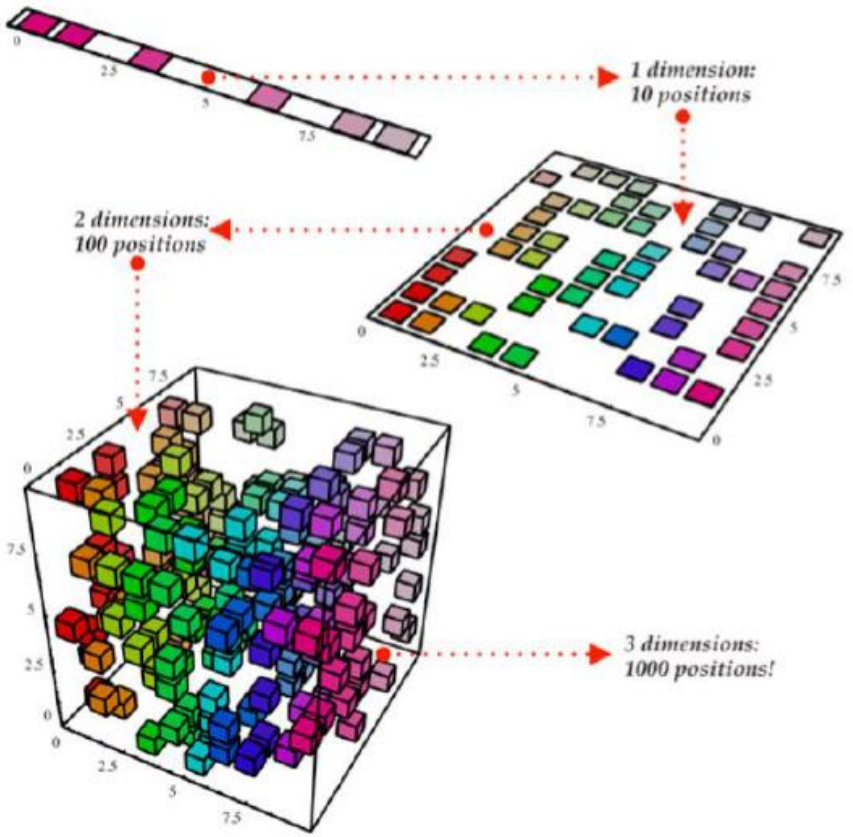
  →공간의 크기 (부피)가 기하급수적으로 증가

  → 동일한 개수의 데이터의 밀도는 급속도로 희박
  - 차원이 증가할수록 필요한 데이터의 개수는

    기하급수적으로 증가..

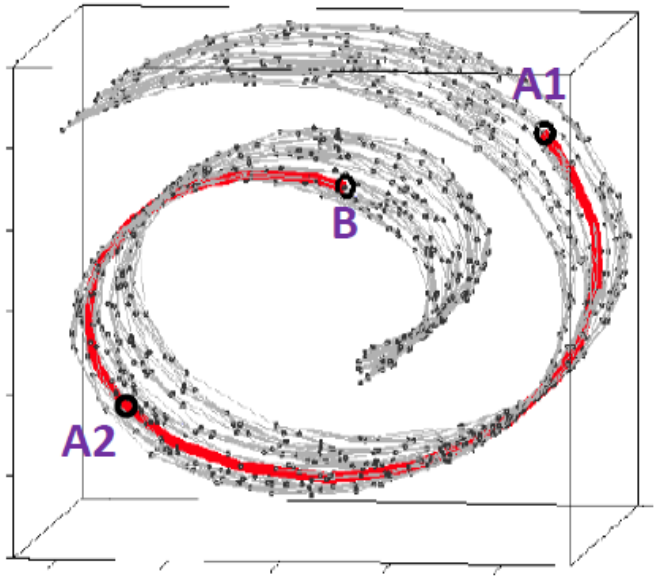
- **Manifold hypothesis**

1. 고차원의 데이터 밀도는 낮지만, 이들의 집합을 포함하는 저차원의 매니폴드가 있다.
2. 이 저차원의 매니폴드를 벗어나는 순간 급격히 밀도는 낮아진다.

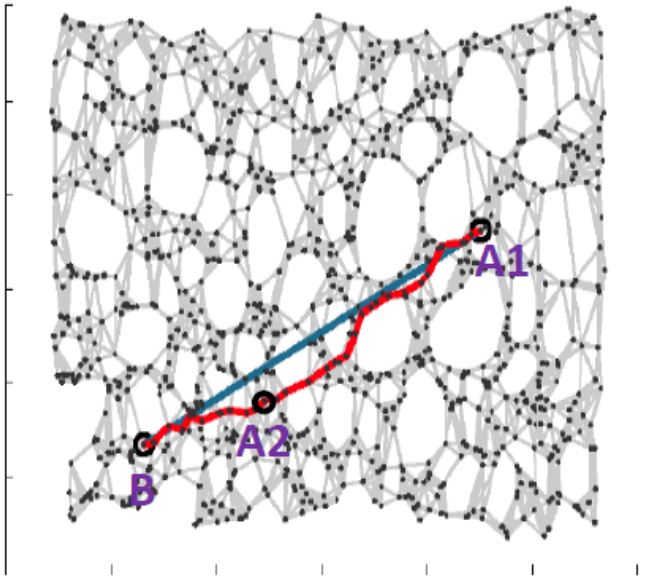


http://darkpgmr.tistory.com/145

Reasonable distance metrix



Distance in high dimension

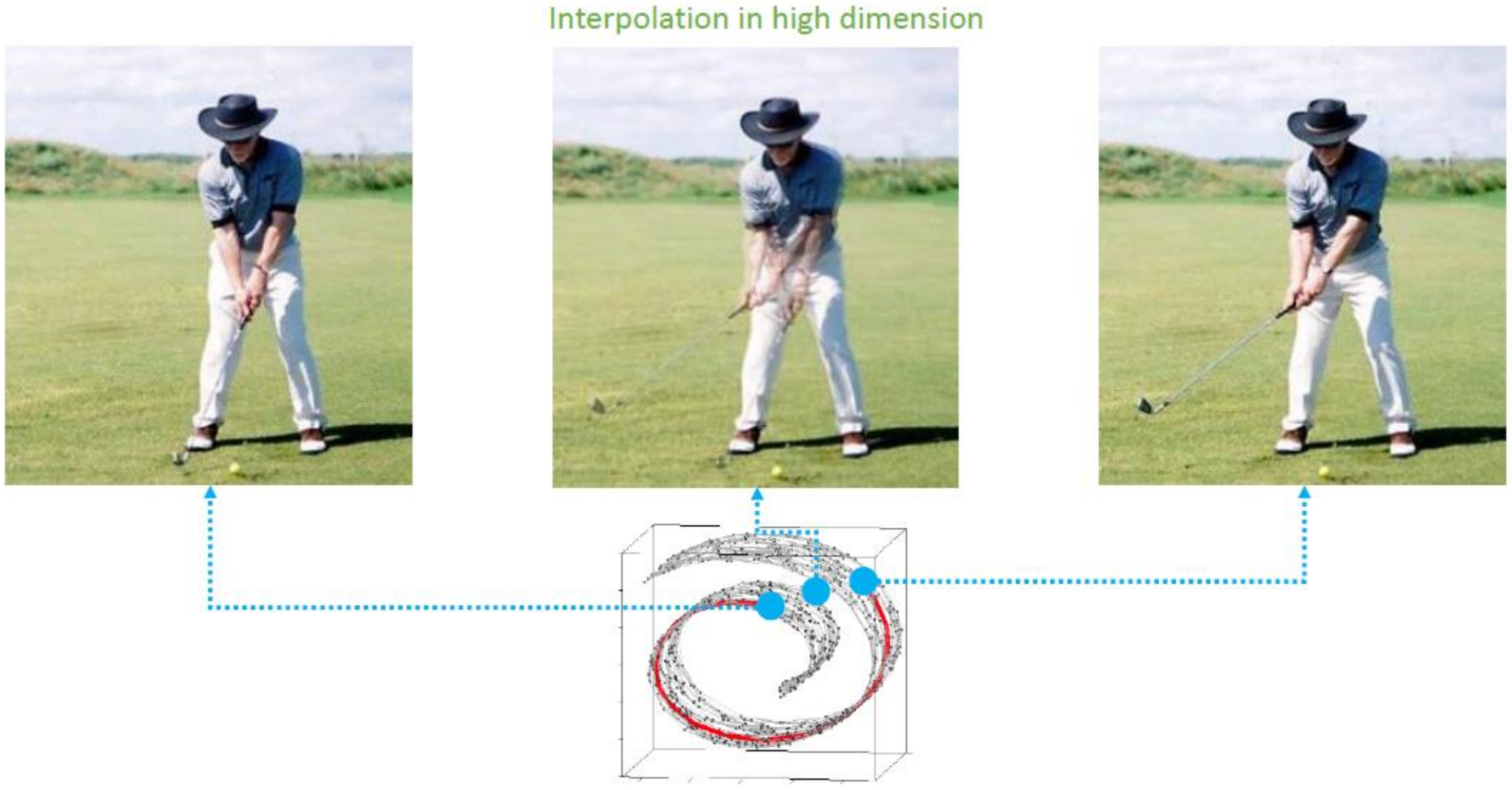


Distance in manifold

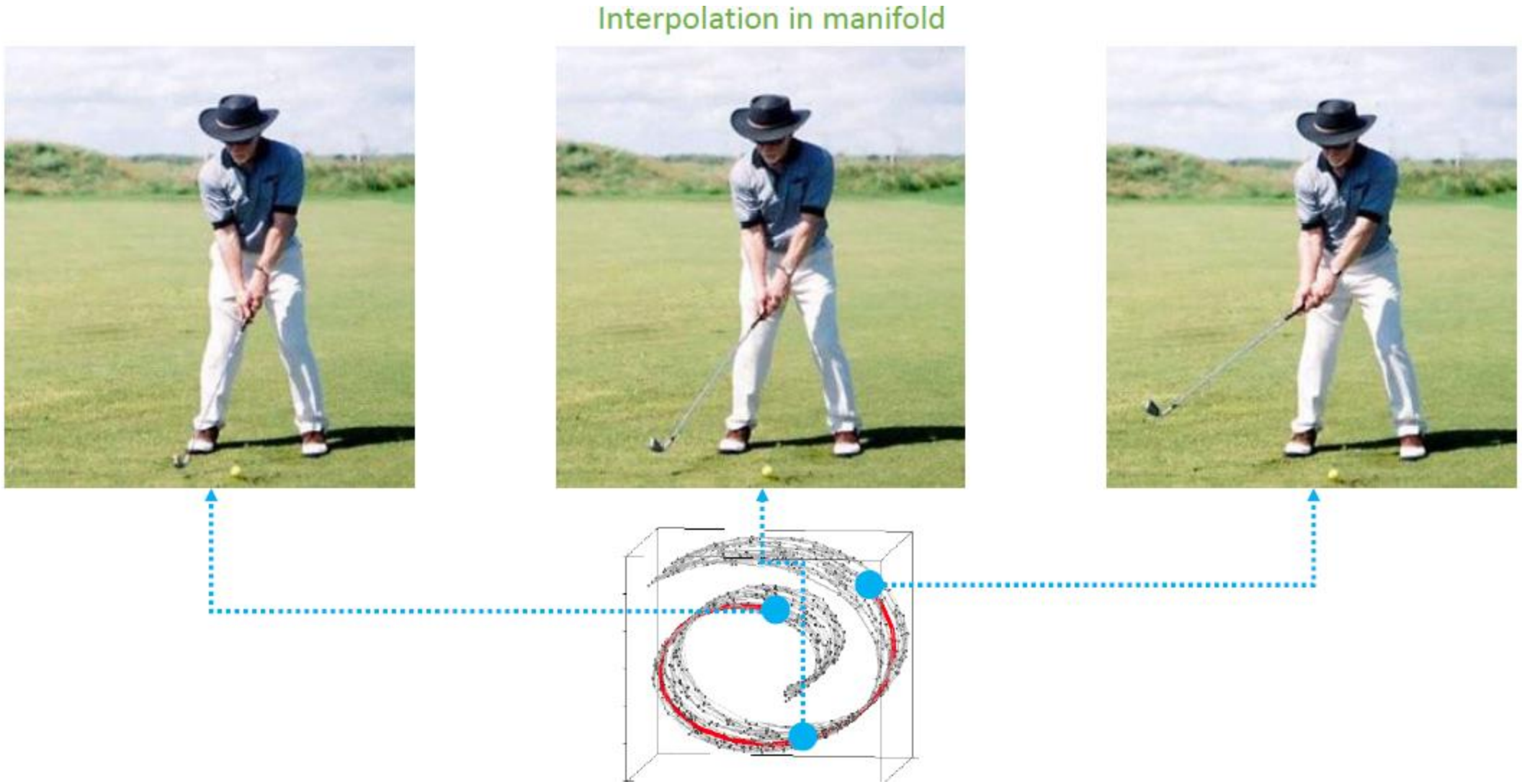
중요한 특징들을
찾았다면 이 특징을
공유하는 샘플들도
찾을 수 있어야 한다.

- **고차원 공간에서의 거리**
  - 의미적으로 가깝다고 생각되더라도 고차원공간에서 두 샘플간의 의미적 거리가 먼 경우가 있음
  - 차원의 저주로 인해 유의미한 거리 측정 방식을 찾기 어려움

Reasonable distance metrix



Reasonable distance metrix



Dimension reduction

Dimensionality Reduction

- Linear
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - etc..
- Non-Linear
  - Autoencoders (AE)
  - t-distributed stochastic neighbor embedding (t-SNE)
  - Isomap
  - Locally-linear embedding (LLE)
  - etc..

Summary

- **Autoencoder**
  - Self-supervised learning
  - Stacking Autoencoder
  - Denoising Autoencoder

- **Manifold learning**
  - Data compression
  - Data visualization
  - Curse of dimension
  - Reasonable distance metrix

2. Project : 외래어종의 개체밀도 예측

- 물환경정보시스템 data collection

  → Selenium 통해 web crawling

  - 생물측정망 : 어종별 개체밀도 (개체수/m^2)

    a) 생태계교란종 : 블루길, 큰입배스

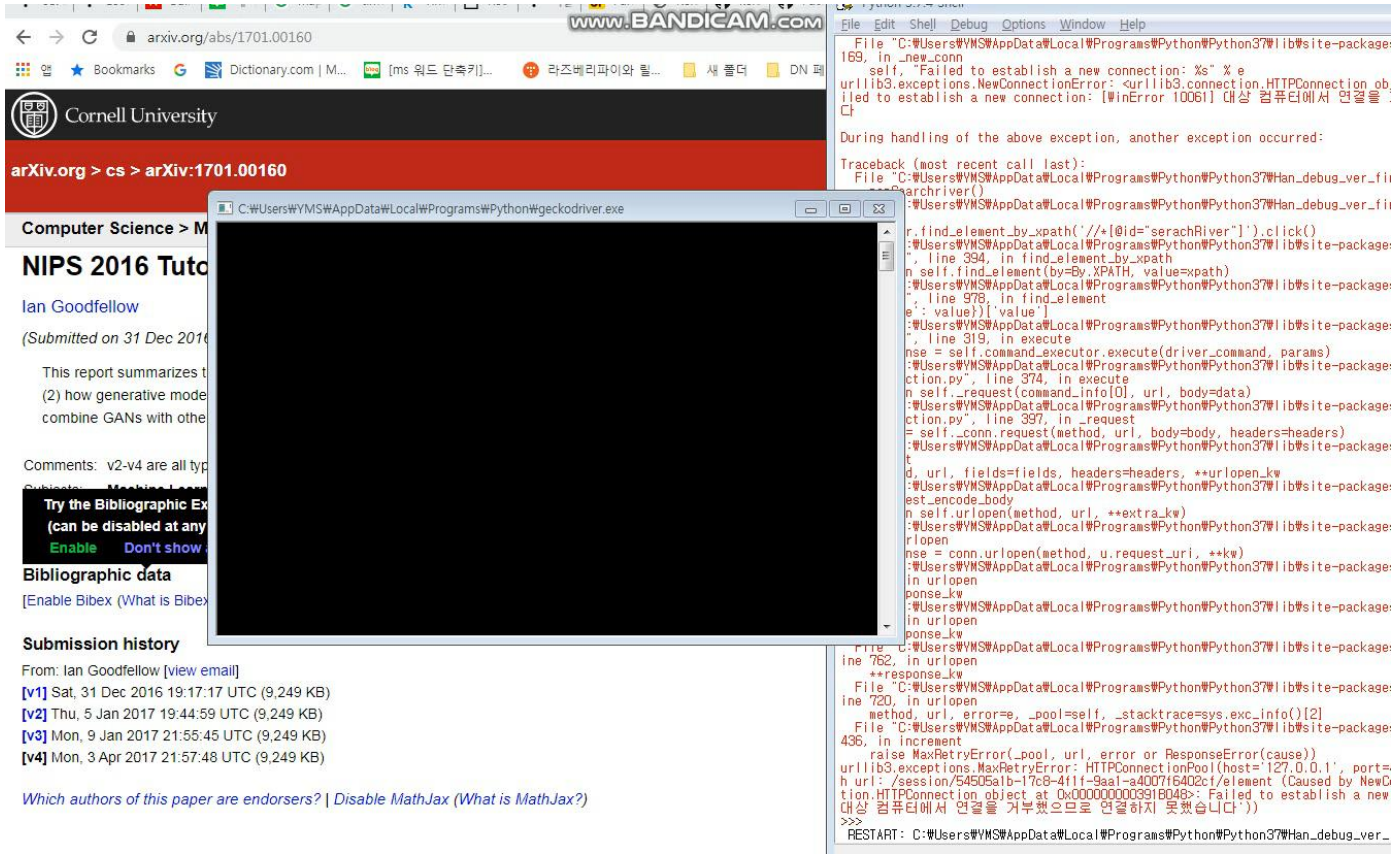
    b) 외래어종 : 떡붕어, 무지개송어, 향어 (이스라엘 잉어)

    c) 토속종

  - 수질환경측정망

    : pH, DO, BOD, COD, SS, TN, TP, TOC, ℃ .. etc

2. Data crawling with Selenium



2. Data crawling with Selenium


```
# start~
url = 'http://water.nier.go.kr/waterData/bioSearch.do?menuIdx=3_1_12&siteTypeCd=A'
driver.get(url)
driver.implicitly_wait(2)
set_startYear(2014)
set_endYear(2018)
set_quarter(2)
selectBio()
time.sleep(1)
openPopup()
time.sleep(8)
popSearchriver()
time.sleep(3)
selectHan()
time.sleep(3)
```

```
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
information = soup.find_all('tbody',{'id':'txtGrid'})[0].find_all('input')


length = len(information)
onetime = 15
compart = math.ceil(length/onetime)

for count in range(0,(compart-1)):
    temp = information[onetime*count:onetime*(count+1)]
    for i,info in enumerate(temp):
        id_code = str(info['id'])
        driver.find_element_by_id(id_code).click()
    popscreenOut()
    time.sleep(2)
    Search_button()
    time.sleep(5)
    downloadAs('xls')
    time.sleep(3)
    print(count)
```

2. Data matching : GIS