2017 2학기
인공지능 강의노트

13주차

강사: 양일호

School of Computer Science
University of Seoul

강의 계획

주차	강의	실습
1	과목 소개 인공지능 및 python 소개 Python 프로그래밍 환경 조성 및 기본 문법	Python 개발 환경 구축 간단한 expert system 구현
2	Search (Blind Search, Heuristic Search)	DFS, BFS, A* 구현
3	Learning / Simulated Annealing	Simulated Annealing 구현
4	Genetic Algorithm	Genetic Algorithm 구현
5	보강 주간 (개천절 / 추석 연휴)	
6	Neural Network	Perceptron 구현
7	Neural Network	(취소됨)
8	Neural Network	Single-Layer Perceptron 구현
9	Neural Network	Multi-Layer Perceptron 구현
10	Deep Neural Network	DNN 구현
11	Convolutional Neural Network	(취소됨)
12	Dropout / BatchNorm / ResNet	Keras 기초, CNN
13	DNN 심화 (혹은 Gaussian Mixture Model)	관련 내용 구현
14	DNN 심화 (혹은 Bayesian Networks)	관련 내용 구현
15	DNN 심화 (혹은 Decision Networks)	관련 내용 구현
16	기말고사	

Python 숙달을 위한 준비 기간

예비 기간 (대체 가능)

Classification / regression model

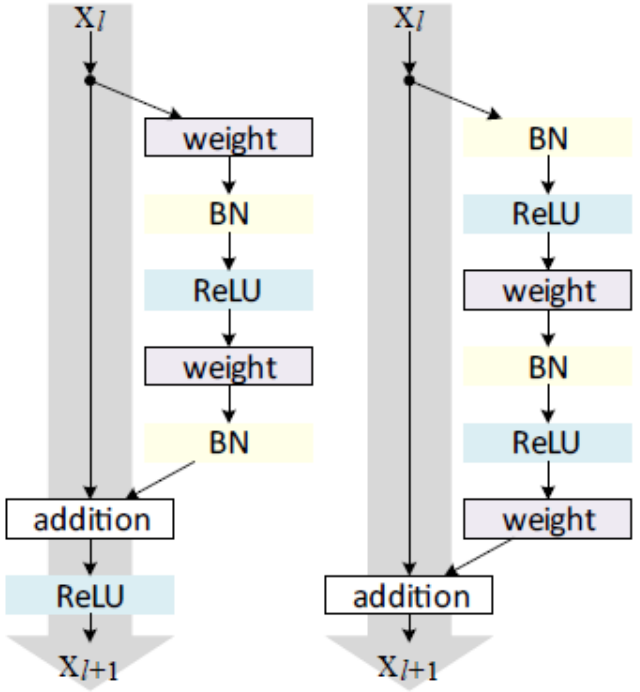
- Classification model
  - 신경망 학습시 target output이 class label (discrete value)
    - Ex〉 one-hot representation
    - Activation value가 가장 큰 output unit의 index (class label)가 중요

- Regression model
  - 신경망 학습시 target output이 continuous value
    - Output unit의 activation value 그 자체가 중요

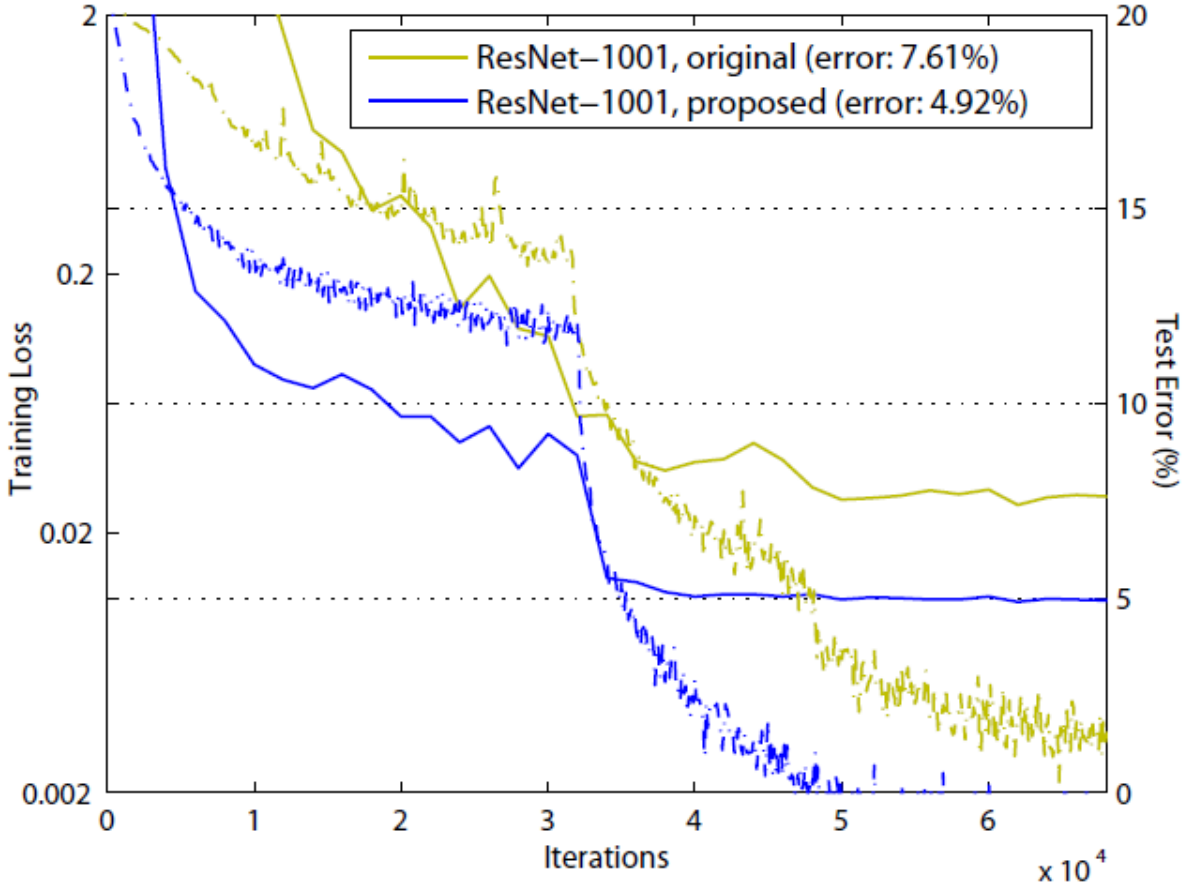
Generalization

- 학습 샘플에 과적합(overfitting)되는 경우
  - 일반화(generalization) 성능 하락
    - 너무 복잡한 모델(capacity가 큰 신경망)로 너무 지나치게 오래 학습 수행
  - 신경망에서의 회피 방법 예시
    1. 단순한 모델(capacity가 작은 신경망) 사용
       - 주어진 문제에 최적인 capacity를 어떻게 정할 것인가?
    2. 학습 오류율이 완전히 수렴하기 전에 일찍 학습을 중단(early stopping)
       - Ex〉 validation set에 대한 오류가 충분히 수렴했을 때 중단
    3. Regularization
    4. Dropout
    5. Batch normalization

ResNet(v2)



(a) original (b) proposed



CIFAR-10 학습(점선)/테스트(실선) 오류율

Q〉 v1과 v2의 차이는 무엇일까?

강의 계획

주차	강의	실습
1	과목 소개 인공지능 및 python 소개 Python 프로그래밍 환경 조성 및 기본 문법	Python 개발 환경 구축 간단한 expert system 구현
2	Search (Blind Search, Heuristic Search)	DFS, BFS, A* 구현
3	Learning / Simulated Annealing	Simulated Annealing 구현
4	Genetic Algorithm	Genetic Algorithm 구현
5	보강 주간 (개천절 / 추석 연휴)	
6	Neural Network	Perceptron 구현
7	Neural Network	(취소됨)
8	Neural Network	Single-Layer Perceptron 구현
9	Neural Network	Multi-Layer Perceptron 구현
10	Deep Neural Network	DNN 구현
11	Convolutional Neural Network	(취소됨)
12	Dropout / BatchNorm / ResNet	Keras 기초, CNN
13	데이터의 가공 및 시각화	관련 내용 구현
14	DNN 심화 (혹은 Bayesian Networks)	관련 내용 구현
15	DNN 심화 (혹은 Decision Networks)	관련 내용 구현
16	기말고사	

Python 숙달을 위한 준비 기간

예비 기간 (대체 가능)

강의개요

- 강의
  - 데이터의 가공 / 시각화
    - Preprocessing
    - Dimensionality reduction
    - Data visualization
  - Clustering

- 실습

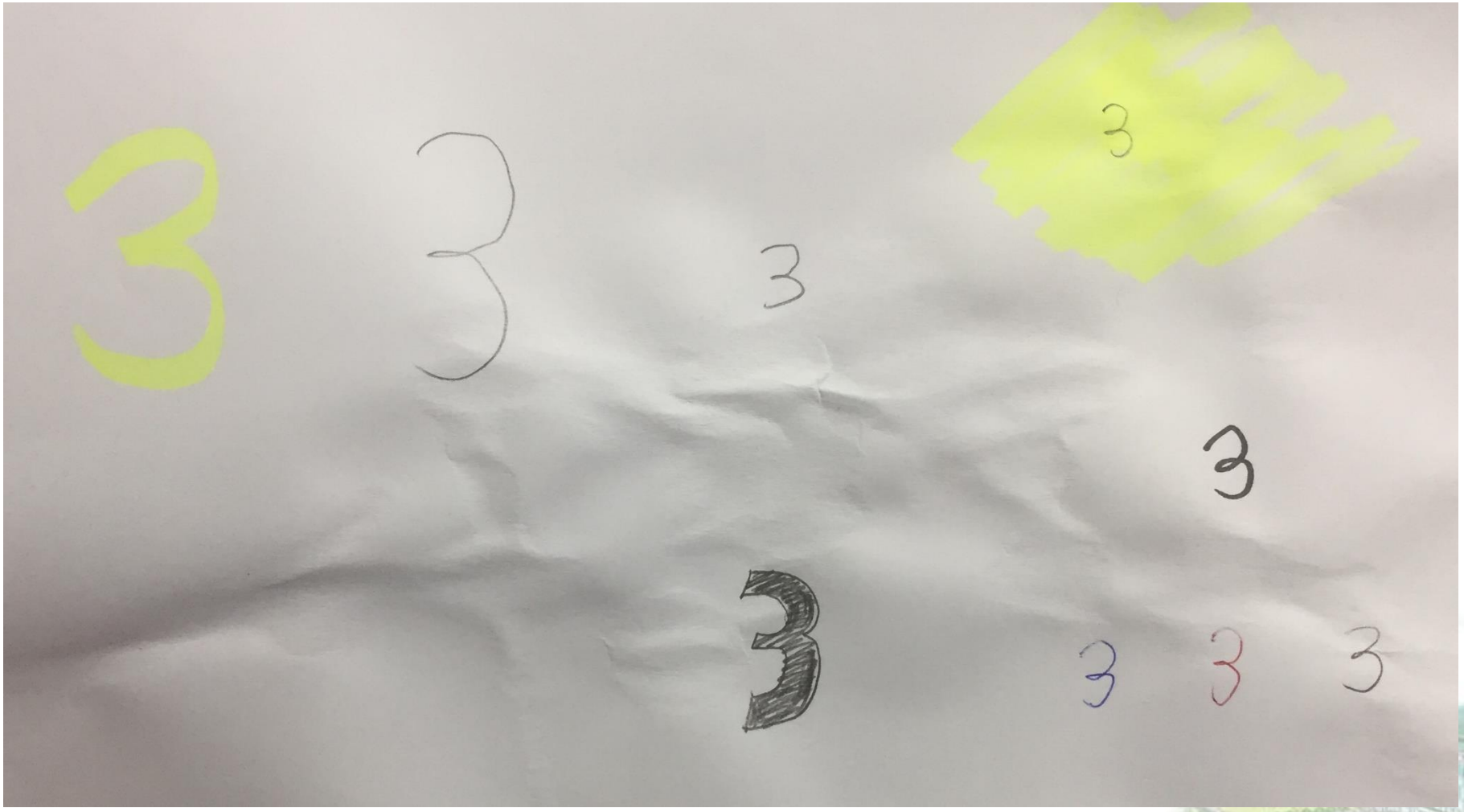
강의

School of Computer Science
University of Seoul

데이터의 가공/시각화

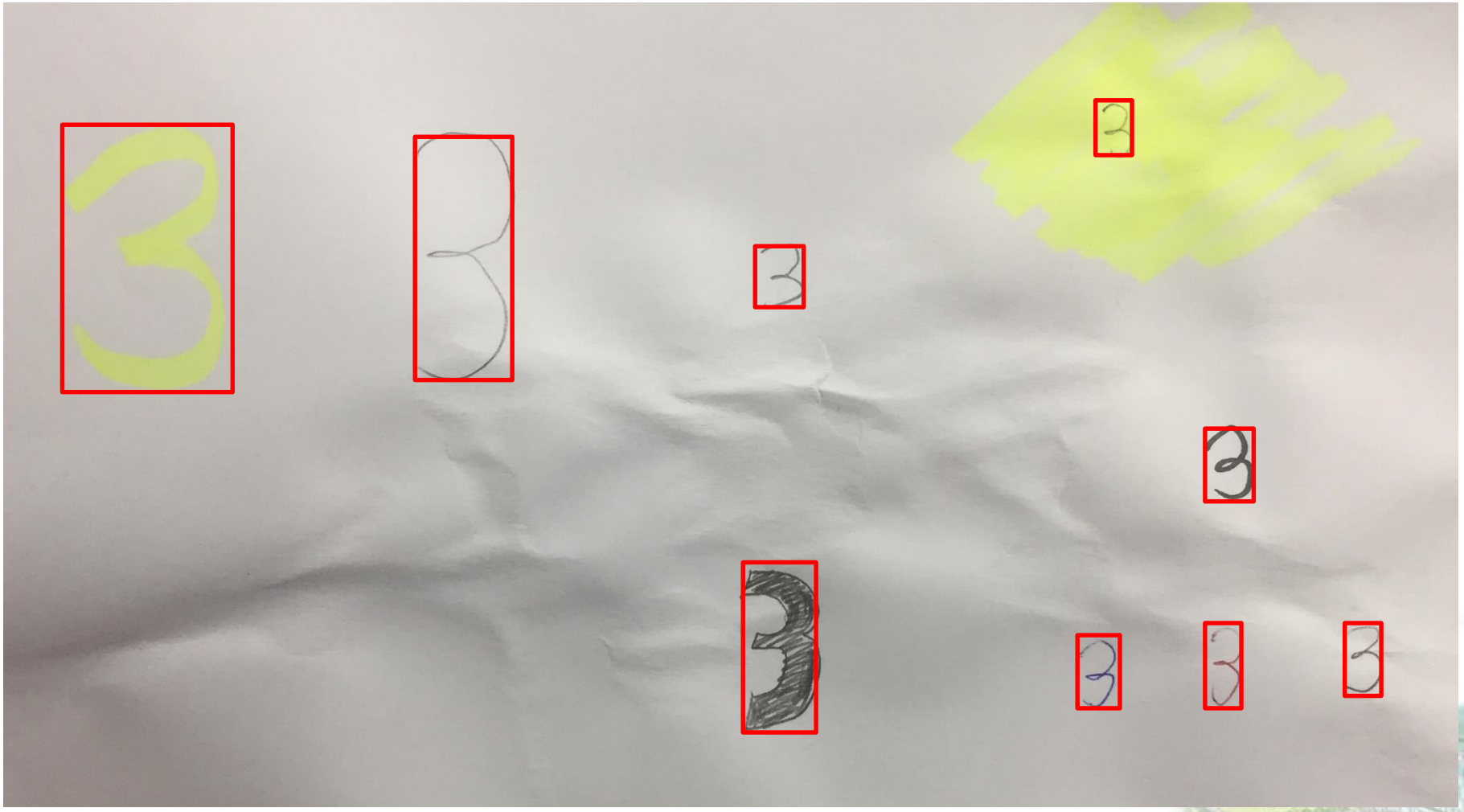
Real data

- 다음 중 “3”을 모두 찾으시오



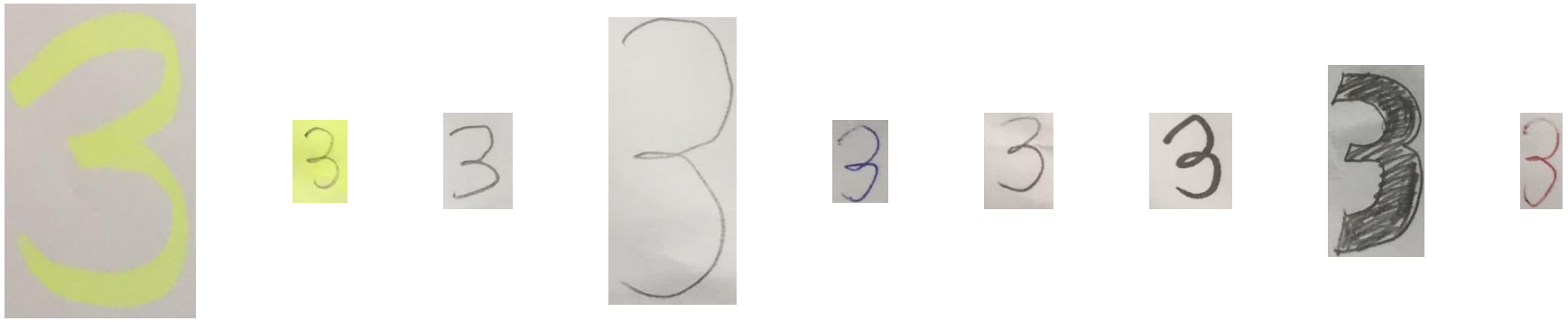
Real data

- 다음 중 “3”을 모두 찾으시오



Real data

- 같지만 다른 데이터들…
  - 다음 이미지들의 공통점은 무엇인가?
  - 다음 이미지들의 차이점은 무엇인가?



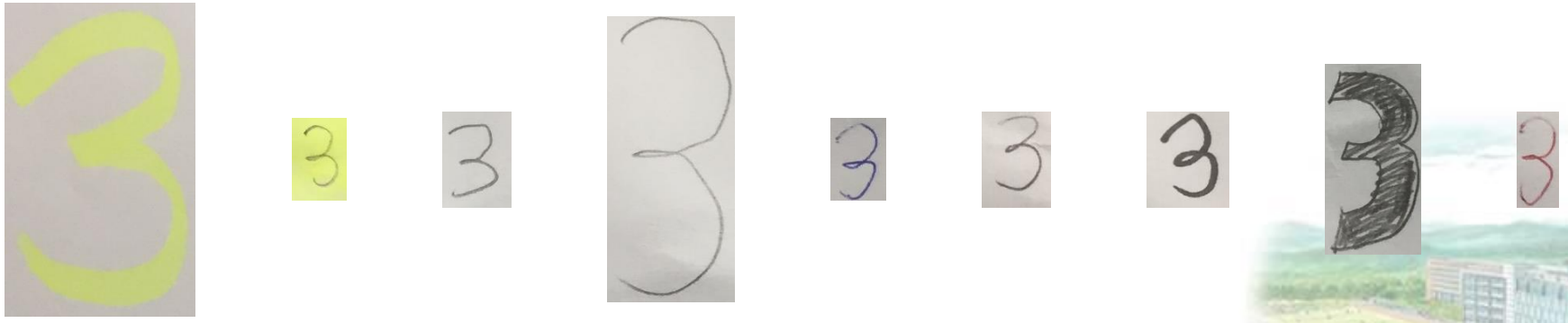
Q〉 위와 같은 데이터를 그대로 입력하면 숫자 인식을 잘 수행할 수 있을까?

Real data

- 다음 이미지들의 공통점은 무엇인가?
  - 모두 동일한 클래스의 데이터 샘플들: “3”을 쓴 이미지

- 다음 이미지들의 차이점은 무엇인가?
  - 이미지의 크기
  - 배경의 밝기/음영
  - 선의 굵기
  - 선의 색깔
  - …

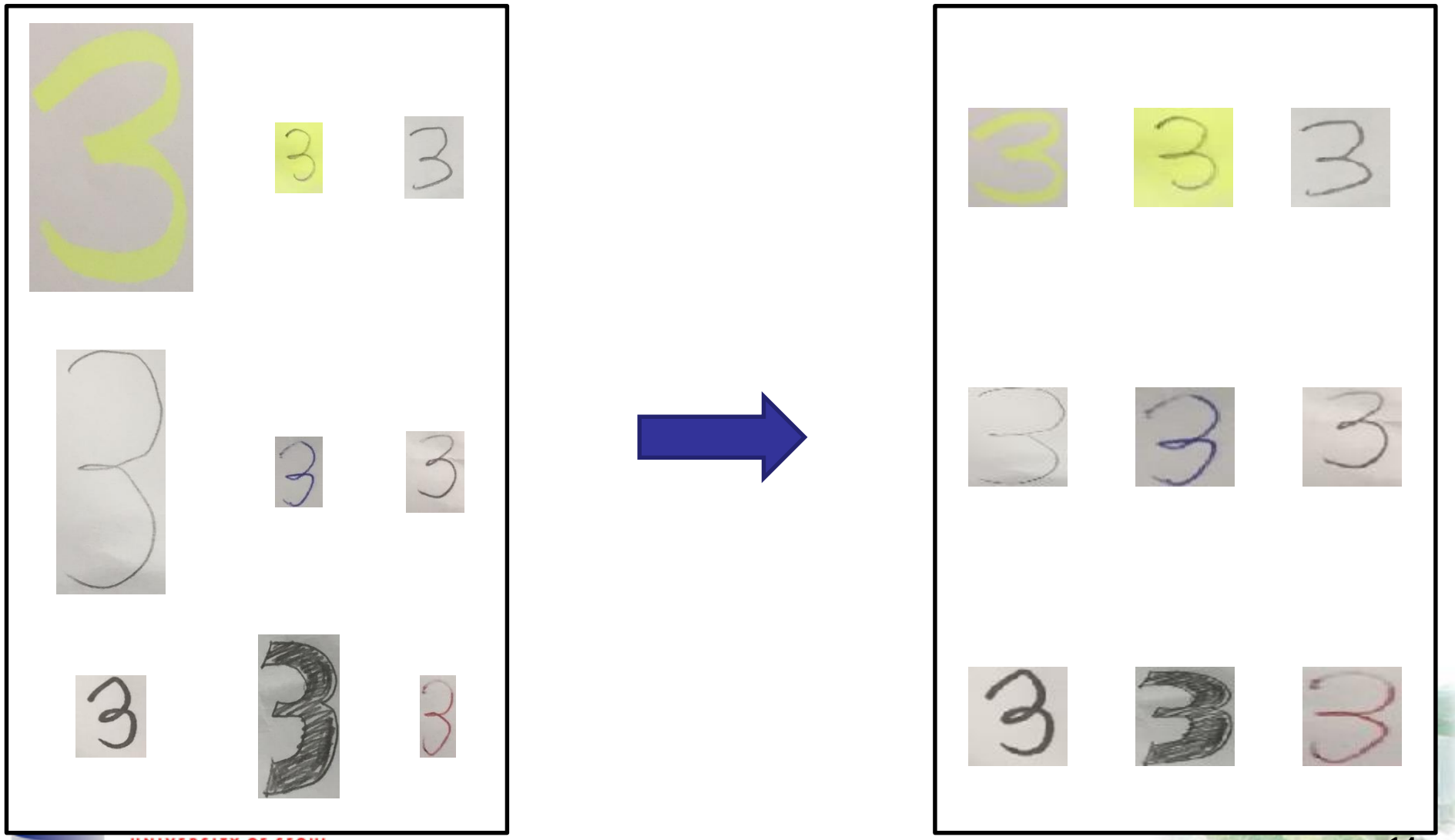
패턴 인식(숫자 인식)에 방해가 되는 요인들



Q〉 위와 같은 데이터로 숫자 인식을 잘 수행하려면 어떻게 해야 할까?

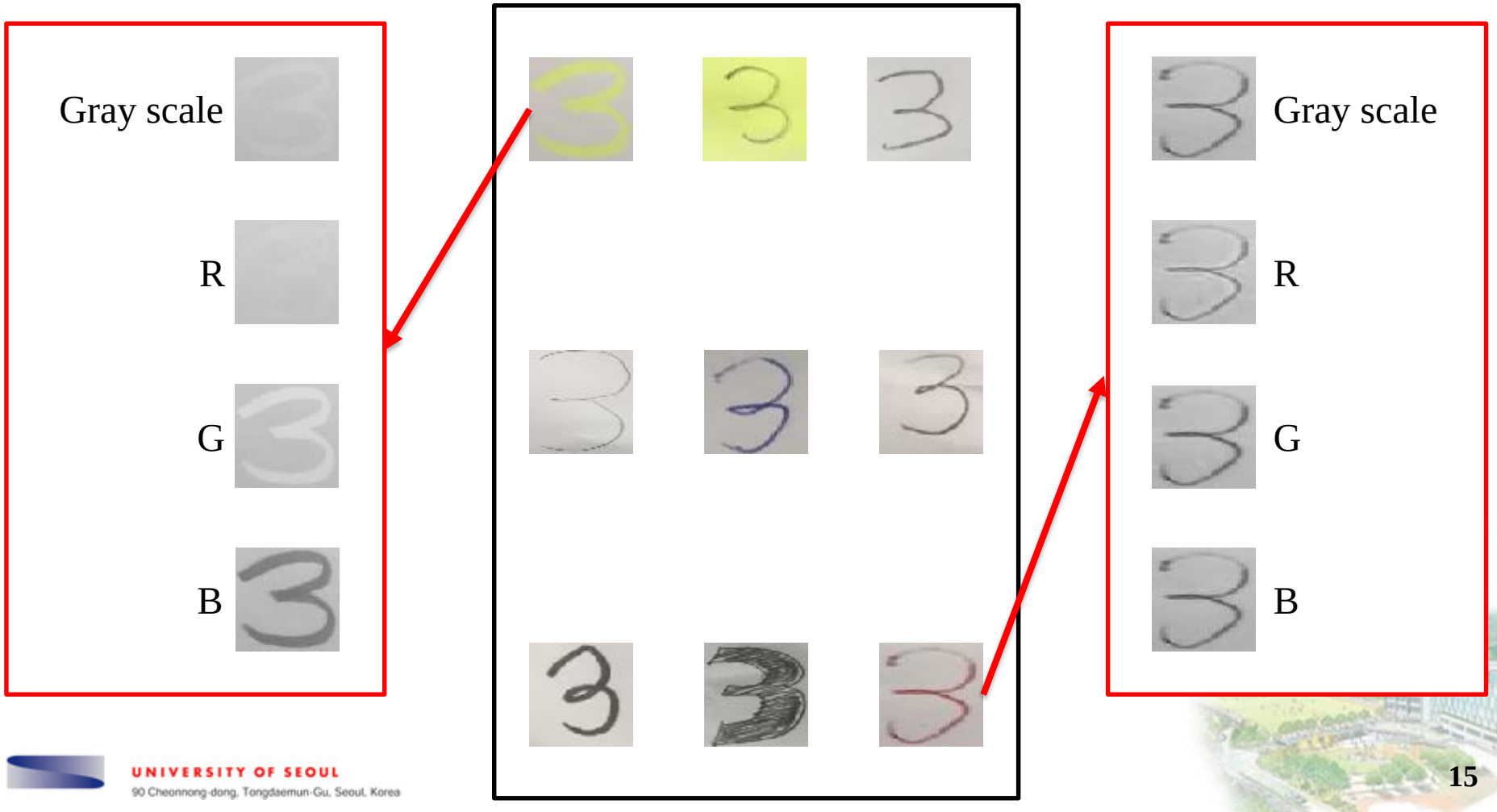
전처리(pre-processing) 과정 예시

- 이미지 크기를 일정하게 scaling



전처리(pre-processing) 과정 예시

- 뭘 보고 분류하지?



전처리(pre-processing) 과정 예시

- 잘 안 보이는데… 픽셀 밝기를 0~255로 조정해보자!
  - 정규화(normalization)

Gray image에 대한 normalization:

$$I_N = (I - \text{Min})\frac{\text{newMax} - \text{newMin}}{\text{Max} - \text{Min}} + \text{newMin}$$

I : 원본 이미지의 한 pixel 값

Min: 원본 이미지 전체 pixel 값들의 최소치

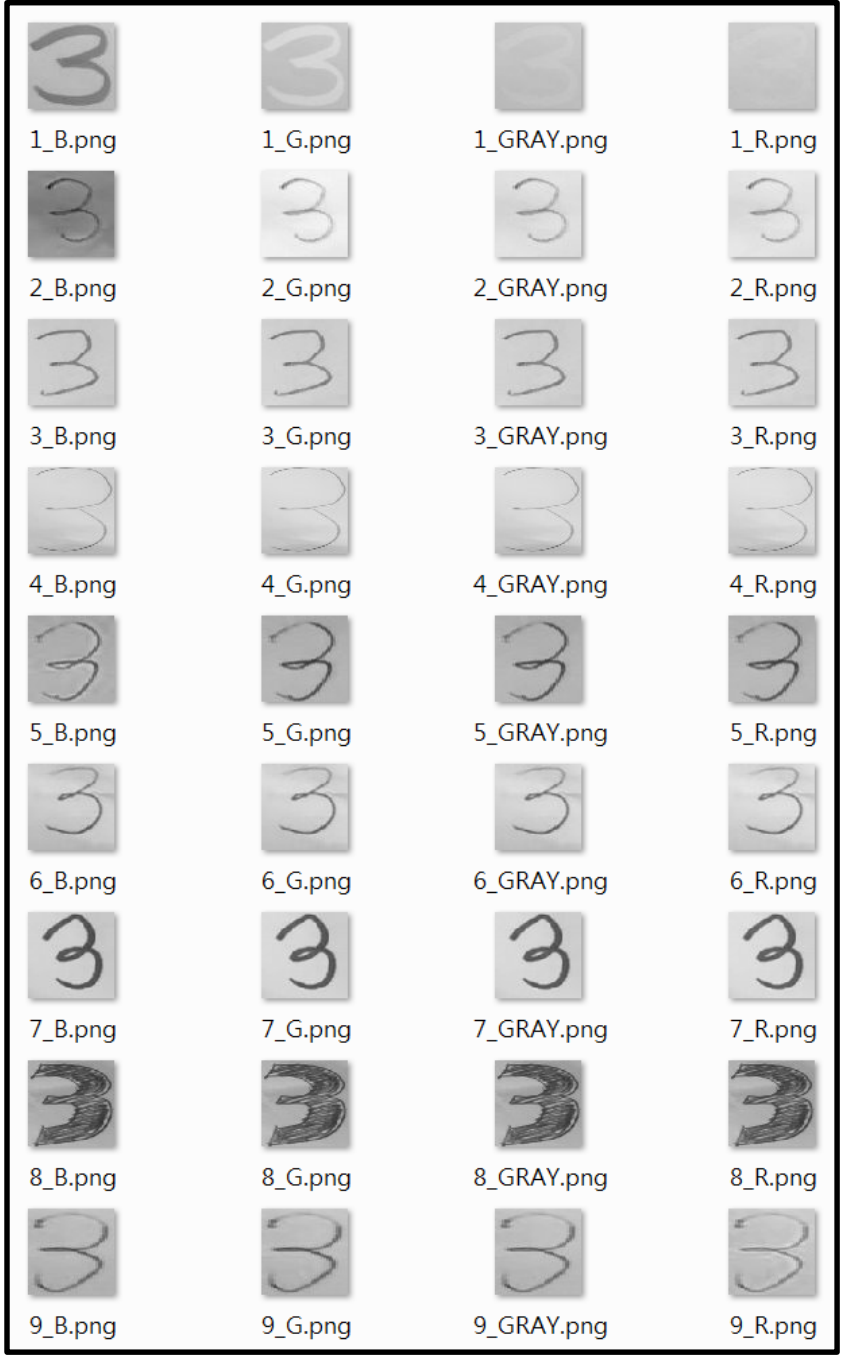
Max: 원본 이미지 전체 pixel 값들의 최대치

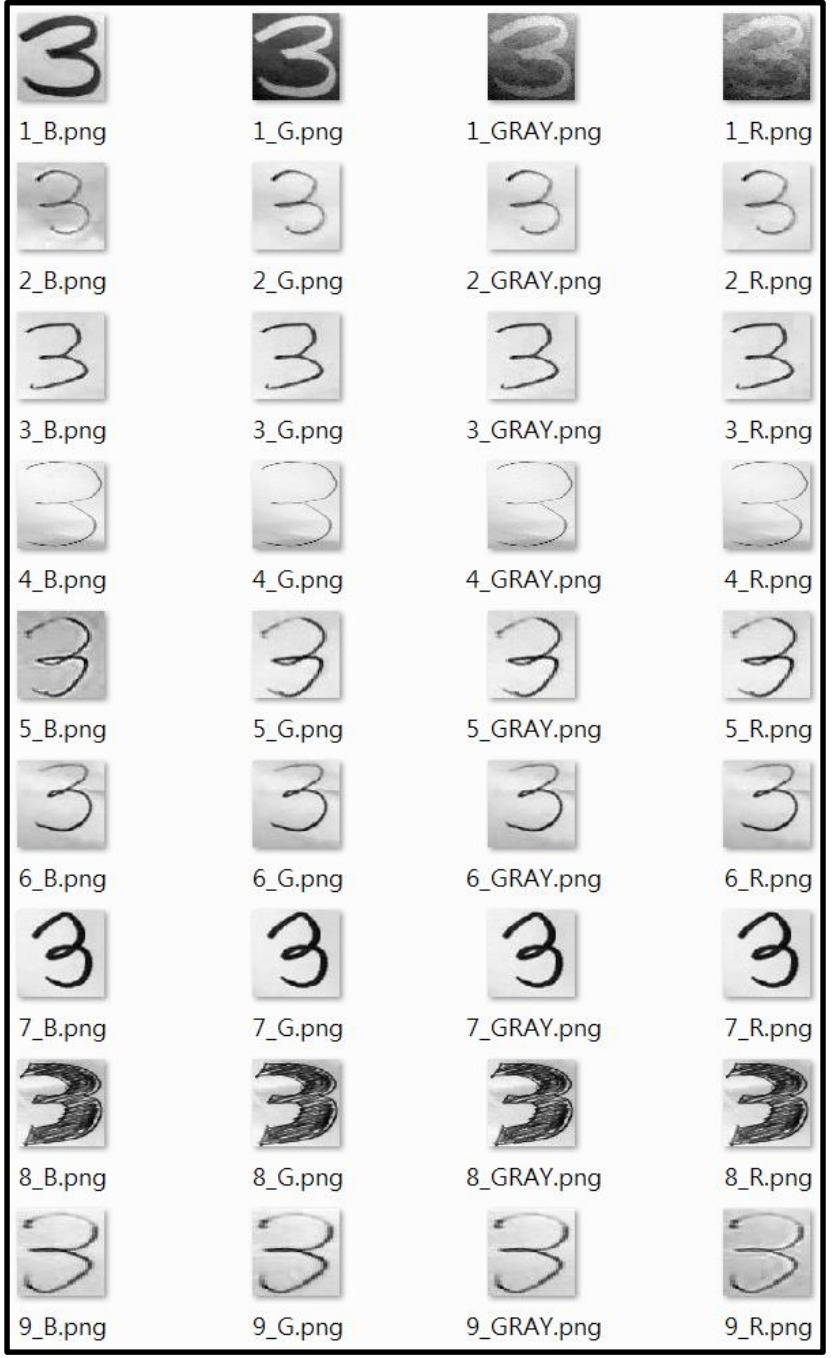
I_N : 정규화한 이미지의 한 pixel 값

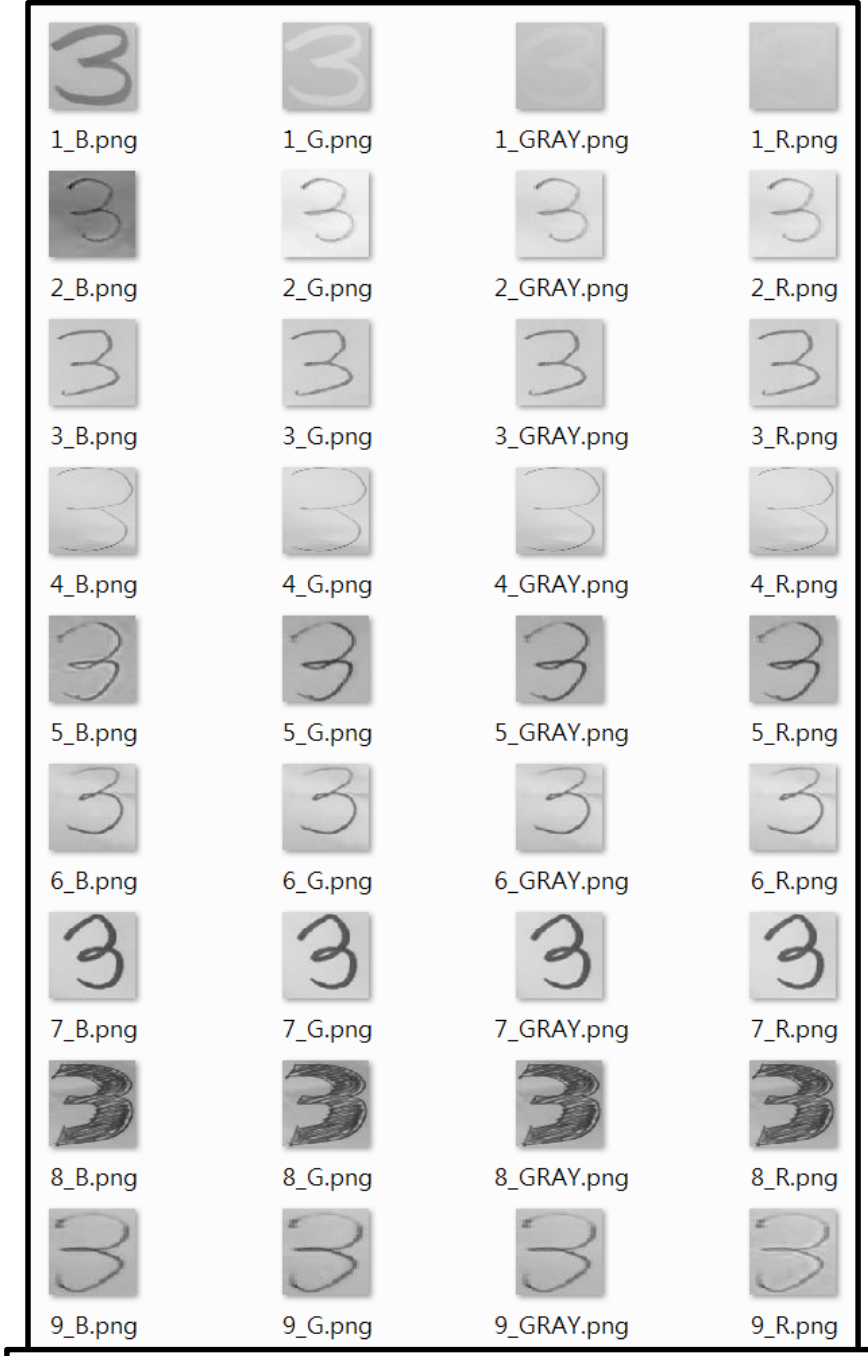
newMin: 정규화한 이미지 전체 pixel 값들의 최소치

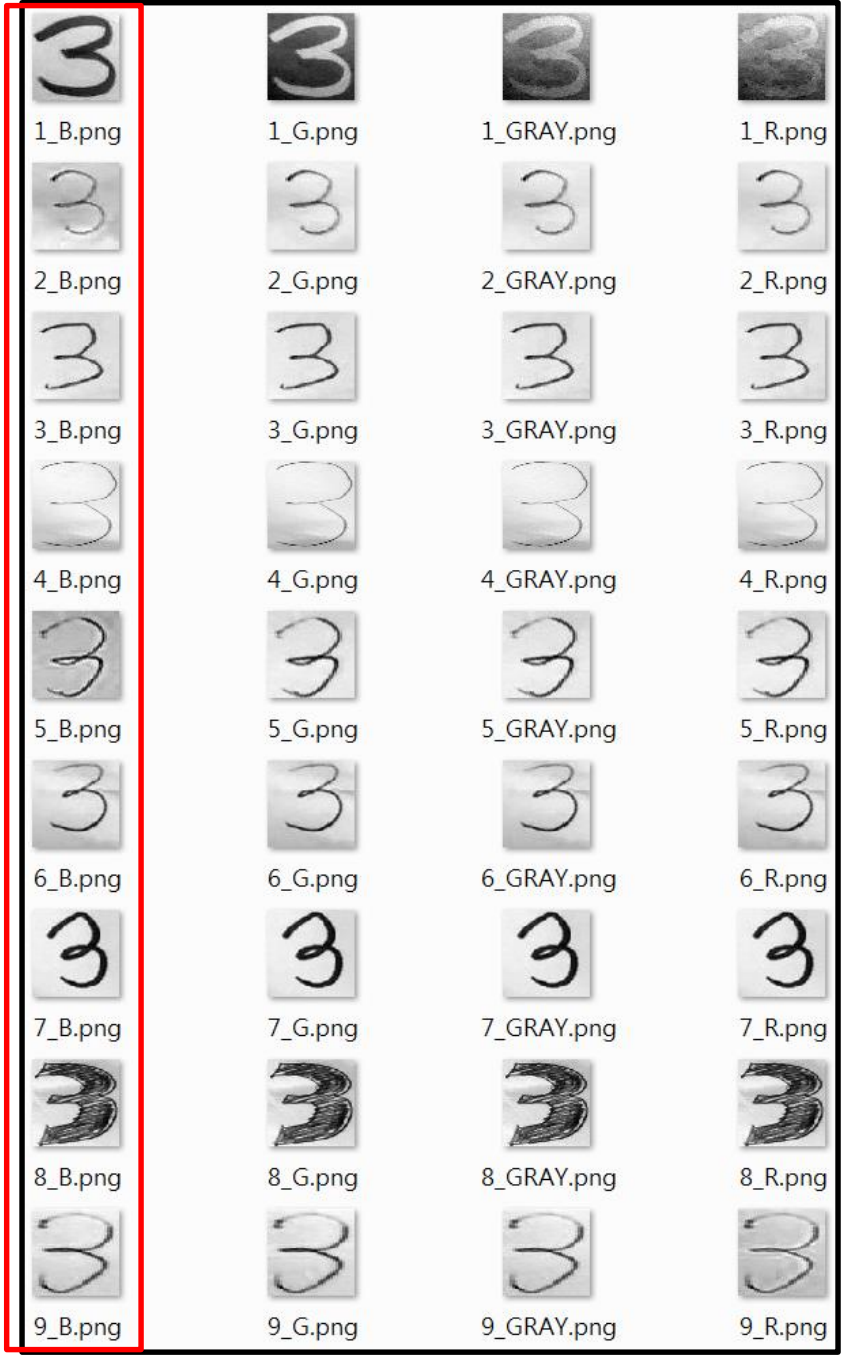
newMax: 정규화한 이미지 전체 pixel 값들의 최대치

https://en.wikipedia.org/wiki/Normalization_(image_processing)


1_B.png
1_G.png
1_GRAY.png
1_R.png
2_B.png
2_G.png
2_GRAY.png
2_R.png
3_B.png
3_G.png
3_GRAY.png
3_R.png
4_B.png
4_G.png
4_GRAY.png
4_R.png
5_B.png
5_G.png
5_GRAY.png
5_R.png
6_B.png
6_G.png
6_GRAY.png
6_R.png
7_B.png
7_G.png
7_GRAY.png
7_R.png
8_B.png
8_G.png
8_GRAY.png
8_R.png
9_B.png
9_G.png
9_GRAY.png
9_R.png


1_B.png
1_G.png
1_GRAY.png
1_R.png
2_B.png
2_G.png
2_GRAY.png
2_R.png
3_B.png
3_G.png
3_GRAY.png
3_R.png
4_B.png
4_G.png
4_GRAY.png
4_R.png
5_B.png
5_G.png
5_GRAY.png
5_R.png
6_B.png
6_G.png
6_GRAY.png
6_R.png
7_B.png
7_G.png
7_GRAY.png
7_R.png
8_B.png
8_G.png
8_GRAY.png
8_R.png
9_B.png
9_G.png
9_GRAY.png
9_R.png



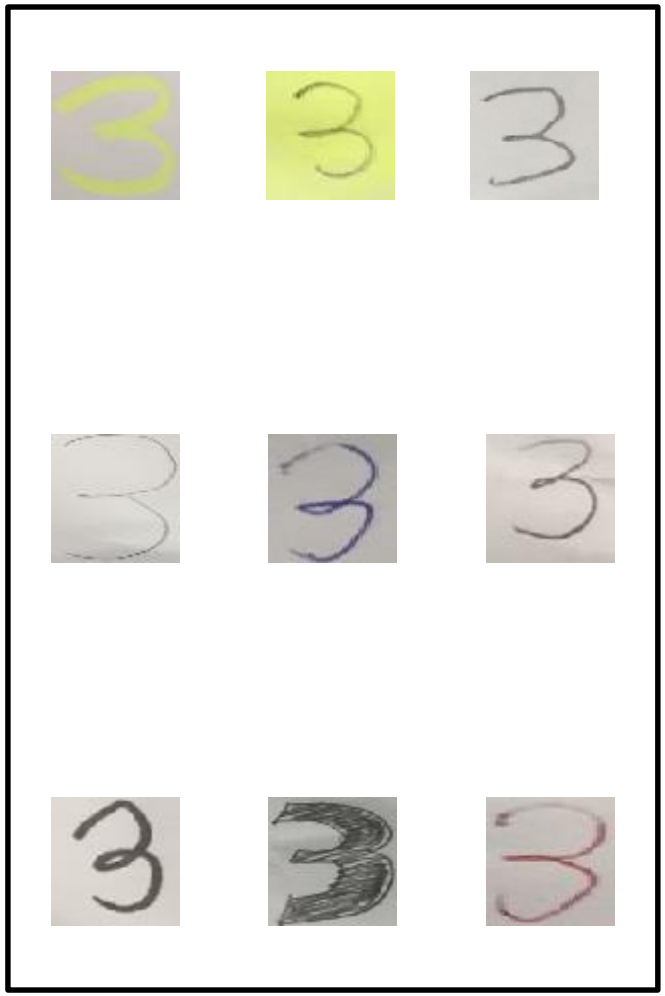


정규화한 B채널 이미지를 특징으로 써 볼까?

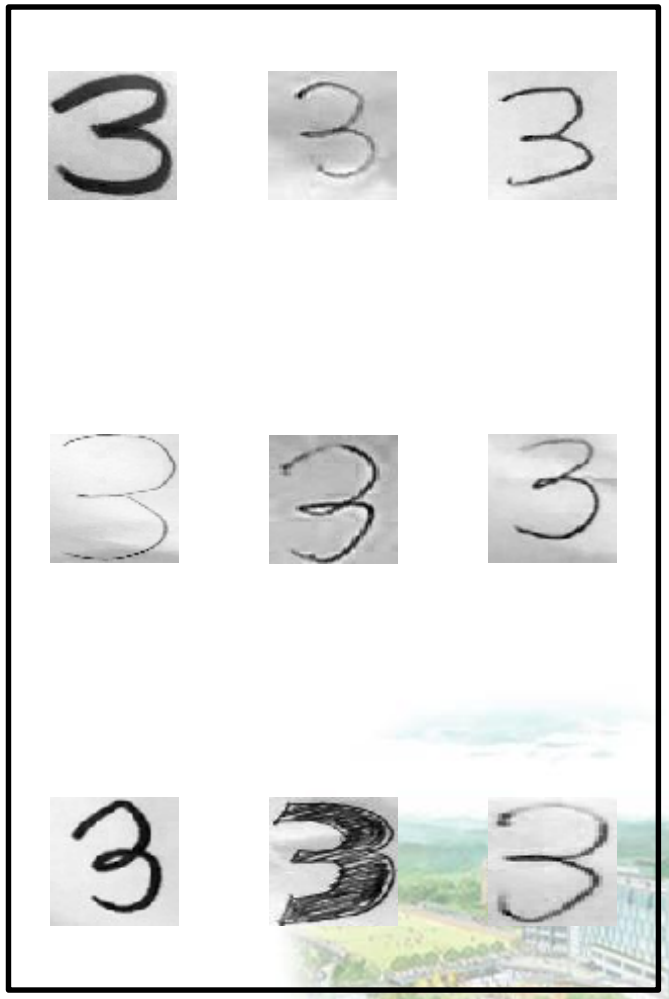
→ (manual) feature selection

전처리(pre-processing) 과정 예시

- 여기서 좀 더 분류하기 쉬운 특징으로 바꿀 수 있을까?

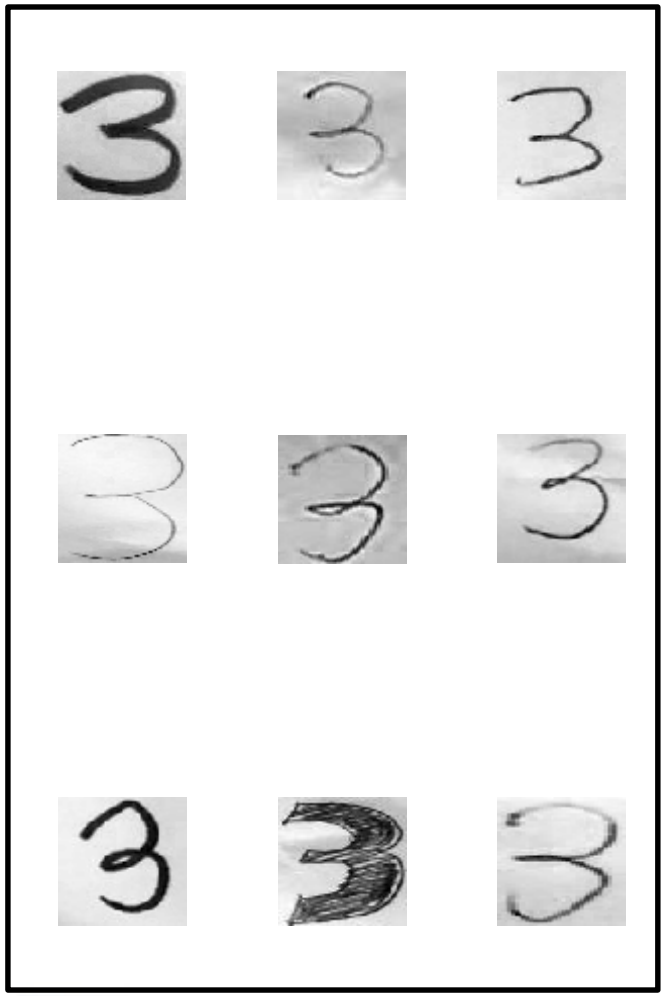


B채널 이미지
+Normalization



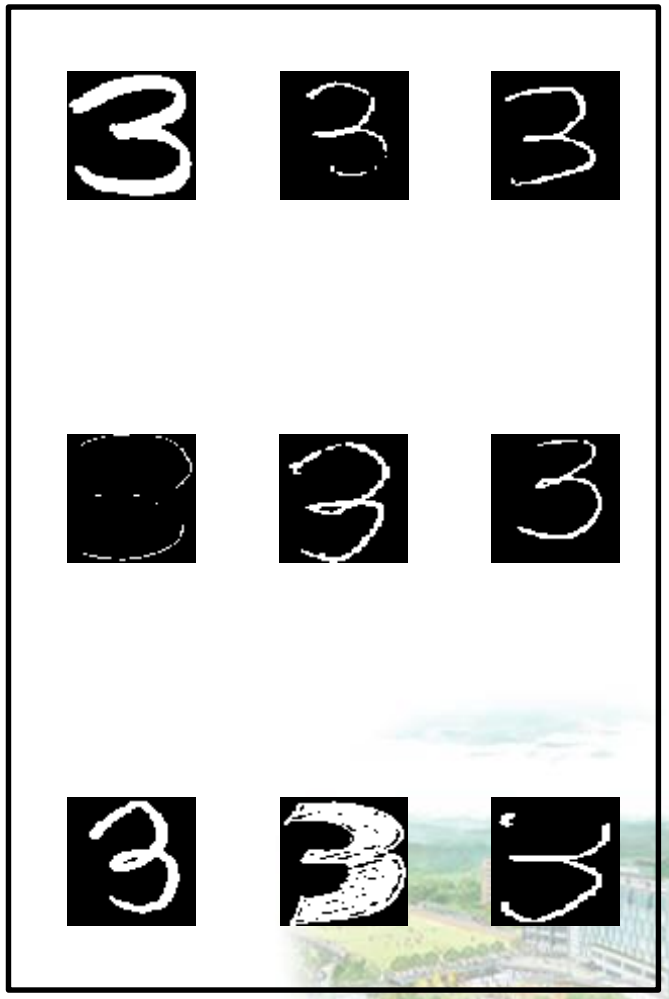
전처리(pre-processing) 과정 예시

- 여기서 좀 더 분류하기 쉬운 특징으로 바꿀 수 있을까?



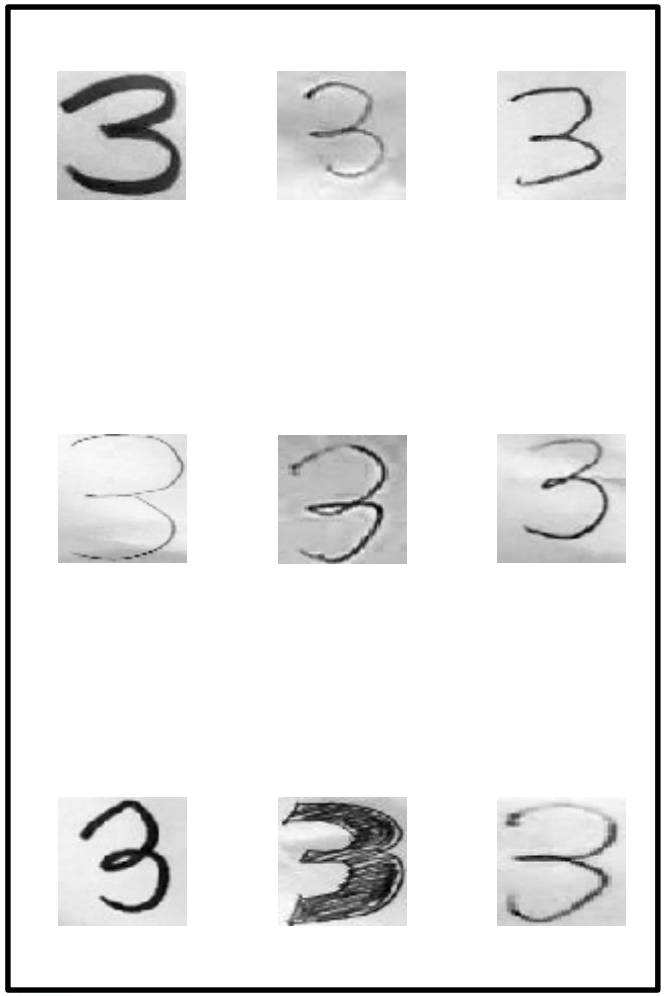
B채널 이미지
+Normalization

Pixel 값이
150보다 작으면 0,
크면 255로 바꾸고 반전



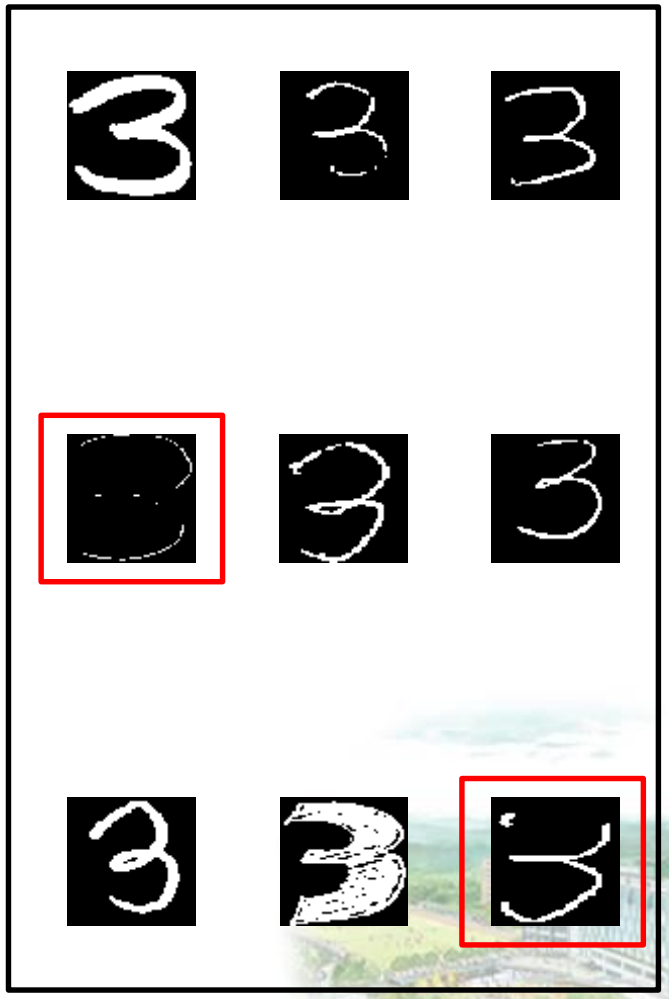
전처리(pre-processing) 과정 예시

- 여기서 좀 더 분류하기 쉬운 특징으로 바꿀 수 있을까?



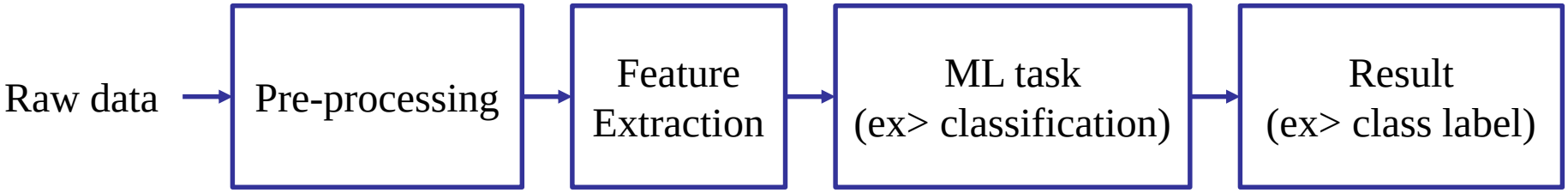
B채널 이미지
+Normalization

Pixel 값이
150보다 작으면 0,
크면 255로 바꾸고 반전

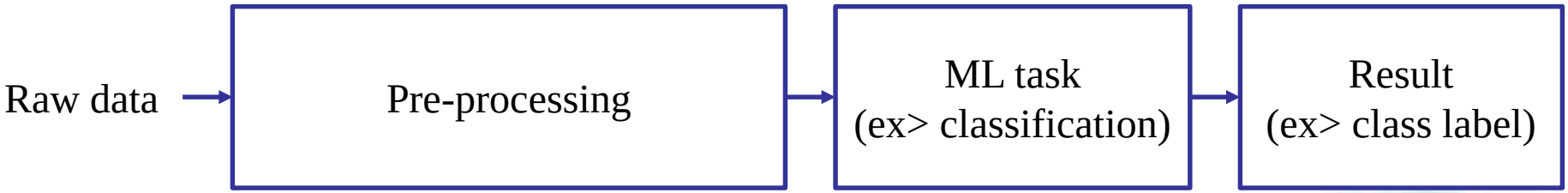


전처리(pre-processing)

- 좁은 의미에서의 전처리
  - 특징 추출 이전 단계에서 raw data를 정제하는 과정



- 넓은 의미에서의 전처리
  - ML task를 수행하기 위해 유의미한 정보(feature)를 추출하는 모든 과정



전처리(pre-processing)

- Data cleansing
  - 학습 데이터 샘플 중에서 이상한 데이터 걸러내기
- Instance selection
  - 학습 데이터 샘플이 너무 많을 때 쓸 만큼만 고르기
- Feature extraction
  - Raw data에서 좀 더 분류하기 편한 특징 추출
- Feature selection
  - 여러 특징 중에서 쓸만한 특징 몇 개만 사용
- Normalization
  - 데이터 샘플별/차원별 특징값이 일정한 규격을 갖도록 정규화하기
- Feature transform
  - 특징의 차원을 축소 / 새로운 특징을 추출
- …

전처리(pre-processing)

- Data cleansing
  - 학습 데이터 샘플 중에서 이상한 데이터 걸러내기
- Instance selection
  - 학습 데이터 샘플이 너무 많을 때 쓸 만큼만 고르기
- Feature extraction
  - Raw data에서 좀 더 분류하기 편한 특징 추출
- Feature selection
  - 여러 특징 중에서 쓸만한 특징 몇 개만 사용
- Normalization
  - 데이터 샘플별/차원별 특징값이 일정한 규격을 갖도록 정규화하기
- Feature transform
  - 특징의 차원을 축소 / 새로운 특징을 추출
- …

Q〉 정보가 많으면 많을수록 좋은 것 아닌가? 왜 특징(의 차원)을 줄여야 되지?

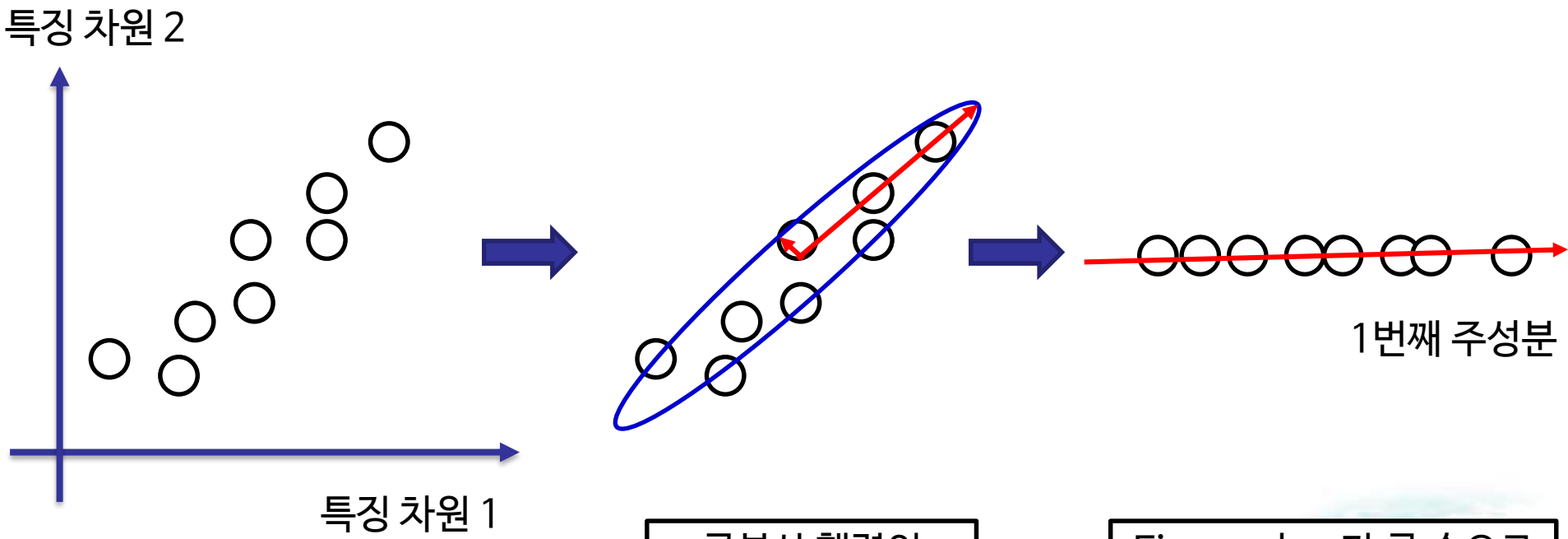
Curse of dimensionality

- 다양한 특징을 동시에 사용할 때의 장점
  - 다양한 정보를 활용 가능

- 다양한 특징을 동시에 사용할 때의 단점
  - 특징 벡터의 차원이 커짐
    - 연산량 증가
    - 특징의 sparsity가 증가
      - Ex〉 1000개의 학습 샘플로 다음 두 SLP를 학습할 때 어느 쪽이 더 유리할까?
        - ≫ CASE 1: 2차원 특징 추출 / 10개의 출력 unit
        - ≫ CASE 2: 100차원 특징 추출 / 10개의 출력 unit
        - ≫ (두 경우 모두 분류에 필요한 정보가 비슷한 수준으로 특징에 포함되었다고 간주)

Dimensionality reduction

- 데이터를 분류할 수 있는 정보를 최대한 유지하면서 차원을 줄이자
  - Ex〉 선형 변환(linear transform)을 이용한 feature transform
    - 주성분 분석(PCA, principal component analysis)

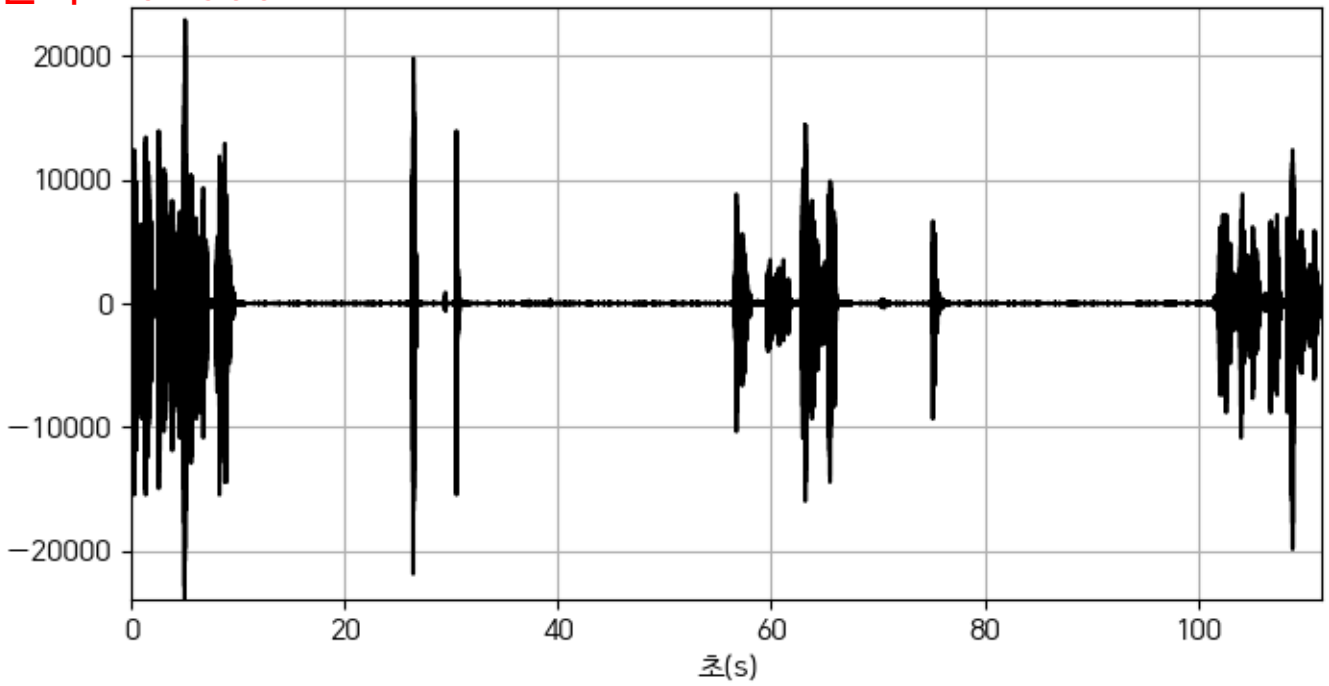


Data visualization

- 고차원 특징 벡터 → 1~3차원으로 축소하여 시각화
  - Ex〉 오디오 신호에서 추출한 고차원 특징을 2차원으로 축소하여 시각화

Sampling rate: 8000Hz

한 발성의 길이: 10~300초

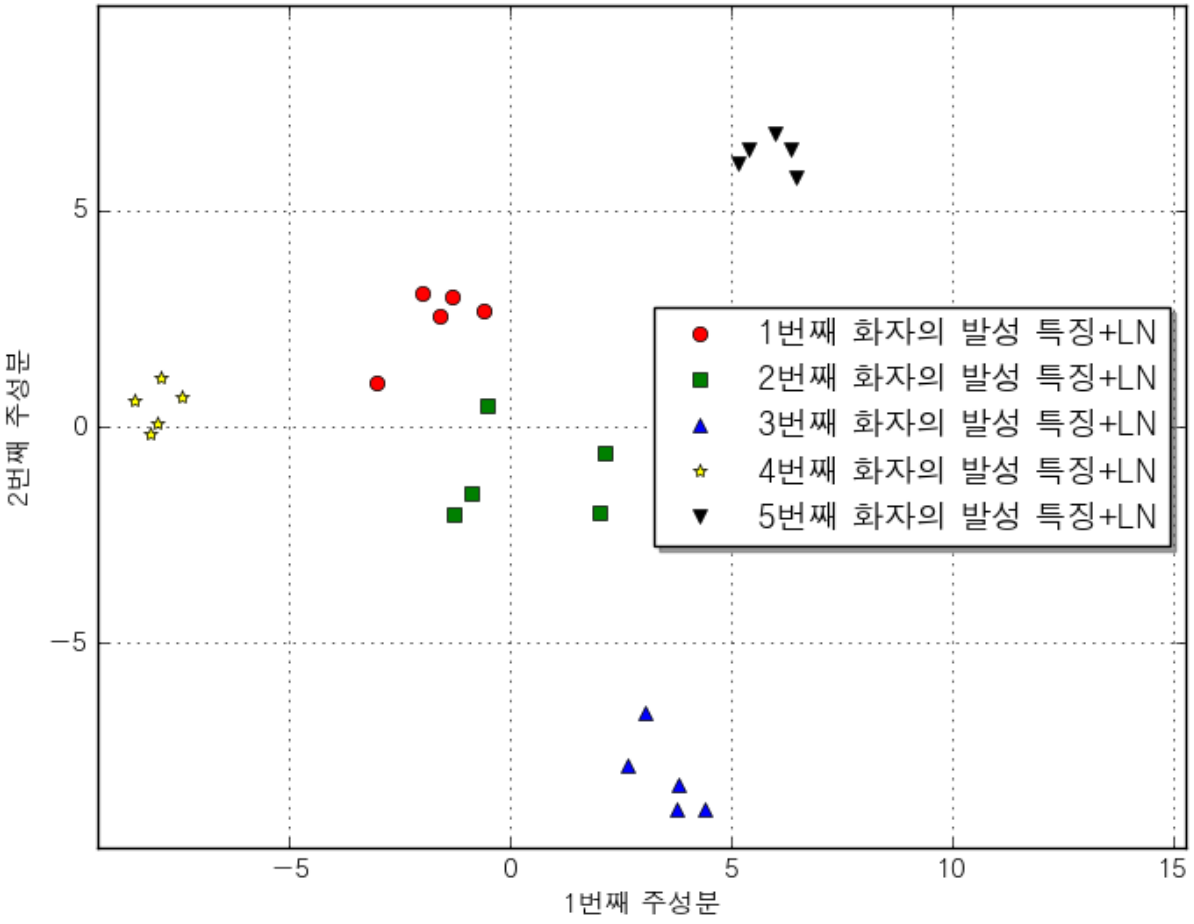


Feature extraction

150차원 특징

Data visualization

- 고차원 특징 벡터 → 1~3차원으로 축소하여 시각화
  - Ex〉 오디오 신호에서 추출한 고차원 특징을 2차원으로 축소하여 시각화
    - 25개 음성 파일에서 각각 추출한 150차원 특징을 PCA로 축소(150차원 → 2차원)



Handcrafted / learned feature

- Handcrafted feature
  - 사람이 고민한 방법으로 추출한 특징

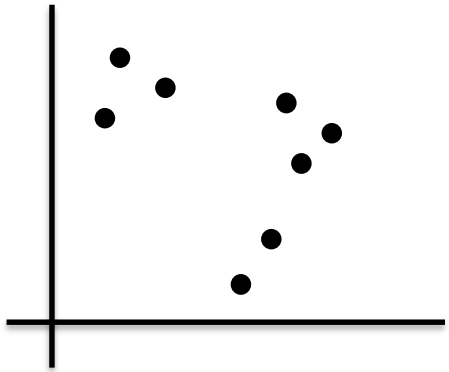
- Learned feature
  - Machine learning을 통해 추출한 특징
    - Ex〉 DNN에서 입력층에 가까운 layer들의 출력
      - 입력층에 가까울수록: low-level feature
      - 출력층에 가까울수록: high-level feature

군집화(clustering)

참고문헌: 오일석. (2008). 패턴인식.

clustering

- Label이 주어지지 않은 데이터를 비슷한 것들끼리 묶기

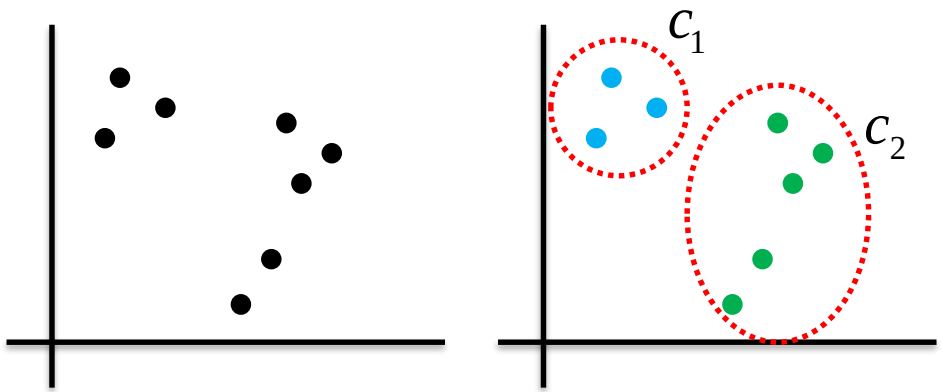


샘플 집합

부류(class) 정보가
주어지지 않음
(unsupervised learning)

clustering

- Label이 주어지지 않은 데이터를 비슷한 것들끼리 묶기

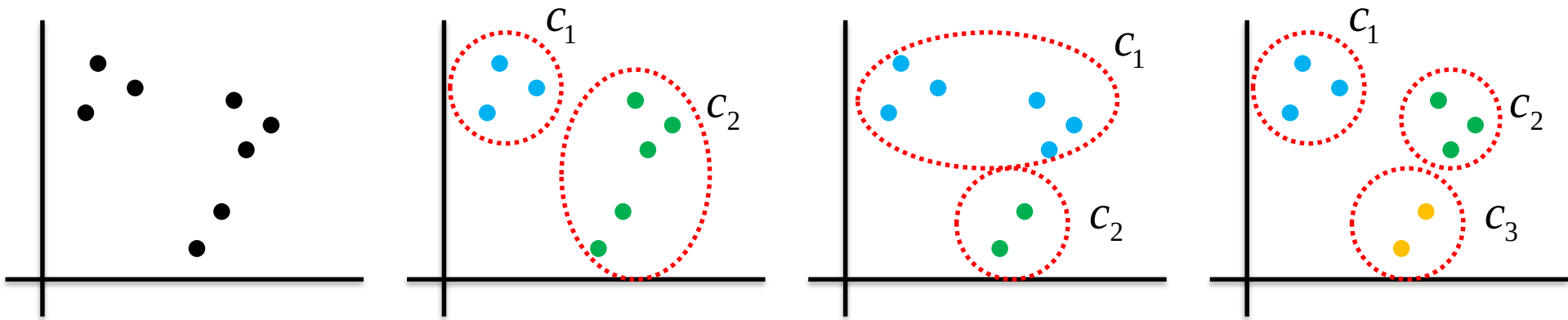


샘플 집합 군집 해1

부류(class) 정보가
주어지지 않음
(unsupervised learning)

clustering

- Label이 주어지지 않은 데이터를 비슷한 것들끼리 묶기

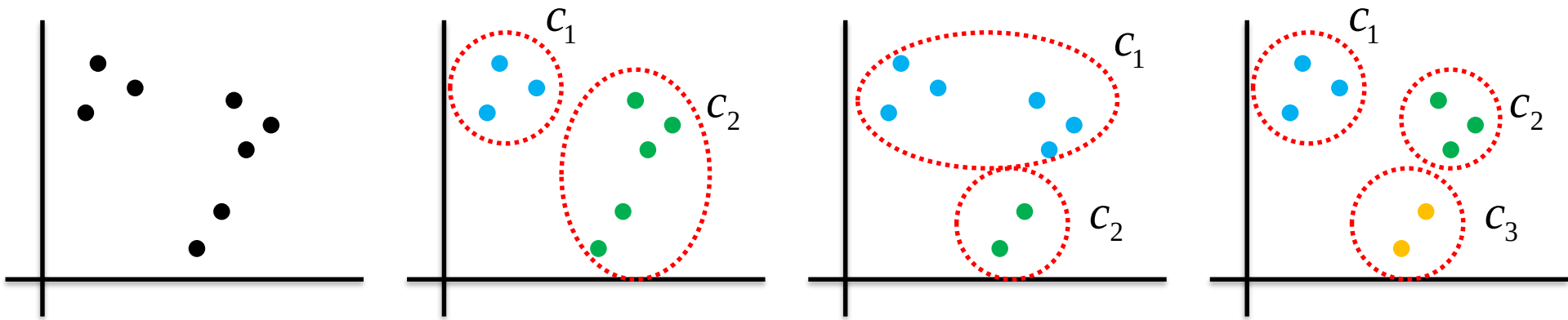


샘플 집합 군집 해1 군집 해2 군집 해3

부류(class) 정보가
주어지지 않음
(unsupervised learning)

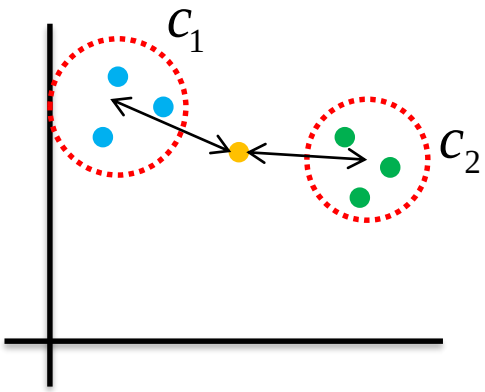
clustering

- Label이 주어지지 않은 데이터를 비슷한 것들끼리 묶기



clustering

- Hard clustering
  - 하나의 샘플은 하나의 군집에 소속됨



이런 경우는 어느 군집에 속해 있다고 봐야 되나?

clustering

- Soft clustering
  - 하나의 샘플이 여러 군집에 걸쳐 소속될 수 있음
    - 단, 모든 군집에 대한 소속변수의 합은 1

소속변수 m_{ij} : i-번째 샘플 $\mathbf{x}_i$가 j-번째 군집 c_j에 속하는 정도(비율)

$0 \le m_{ij} \le 1$ → 하나의 샘플은 어떤 군집에 완전히 소속되거나($m_{ij} = 1$)
소속되지 않거나($m_{ij} = 0$) 일부만 소속될 수 있음($0 \le m_{ij} \le 1$)

$\sum_{j=1}^{k} m_{ij} = 1$ → 한 샘플의 소속변수의 합은 1

$\sum_{i=1}^{N} m_{ij} < N$ → 하나의 군집에 모든 샘플이 완전히 소속되지는 않음 (총 N개의 샘플)

K-means

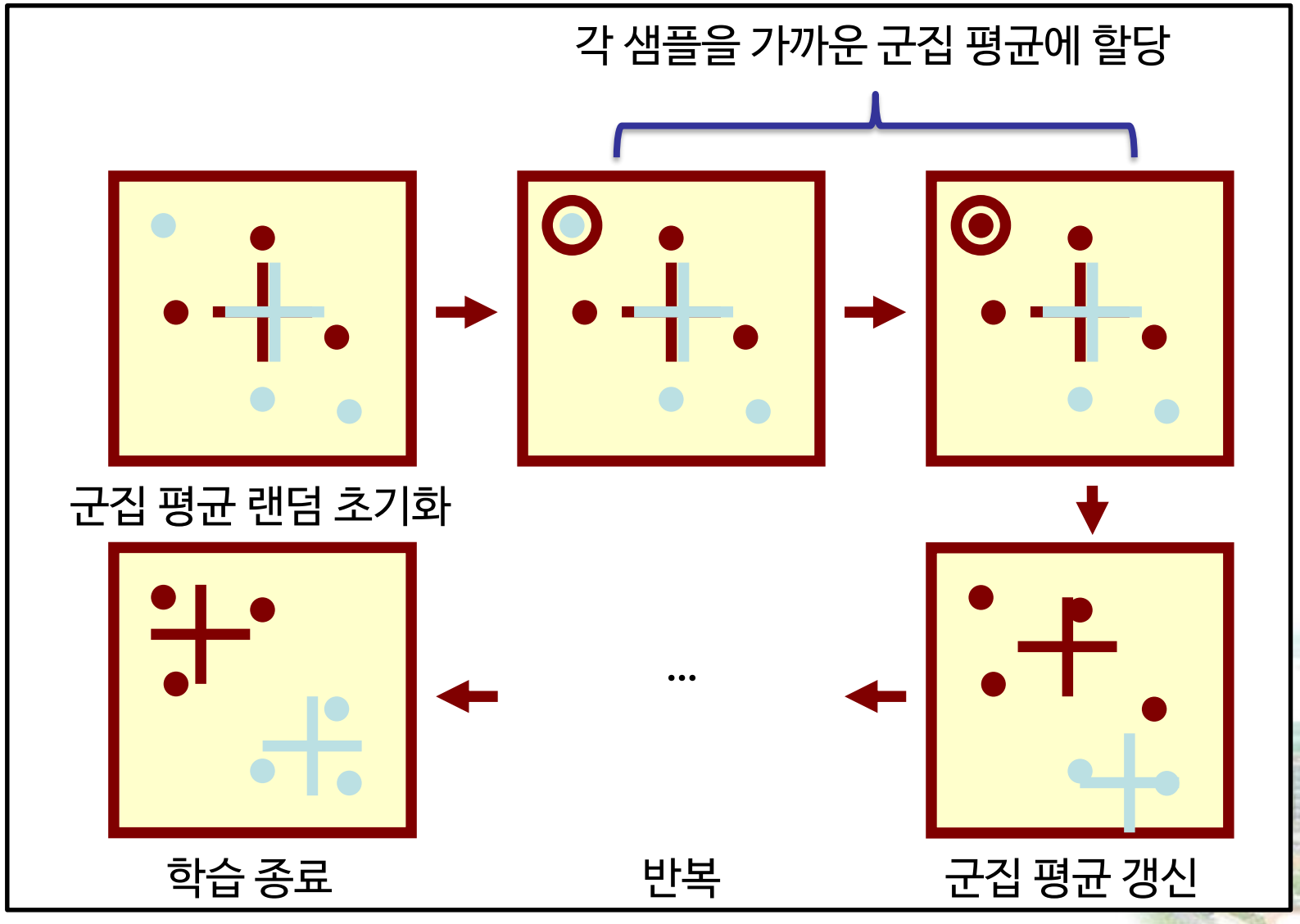
- Hard-clustering

- 전체 데이터 샘플을 K개의 군집으로 묶기
  - 몇 개의 군집으로 묶을지 정해 줌

- 수행 과정
  1. K개의 군집 평균을 랜덤 초기화
  2. 각 샘플을 평균이 가까운 군집에 할당
  3. 군집 평균 갱신
  4. 수렴할 때까지 2~3 반복

K-means

- 동작 예시



Distance / similarity

- 군집화의 목표
  - 가까운 샘플끼리는 같은 군집으로, 먼 샘플끼리는 다른 군집으로 묶기
    - 거리 계산 방법이 중요

거리(distance)와 유사도(similarity)는 반대개념이며
한 쪽을 알면 다른 쪽으로 변환 가능

거리 ←→ 유사도 변환 예시:

샘플 i와 j간의 거리 d_{ij}

샘플 i와 j간의 유사도 $s_{ij} = d_{\max} - d_{ij}$

$d_{\max}$: 최대 거리

Distance measure

- 두 점(특징 벡터)간의 거리 계산
  - 실수 데이터간의 거리 계산 예시
    - Minkowski distance

$\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T \quad \mathbf{x}_j = (x_{j1}, \ldots, x_{jd})^T$ 간의 Minkowski distance:

$$d_{ij} = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p \right)^{1/p}$$

$p = 2$ $\quad d_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$ Euclidean distance

$p = 1$ $\quad d_{ij} = \sum_{k=1}^{d} |x_{ik} - x_{jk}|$ Manhattan distance

Distance measure

- 두 점(특징 벡터)간의 거리 계산
  - 이진 데이터간의 거리 계산 예시
    - Hamming distance

$$(1,0,1,0,0,0,1,1)^T$$

$$(1,0,0,1,0,0,1,0)^T$$

서로 다른 비트의 개수 : 3

Similarity measure

- 두 점(특징 벡터)간의 유사도 계산
  - 실수 데이터간의 유사도 계산 예시
    - Cosine similarity

$$s_{ij} = \cos\theta = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

-1 (두 벡터의 방향이 반대) ~ 1 (두 벡터의 방향이 동일)

  - 이진 데이터간의 유사도 계산 예시

$$s_{ij} = \frac{n_{00} + n_{11}}{n_{00} + n_{11} + n_{01} + n_{10}}$$

$$s_{ij} = \frac{n_{11}}{n_{11} + n_{01} + n_{10}}$$

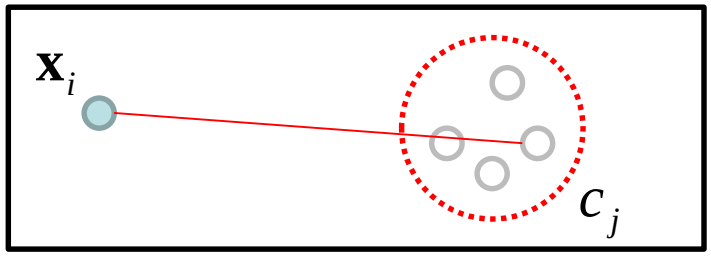
n_{01} : i는 0이고 j는 1인 특징의 개수

Distance measure

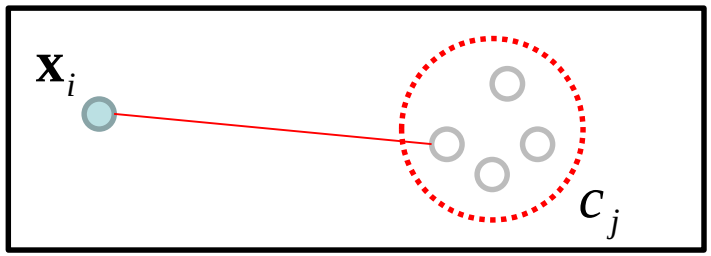
- 점과 군집 사이의 거리 계산

어떤 샘플 $\mathbf{x}_i$와 군집 c_j 간의 거리 : $\mathbf{y}_k$: c_j에 속한 샘플

가장 먼 점과의 거리 : $D_{\max}\left(\mathbf{x}_i, c_j\right) = \max_{\mathbf{y}_k \in c_j} d_{ik}$



가장 가까운 점과의 거리 : $D_{\min}\left(\mathbf{x}_i, c_j\right) = \min_{\mathbf{y}_k \in c_j} d_{ik}$

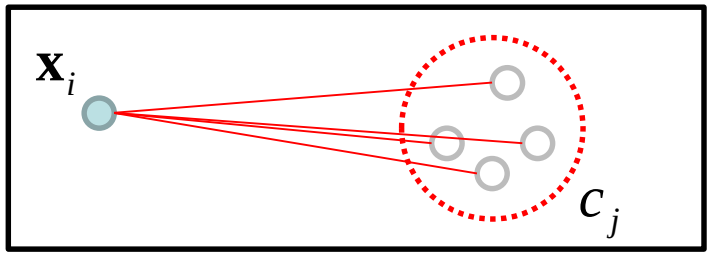


Distance measure

- 점과 군집 사이의 거리 계산

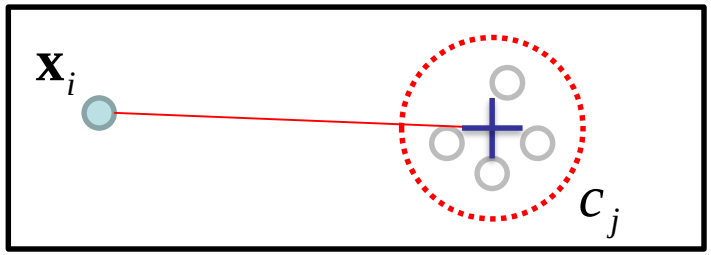
어떤 샘플 $\mathbf{x}_i$와 군집 c_j 간의 거리 : $\mathbf{y}_k$: c_j에 속한 샘플

모든 점과의 평균 거리 : $D_{ave}\left(\mathbf{x}_i, c_j\right) = \frac{1}{\left|c_j\right|} \sum_{\mathbf{y}_k \in c_j} d_{ik}$



대표점과의 거리: $D_{mean}\left(\mathbf{x}_i, c_j\right) = d_{i,mean}$

$\mathbf{y}_{mean} = \frac{1}{\left|c_j\right|} \sum_{\mathbf{y}_k \in c_j} \mathbf{y}_k$



Hierarchical clustering

- 응집 (agglomerative) 방식
  - 작은 군집들로부터 시작해서 점차 큰 군집으로 병합
  - bottom-up 방식

- 분열 (divisive) 방식
  - 큰 군집에서 시작하여 나눔
  - top-down 방식

Hierarchical clustering

- 응집 (agglomerative) 방식
  - M회 반복

$$C_0 = \{c_1 = \{\mathbf{x}_1\}, \ldots, c_N = \{\mathbf{x}_N\}\}$$

For (t=1 to M-1)

{

C_{t-1} 의 모든 군집 쌍 (c_i, c_j) 중 다음을 만족하는 쌍 (c_p, c_q) 탐색

$$D(c_p, c_q) = \min_{c_i, c_j \in C_{t-1}} D(c_i, c_j)$$

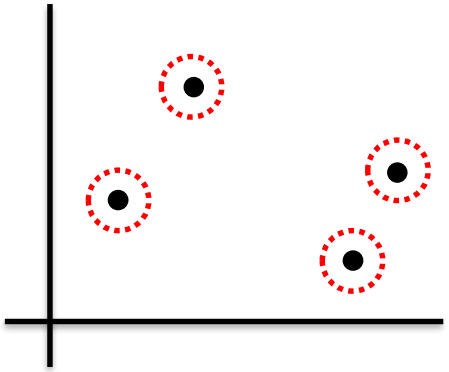
$$c_r = c_p \cup c_q$$

$$C_t = (C_{t-1} - c_p - c_q) \cup c_r$$

}

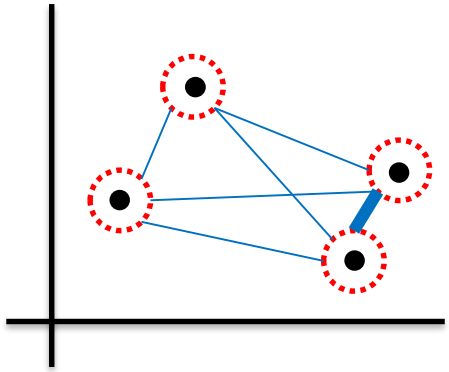
Hierarchical clustering

- 응집 (agglomerative) 방식



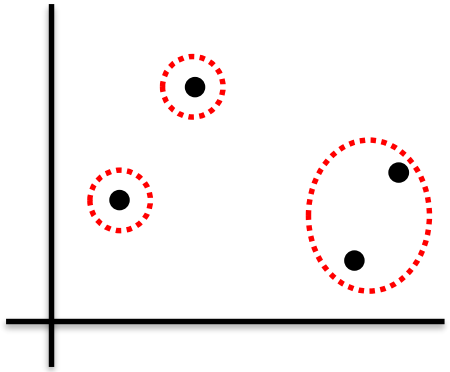
$$C_0 = \{c_1 = \{\mathbf{x}_1\}, ..., c_N = \{\mathbf{x}_N\}\}$$

각 샘플이
서로 다른 군집을 구성



$$D(c_p, c_q) = \min_{c_i, c_j \in C_{t-1}} D(c_i, c_j)$$

가장 거리가 짧은 군집 쌍 탐색



$$c_r = c_p \cup c_q$$

병합

Hierarchical clustering

- 분열 (divisive) 방식
  - M회 반복

$C_0 = \{c_1 = \{x_1, x_2, \ldots, x_N\}\}$

For (t=1 to M−1)

{

for (C_{t-1}의 모든 군집 c_i 에 대하여)

c_i 의 모든 이진 분할 중에 거리가 가장 먼 것 탐색

앞에서 찾은 t개의 이진 분할 중 가장 거리가 먼 군집 c_q를 c_q^1 c_q^2로 분할

$$C_t = \left(C_{t-1} - c_q\right) \cup \left\{c_q^1, c_q^2\right\}$$

}

계산량이 매우 많음

QnA

실습

School of Computer Science
University of Seoul

실습과제 (1 / 3)

- MNIST DB 다운로드 받기
  - http://yann.lecun.com/exdb/mnist/

- 학습 과정(train.py)
  - CNN 숫자인식기 학습
    - keras(theano backend)를 이용하여 구현
    - 세부 설계는 자유(단, convolutional layer를 하나 이상 사용할 것)
    - 엄격하게 validation을 할 필요는 없음(테스트 오류가 적당히 줄어들면 중단)
    - 테스트 오류율이 10% 이내로는 줄어들도록
  - 매 학습 반복(epoch)마다 오류율 혹은 코스트 콘솔 출력 + 텍스트 파일로 저장
    - “train_log.txt”
  - 가장 좋은 모델 파라미터를 파일로 저장
    - “best_param.h5”

실습과제 (2 / 3)

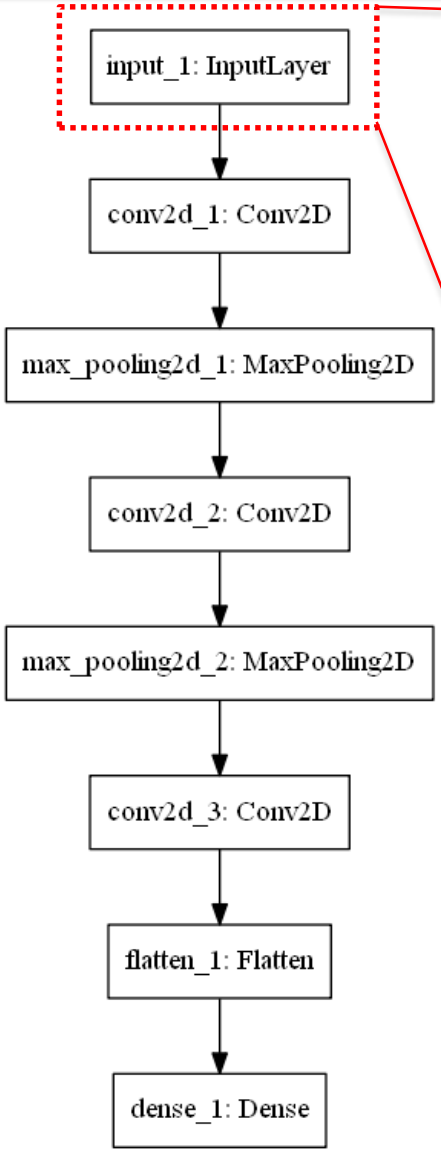
- 테스트 과정(test.py)
  - 파일로 저장한 가장 좋은 모델 파라미터 읽기
    - “best_param.h5”
  - 분류 결과를 출력 + 텍스트 파일로 저장
    - 각 샘플별 분류 결과는 너무 많으니 콘솔 출력하지 말고 텍스트 파일로만 저장
    - 마지막에 오류율 계산하여 보여주기
    - “test_output.txt”
  - 테스트 데이터 중 몇 개 샘플에 대해서만 feature map 그려보기
    - 2종류의 숫자 클래스 + 각 숫자 클래스별 3개 샘플
    - 중간의 convolutional layer activation을 계산하여 이미지 파일로 출력
      - 파일명: “[class_label]_[sample_index]_[filter_index].png”

- 주의사항
  - 학습 스크립트와 테스트 스크립트를 분리할 것
  - 가장 좋은 모델 파라미터를 파일로 저장하여 동봉할 것
  - 테스트 스크립트에서는 학습이 완료된 모델 파라미터를 불러와 사용할 것

실습과제 (3 / 3)

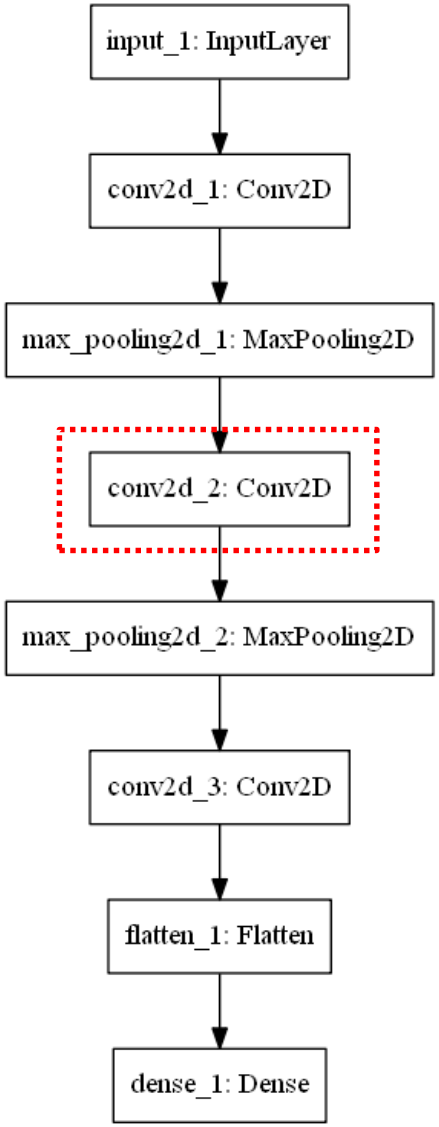
- 제출 형식
  - 입력 파일은 굳이 같이 제출할 필요 없음(필요한 경우는 함께 제출)
    - 단, 한 단계 상위 폴더에서 로드하기
      - 학습 데이터
        - ≫ ../train-images.idx3-ubyte
        - ≫ ../train-labels.idx1-ubyte
      - 테스트 데이터
        - ≫ ../t10k-images.idx3-ubyte
        - ≫ ../t10k-labels.idx1-ubyte
  - 출력 파일과 함께 제출
    - 학습 스크립트(train.py) / 테스트 스크립트(test.py)
    - 학습 로그("train_log.txt") / 테스트 결과("test_output.txt")
    - 가장 좋은 모델 파라미터 저장 파일("best_param.h5")
      - (필요한 경우) feature map 추출을 위한 파라미터를 별도 파일로 저장
      - Ex〉 "best_param_for_feature_map.h5"
    - Feature map 이미지 파일들("*.png")

실습과제 예시

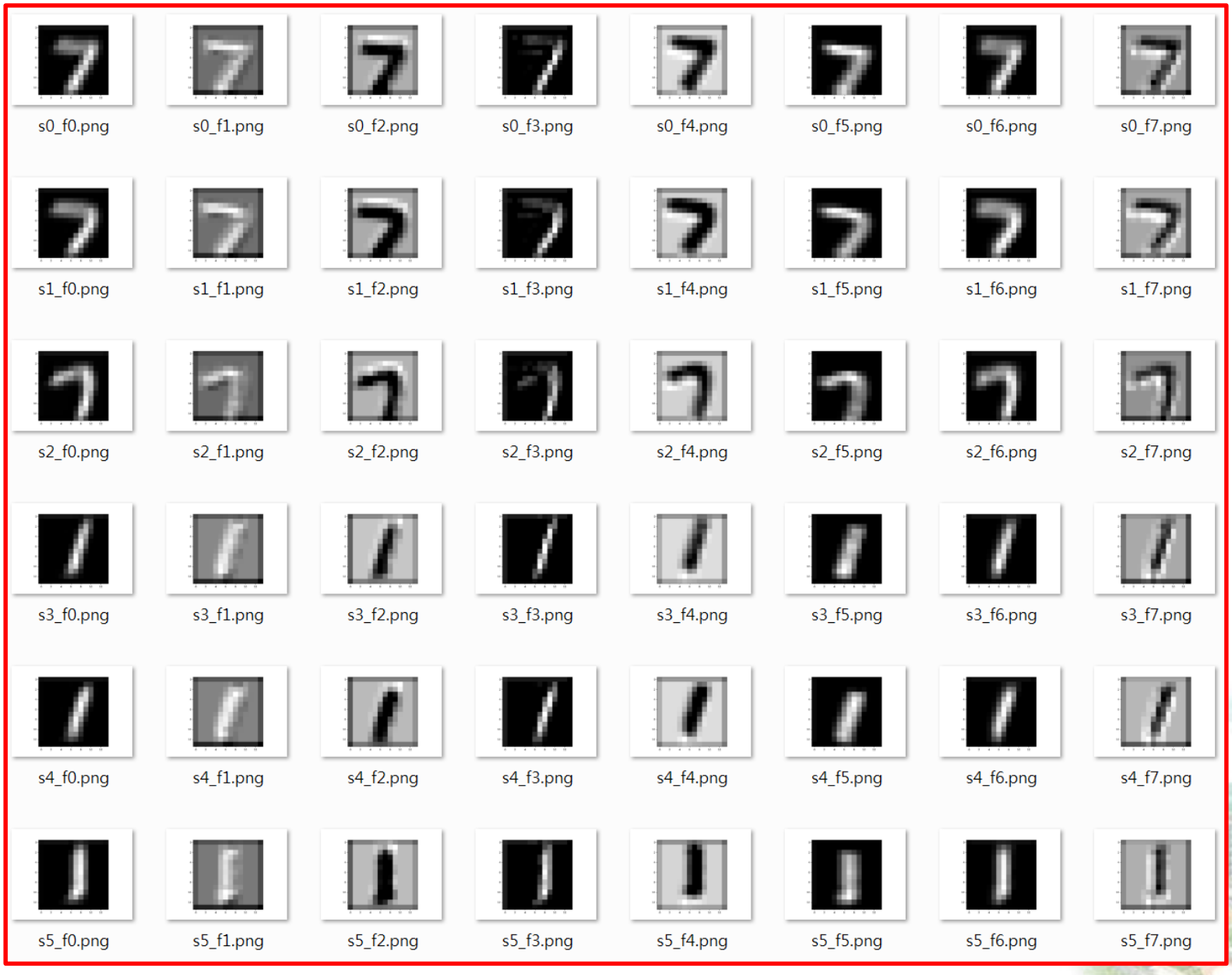

input_1: InputLayer
conv2d_1: Conv2D
max_pooling2d_1: MaxPooling2D
conv2d_2: Conv2D
max_pooling2d_2: MaxPooling2D
conv2d_3: Conv2D
flatten_1: Flatten
dense_1: Dense


s0_f0.png
s1_f0.png
s2_f0.png
s3_f0.png
s4_f0.png
s5_f0.png

실습과제 예시


input_1: InputLayer
conv2d_1: Conv2D
max_pooling2d_1: MaxPooling2D
conv2d_2: Conv2D
max_pooling2d_2: MaxPooling2D
conv2d_3: Conv2D
flatten_1: Flatten
dense_1: Dense

실습과제 예시


s0_f0.png
s0_f1.png
s0_f2.png
s0_f3.png
s0_f4.png
s0_f5.png
s0_f6.png
s0_f7.png
s1_f0.png
s1_f1.png
s1_f2.png
s1_f3.png
s1_f4.png
s1_f5.png
s1_f6.png
s1_f7.png
s2_f0.png
s2_f1.png
s2_f2.png
s2_f3.png
s2_f4.png
s2_f5.png
s2_f6.png
s2_f7.png
s3_f0.png
s3_f1.png
s3_f2.png
s3_f3.png
s3_f4.png
s3_f5.png
s3_f6.png
s3_f7.png
s4_f0.png
s4_f1.png
s4_f2.png
s4_f3.png
s4_f4.png
s4_f5.png
s4_f6.png
s4_f7.png
s5_f0.png
s5_f1.png
s5_f2.png
s5_f3.png
s5_f4.png
s5_f5.png
s5_f6.png
s5_f7.png

이미지 저장 방법

```
import matplotlib.pyplot as plt

def drawFeatMap(dstFn, featMap):
    fig, ax = plt.subplots()
    ax.imshow(featMap, cmap='gray')
    plt.savefig(dstFn)

    # flush
    plt.cla()
    plt.clf()
    plt.close()
```

중간 레이어 출력 얻기

```
# 모델 구성(2(input) -> CONV(ReLU) -> CONV(ReLU) -> FC(sigmoid))
inputFeat = kl.Input(shape=(4, 4, 1))

conv1 = kl.Conv2D(filters=5,
                  kernel_size=(3, 3), strides=1,
                  padding='same')(inputFeat)    # zero-padding
relu1 = kl.Activation('relu')(conv1)
conv2 = kl.Conv2D(filters=3,
                  kernel_size=(3, 3), strides=1)(relu1)
relu2 = kl.Activation('relu')(conv2)
flatten = kl.Flatten()(relu2)
dense = kl.Dense(units=2)(flatten)
output = kl.Activation('sigmoid')(dense)

modelFull = km.Model(inputs=[inputFeat], outputs=[output])
modelRelu1 = km.Model(inputs=[inputFeat], outputs=[relu1])
```

중간 레이어(relu1)의
activation을 출력하는 모델 별도 정의

QnA