

# cs231n week2

Image Classification

발표자 20 김현서

# 목차

01

---

Image Classification

02

---

Nearest Neighbor

03

---

Hyperparameters

04

---

Linear Classification

01.

# Image Classification

사람과 달리 컴퓨터는 image를 0~255 사이의 숫자로 인식.

이에 따라 컴퓨터가 image를 분류하는 것을  
image classification이라 함.



08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 31 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 05 30 03 49 13 36 65
52 70 95 23 04 60 11 42 63 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 62 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 33 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 79 33 27 98 66
66 34 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 55 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 69 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 66 61 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

What the computer sees

→ 82% cat  
15% dog  
2% hat  
1% mug

image classification

01.

# Image Classification

---

Challenge: Viewpoint variation / Scale variation / Deformation / Occlusion / Illumination conditions  
Background clutter / Intra-class variation

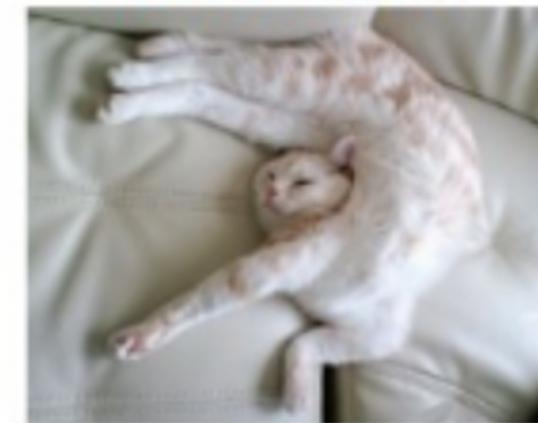
Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter



Intra-class variation

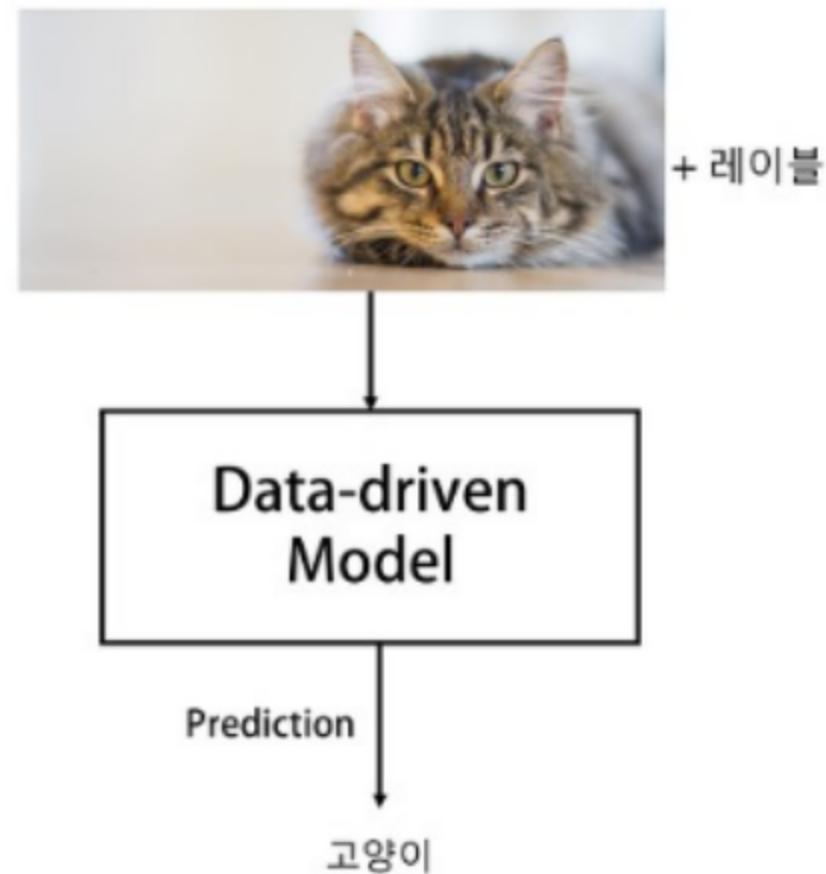


01.

# Image Classification

---

## Data-driven approach

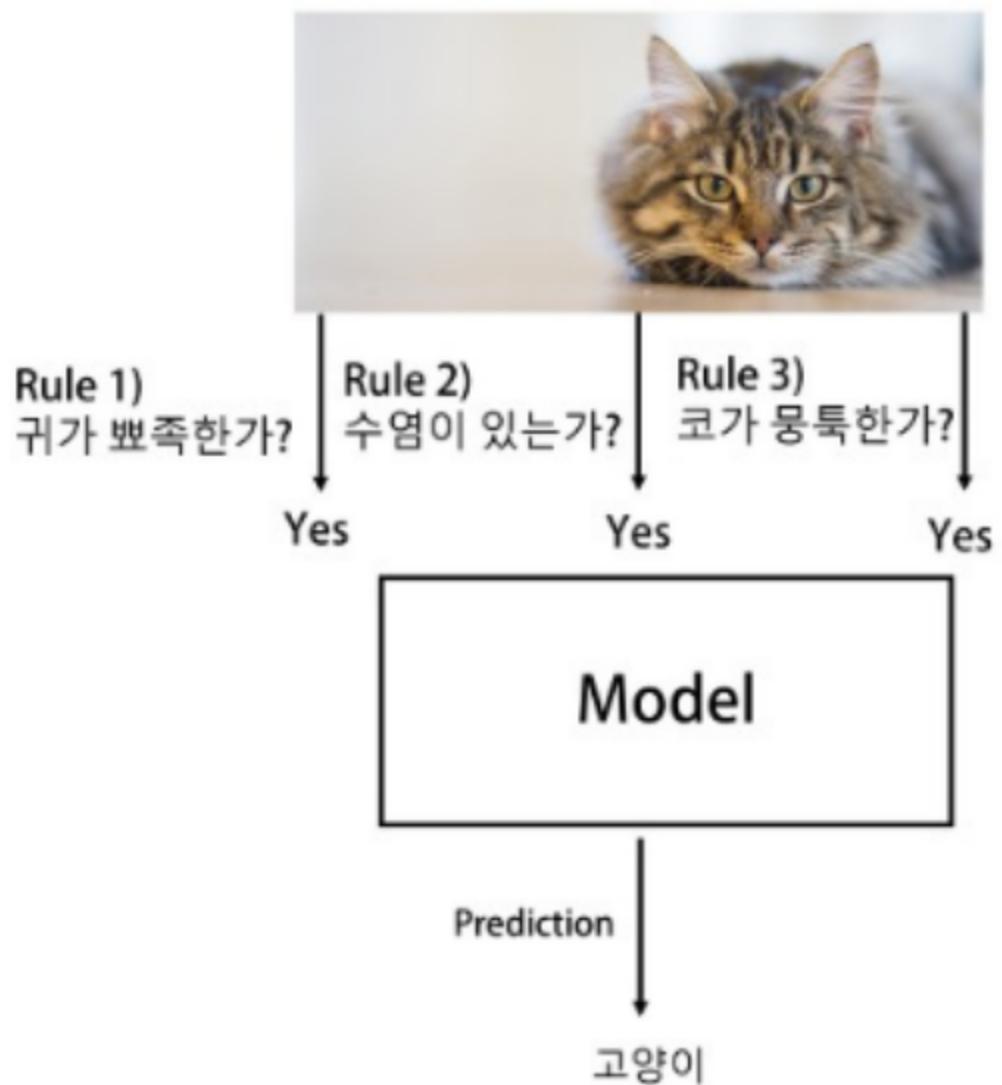


1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

01.

# Image Classification

## Rule-based approach



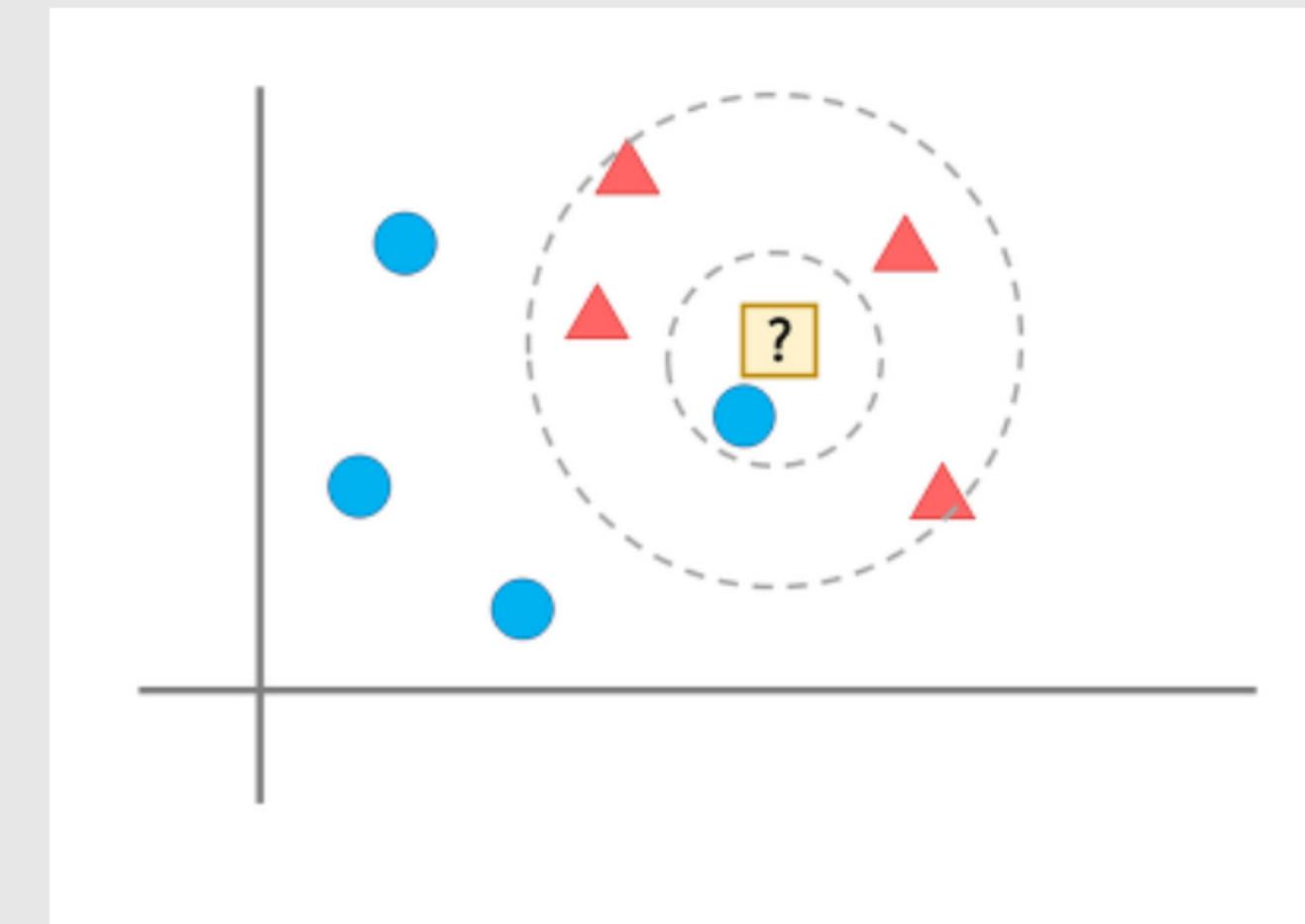
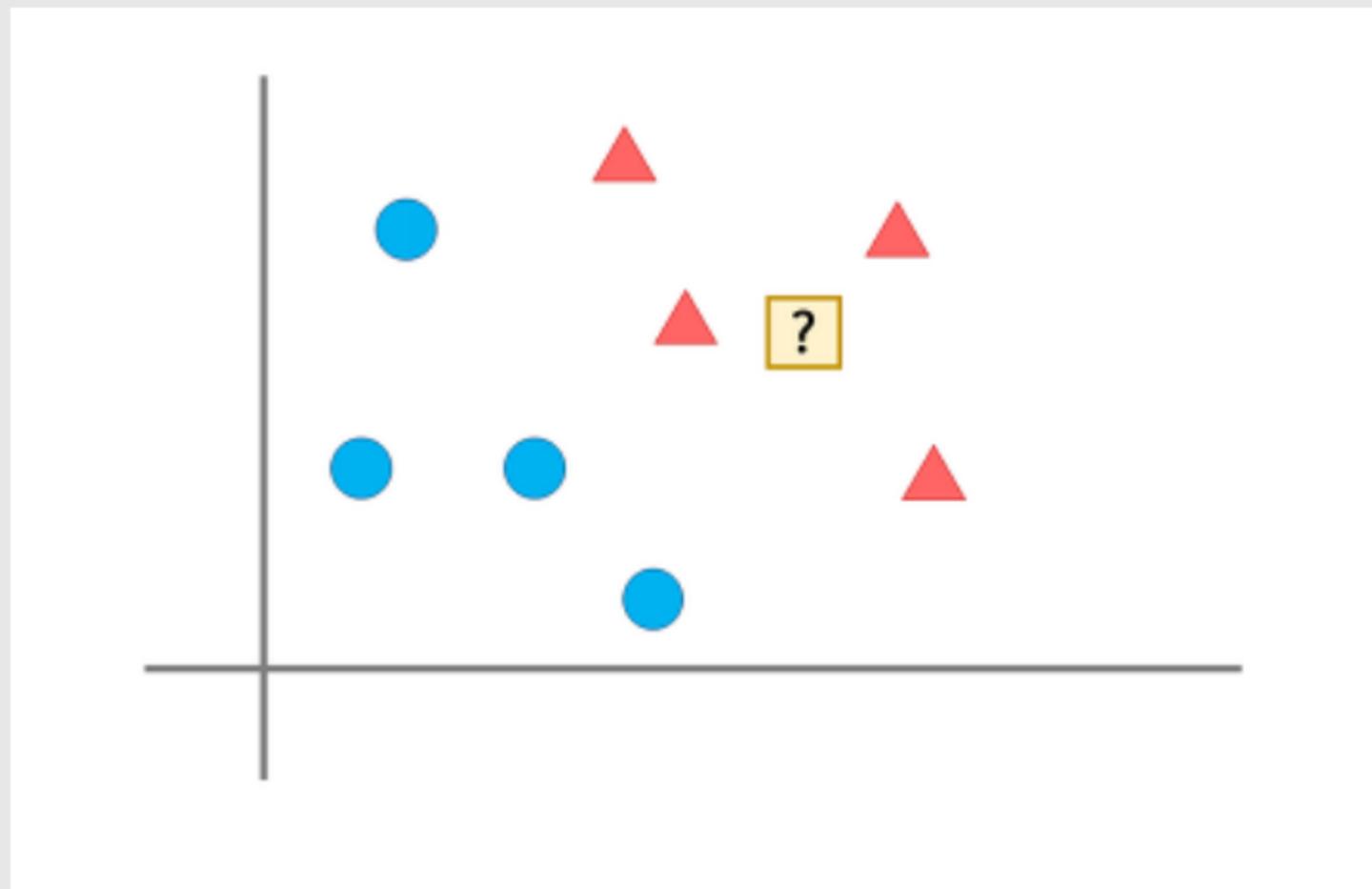
주어진 입력에 대해서 결과값을 도출하는 방법

규칙(rule)에 따라 학습 및 예측을 하는 방법

02.

## Nearest Neighbor

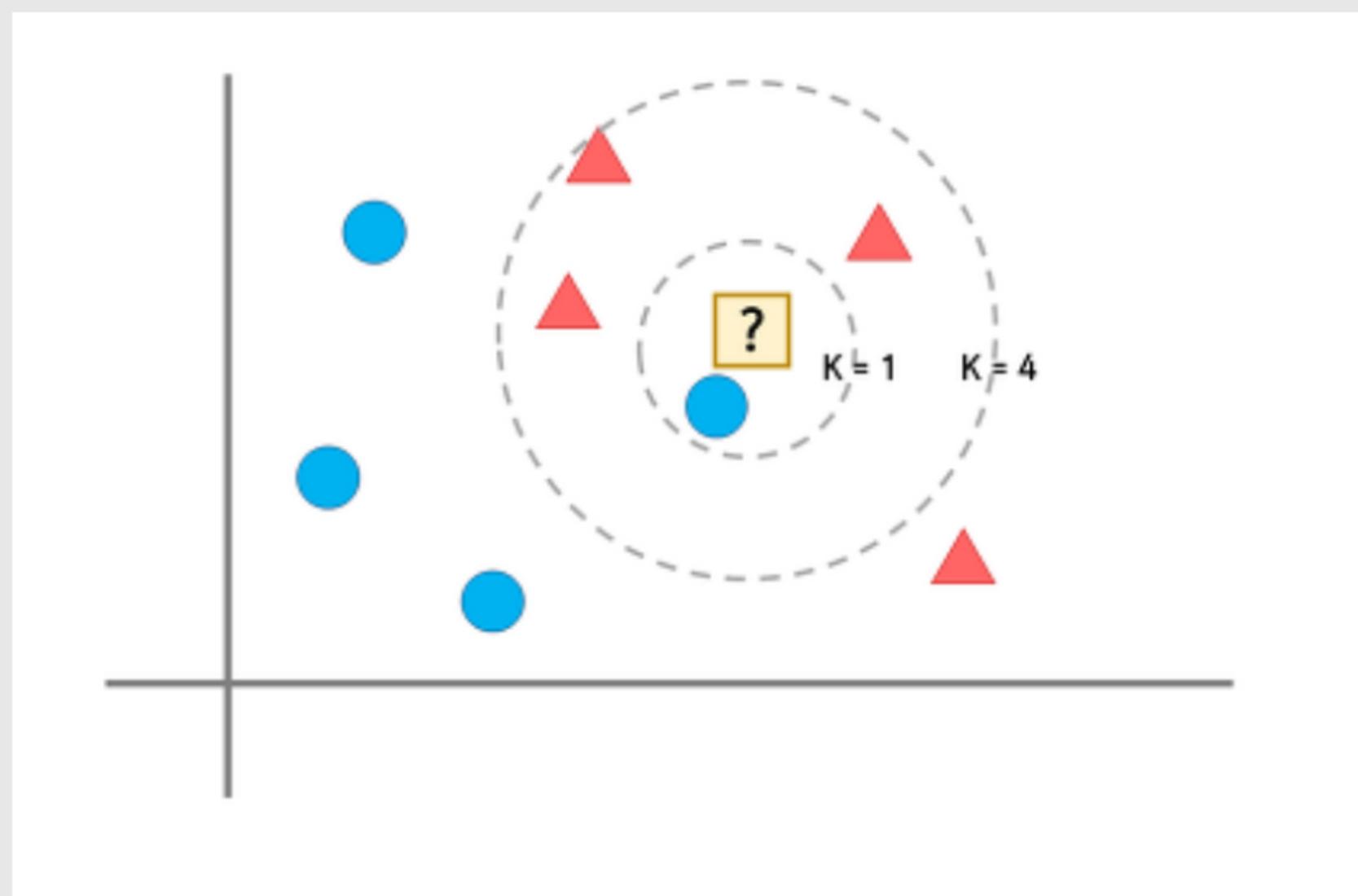
새로운 데이터를 입력받았을 때, 가장 가까운 데이터의 종류에 따라 새로운 데이터의 종류를 정함.



02.

## k-Nearest Neighbor

예측 단계에서 input과 가까운 순으로 총 k 개의 데이터의 label을 구하여 가장 빈번하게 나오는 label로 예측하는 방법. (Voting)



$k=1$  : 파란 원

$k=4$  : 빨간 세모

02.

## k-Nearest Neighbor

---

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

Depend on coordinate system

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

Independent

02.

## k-Nearest Neighbor

Nearest Neighbor은 images에는 사용되지 않음.

1. Test computation 이 매우 느리기 때문에
2. Distance를 측정하기 적절하지 않음



모두 동일한 distance로 나타남 - 이미지들 간의 perceptual similarity 측정에 적절하지 않음

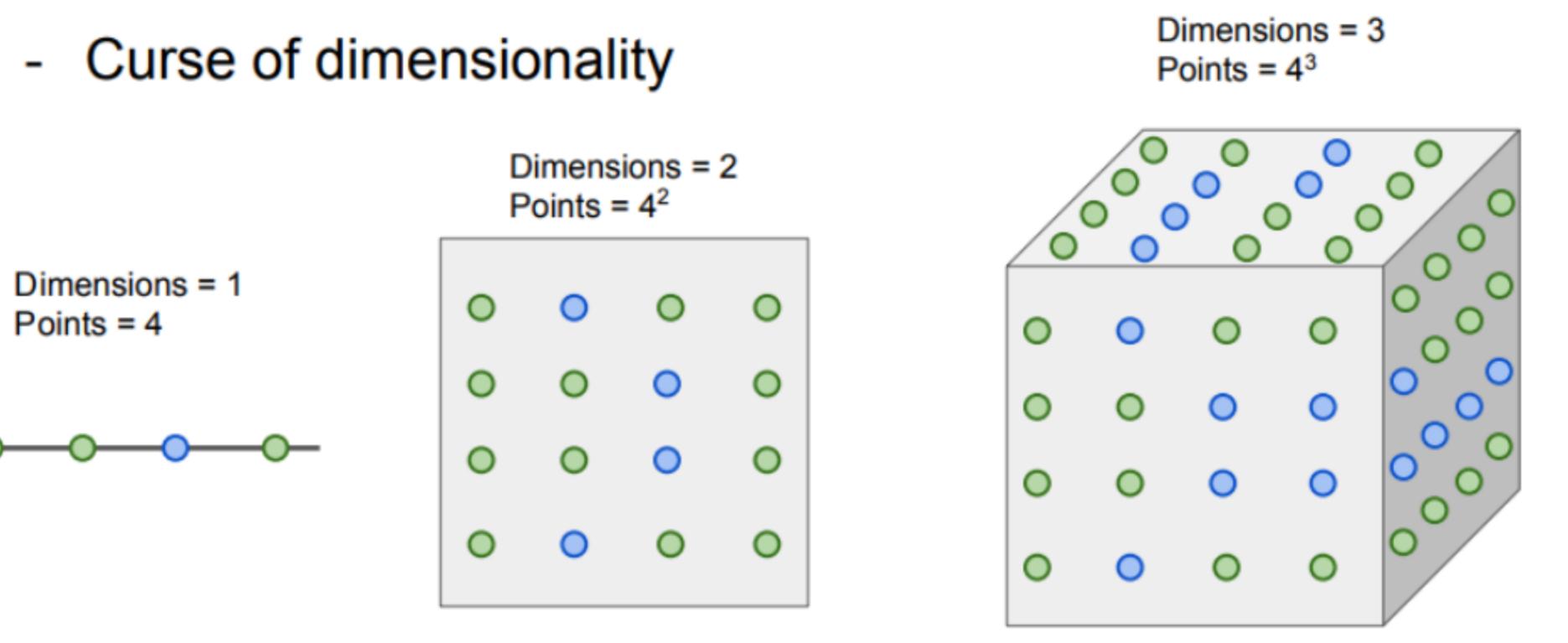
02.

## k-Nearest Neighbor

Nearest Neighbor은 images에는 사용되지 않음.

### 3. Curse of dimensionality

#### - Curse of dimensionality



전체 공간을 채울만큼 많은 데이터가 필요함 - 차원이 증가하면 필요한 데이터가 기하급수적으로 증가함

03.

## Hyperparameters

---

k, distance 등 algorithm에 영향을 끼치는 parameters

학습데이터의 정확도와 성능을 최대화하는 Hyperparameters를 얻기 위해 training set의 일부를 validation set으로 이용.

03.

# Hyperparameters

---

**Idea #1:** Choose hyperparameters  
that work best on the data

**BAD:** K = 1 always works  
perfectly on training data

Your Dataset

train set에만 집중하여 새로운 데이터를 예측하는 데 적절하지 않음.

ex. k=1인 경우 : train set을 완벽하게 분류할 수 있지만 새로운 데이터 분류에는 용이하지 않음.

03.

# Hyperparameters

---

**Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data

**BAD:** No idea how algorithm will perform on new data

train

test

train set으로 다양한 hyperparameters 값들을 학습을 시킨 후,  
test set에 적용하여 적절한 hyperparameters를 선택.

→test set에만 적절하고 새로운 데이터 분류에는 용이하지 않음.

03.

# Hyperparameters

---

**Idea #3:** Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!



train set으로 다양한 hyperparameters 값을 학습을 시킨 후,  
validation set에 적용하여 적절한 hyperparameters를 선택,  
이후 test set에 적용함.

→적절함.

03.

# Hyperparameters

**Idea #4: Cross-Validation:** Split data into **folds**,  
try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

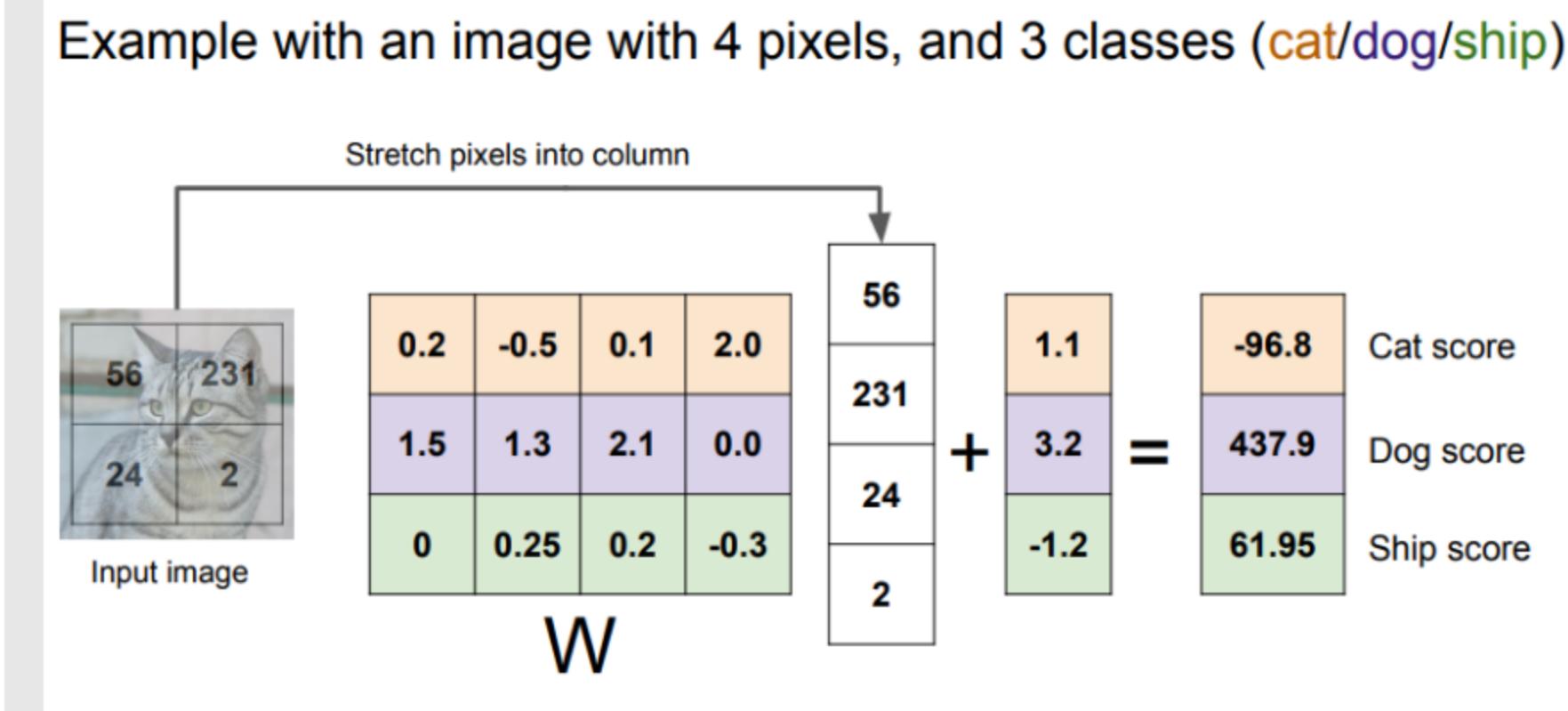
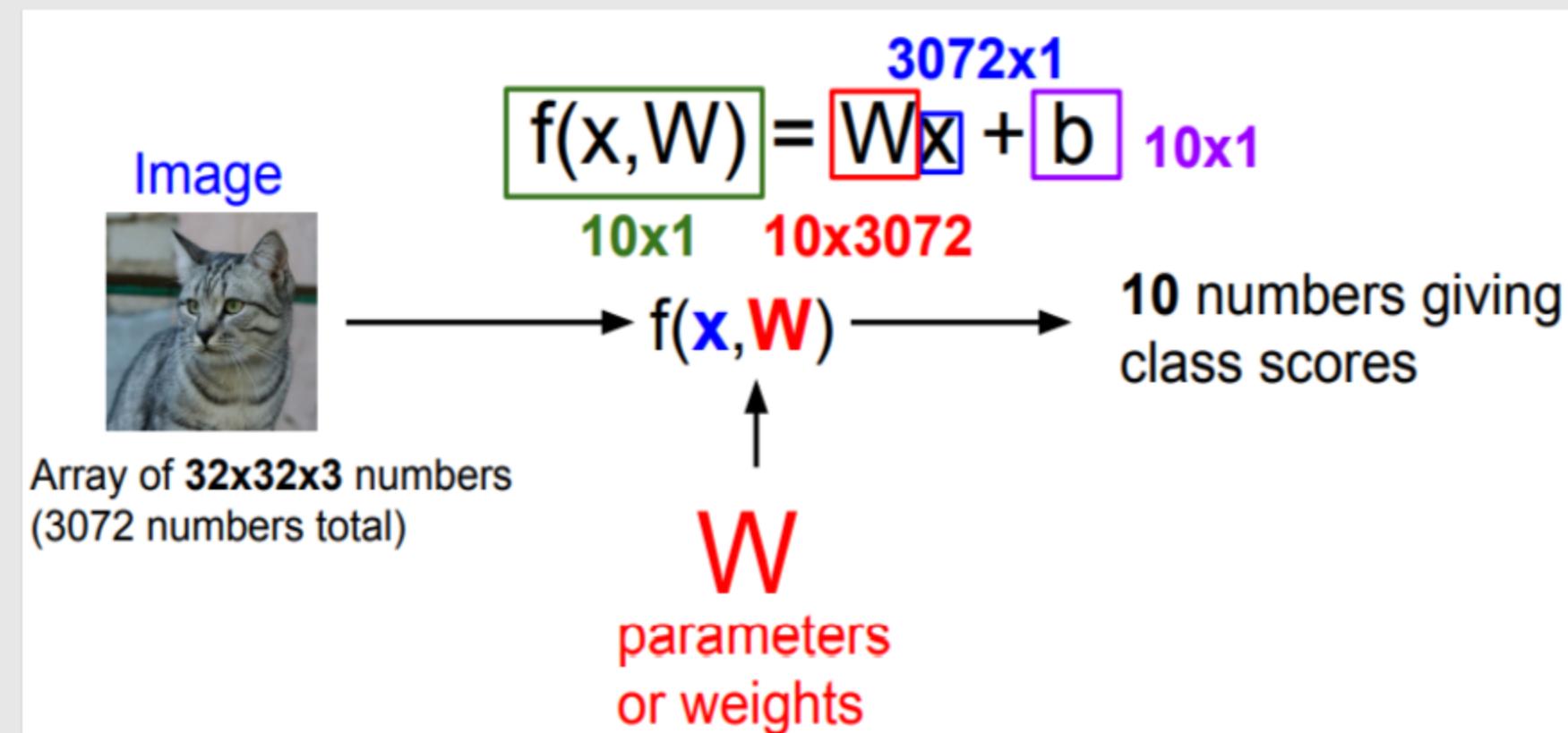
Useful for small datasets, but not used too frequently in deep learning

train set을 여러 개(fold)로 나누고 번갈아가며 validation으로 사용함.  
이후 그 결과의 평균을 이용해 hyperparameters를 구함.

→ 계산량이 많아 작은 dataset에 적절함.

04.

# Linear Classification



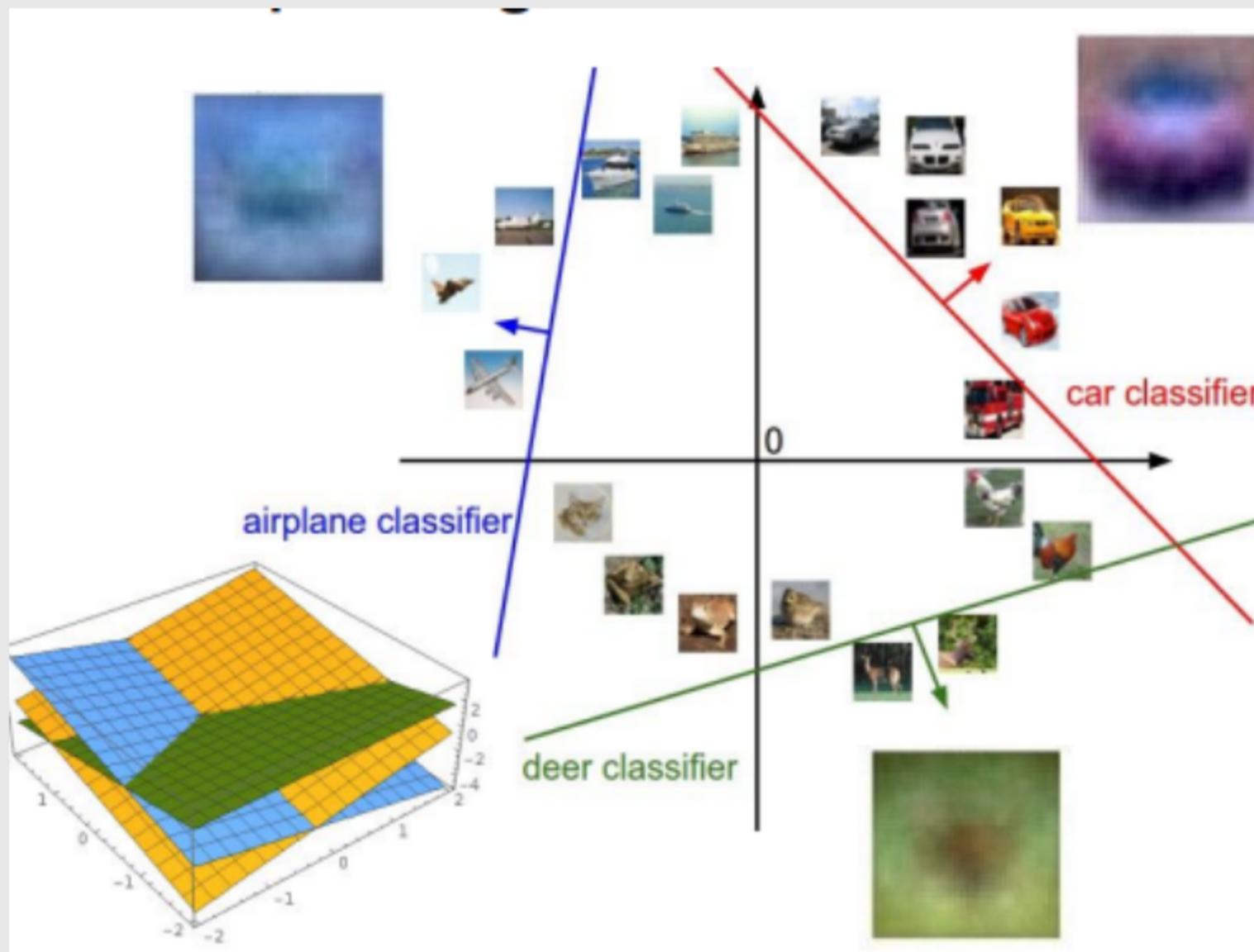
Weight : train set의 정보. 선형 회귀가 학습한 직선의 기울기.

ex. 집을 고르는 경우 : 집의 크기, 주변 교통, 지역 등이 각각 다른 비중치(Weight)를 가짐

Bias : 데이터와 무관하게 더해지는 값. 각 클래스에 더해지는 scaling offsets.

04.

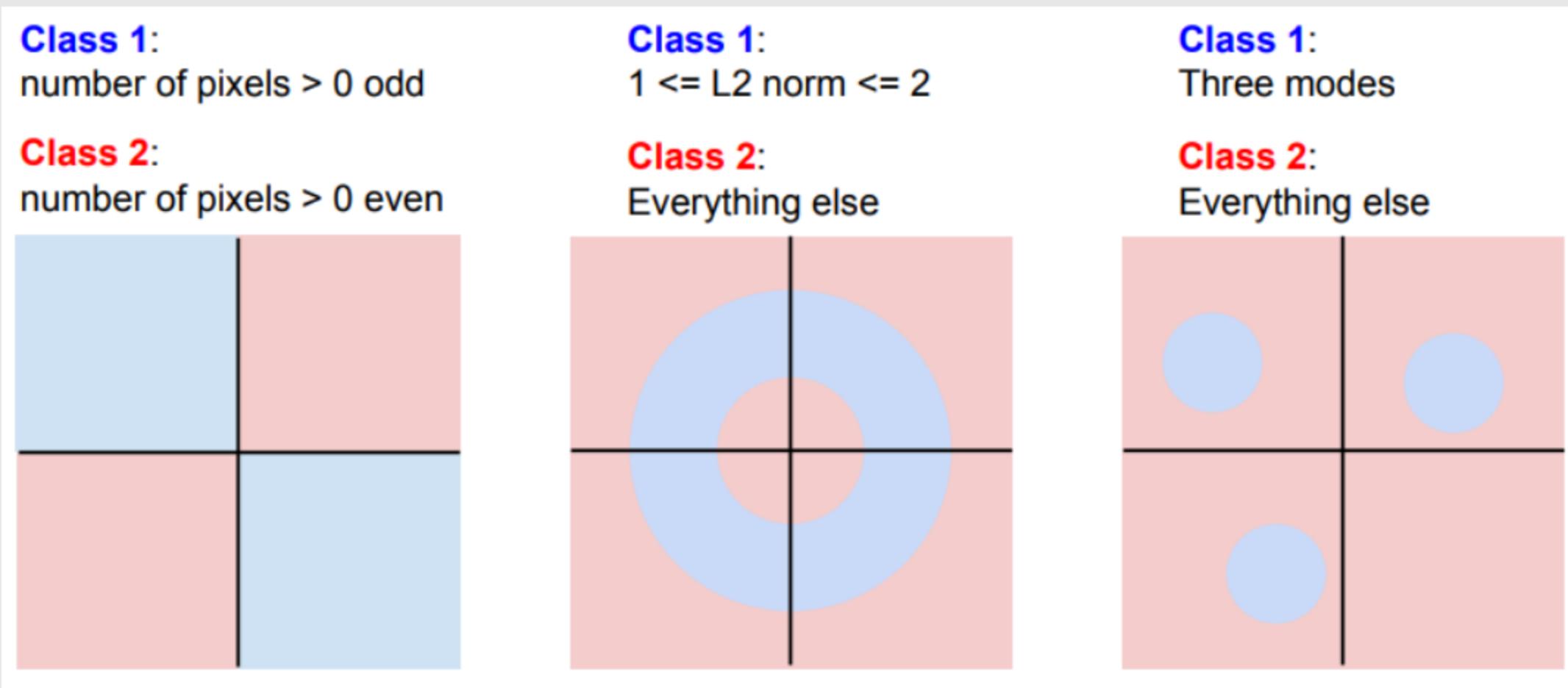
# Linear Classification



각 클래스에 해당하는 하나의 template을 기반으로  
각 클래스를 구분하는 linear classifier을 만듦.

04.

# Linear Classification



위와 같이 하나의 linear classifier로 두 개의 클래스를 구분할 수 없음.

# THANK YOU

cs231n week2

---