[2주차]
Image Classification

박가현 황시은

Semantic Gap

강아지

```
array([[[194, 214, 219],
        [202, 222, 227],
        [212, 232, 237],
        ...,
        [ 77,  85,  85],
        [ 76,  84,  84],
        [ 73,  81,  81]],

       [[188, 208, 213],
        [190, 210, 215],
        [196, 216, 221],
        ...,
        [ 74,  82,  82],
        [ 74,  82,  82],
        [ 75,  83,  83]],

       [[184, 204, 209],
        [184, 204, 209],
        [187, 206, 213],
        ...,
```

(2048, 1536, 3)

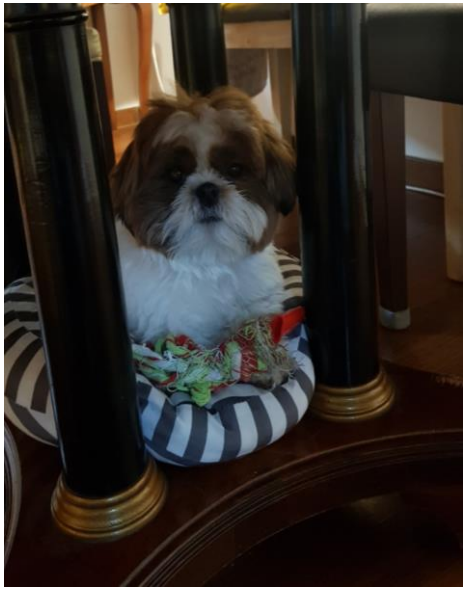
규칙기반 방법

규칙을 만든다	뾰족한 귀가 있다. 수염이 있다 눈꼬리가 위로 올라가있다
알고리즘을 통해 규칙 확인	입력받은 이미지가 위의 특징 만족?
판단	YES. 고양이다 NO. 고양이가 아니다

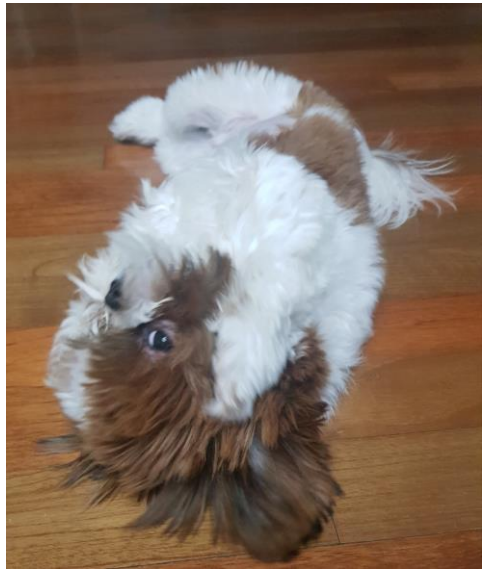
Challenge

View Point (사진 찍은 각도)

Illumination (밝기)



Deformation(변형)



Challenge

Occulusion

Background Clutter

Interclass variation

Challenge

```python
def isDog(img_arr):
    flag=0
    for i in img_arr:
        if(i<img.shape[0]):
            if(center_pixel == '        '):
                ...
        else:
            ...


    return flag

```

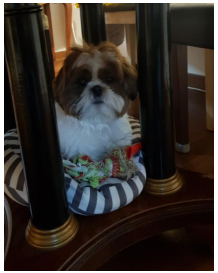
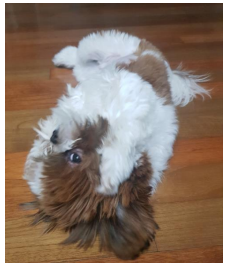
단순한
코드로
이렇게 저렇게

규칙 만들기

Data Driven Approach

Data Driven Approach



확장성



안정성

강아지

Data Driven Approach

Dataset

- Labeling 된 이미지 수집

Train

- 머신러닝 이용 classifier 학습

Evaluate

- 새로운 이미지 -> 성능 확인

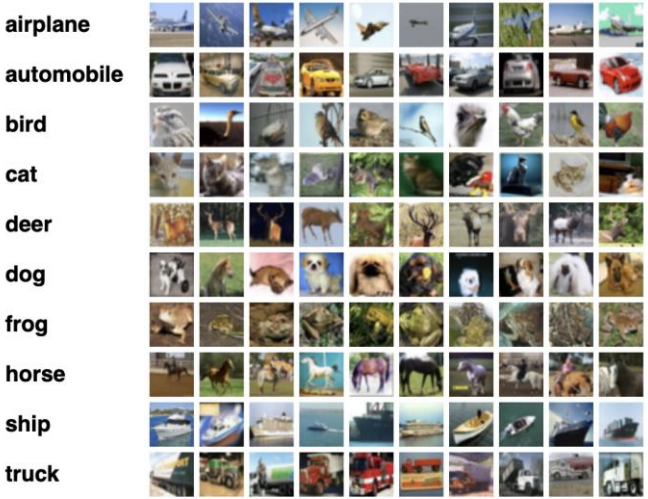
Dataset

The CIFAR-10 dataset

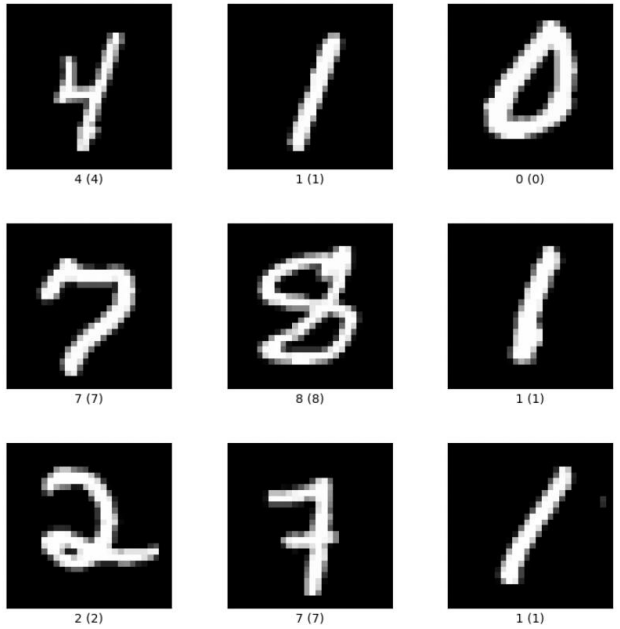
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 ima

The dataset is divided into five training batches and one test batch, each with 10000 image contain the remaining images in random order, but some training batches may contain mor from each class.

Here are the classes in the dataset, as well as 10 random images from each:



MNIST

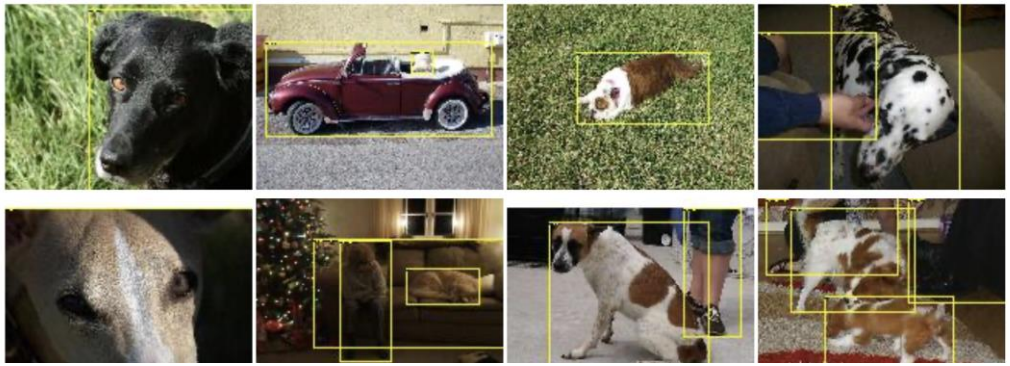


Dataset

- Aeroplanes
- Bicycles
- Birds
- Boats
- Bottles
- Buses
- Cars
- Cats
- Chairs
- Cows
- Dining tables
- Dogs
- Horses
- Motorbikes
- People
- Potted plants
- Sheep
- Sofas
- Trains
- TV/Monitors

Dining tables - *all images contain at least one dining table.*

Dogs - *all images contain at least one dog.*



http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Dataset

ImageNet

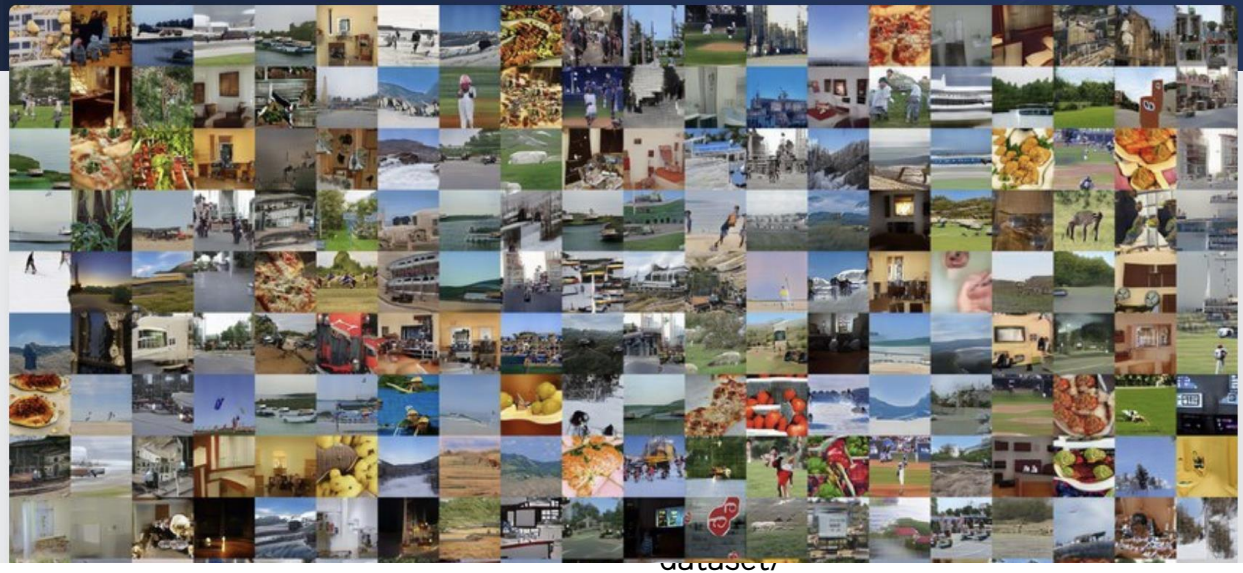
Introduced by Jia Deng et al. in ImageNet: A large-scale hierarchical image database

The **ImageNet** dataset contains 14,197,122 annotated images according to the WordNet hierarchy. Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection. The publicly released dataset contains a set of manually annotated training images. A set of test images is also released, with the manual annotations withheld. ILSVRC annotations fall into one of two categories: (1) image-level annotation of a binary label for the presence or absence of an object class in the image, e.g., "there are cars in this image" but "there are no tigers," and (2) object-level annotation of a tight bounding box and class label around an object instance in the image, e.g., "there is a screwdriver centered at position (20,25) with width of 50 pixels and height of 30 pixels". The ImageNet project does not own the copyright of the images, therefore only thumbnails and URLs of images are provided.

https://paperswithcode.com/dataset/imagenet

Dataset

COCO/MS-COCO Dataset



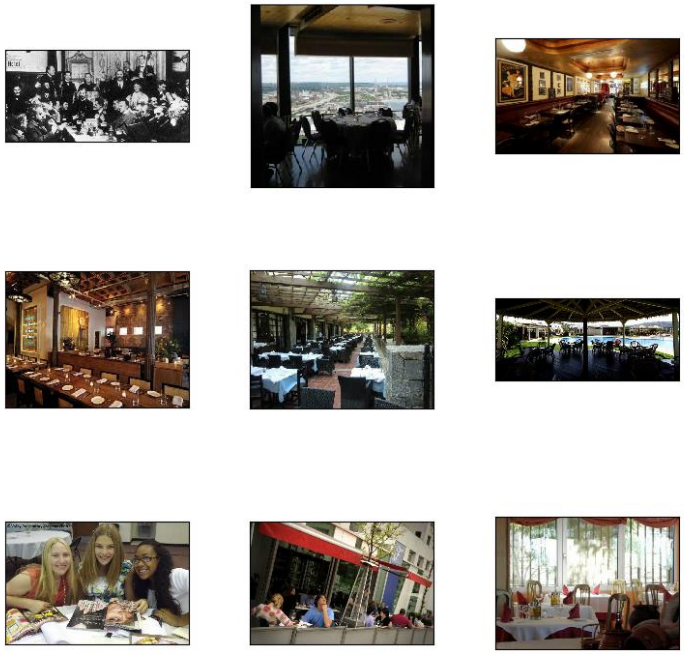
https://content.alegion.com/datasets/coco-ms-coco-dataset

Dataset

Sample Images

http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Dataset



https://www.tensorflow.org/datasets/catalog/lsun

Index of /lsun/objects/

../
airplane.zip
airplane.zip.md5
bicycle.zip
bicycle.zip.md5
bird.zip
bird.zip.md5
boat.zip
boat.zip.md5
bottle.zip
bottle.zip.md5
bus.zip
bus.zip.md5
car.zip
car.zip.md5
cat.zip
cat.zip.md5
chair.zip
chair.zip.md5
cow.zip
cow.zip.md5
dining_table.zip
dining_table.zip.md5
dog.zip
dog.zip.md5
horse.zip
horse.zip.md5
motorbike.zip
motorbike.zip.md5
person.zip
person.zip.md5
potted_plant.zip
potted_plant.zip.md5
sheep.zip
sheep.zip.md5
sofa.zip
sofa.zip.md5
train.zip
train.zip.md5
tv-monitor.zip
tv-monitor.zip.md5

Nearest Neighbor Classifier

Train 주어진 모든 데이터와 label을 ' 외움'

Predict 가장 비슷한 label 찾기

HOW?

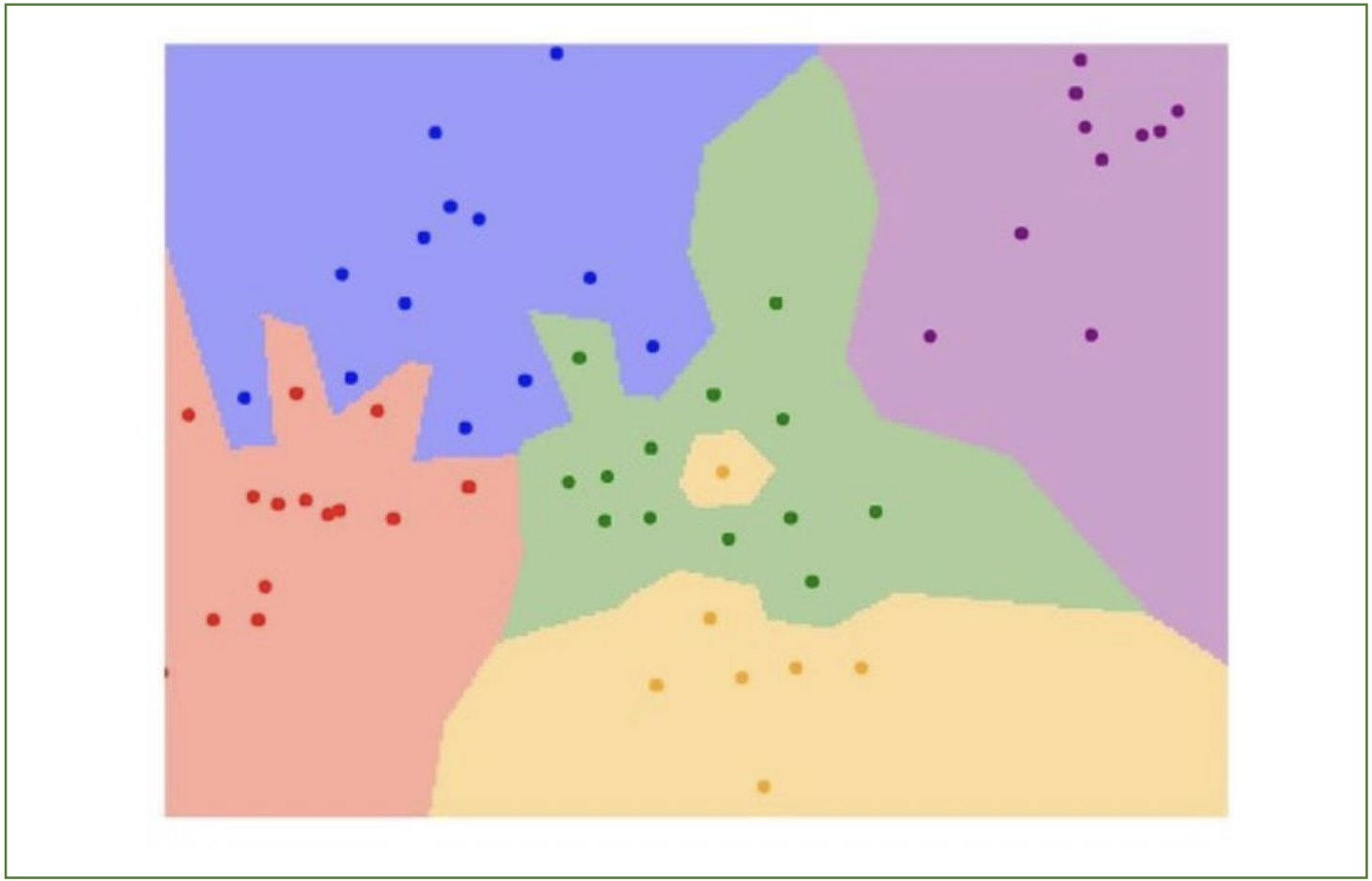
L1 distance

픽셀간의 차들의 합 (Sum of absolute values between pixels)

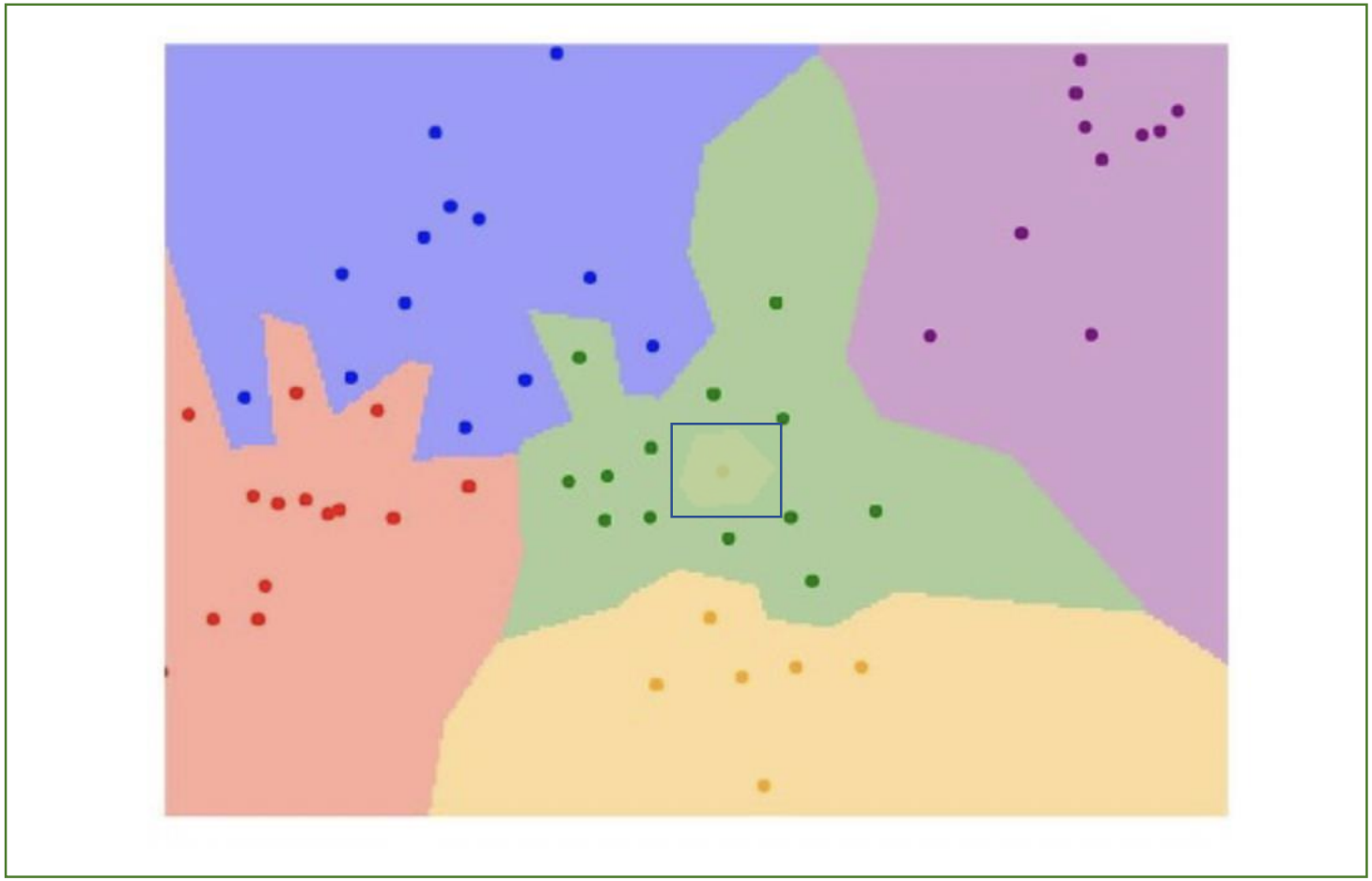
$$\sum_{k=1}^{n} |x_k| = |x_1| + |x_2| + |x_3| + \ldots |x_n|$$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

Nearest Neighbor Classifier



Nearest Neighbor Classifier



Nearest Neighbor Classifier

```
[9]  1 # Now implement the function predict_labels and run the code below:
     2 # We use k = 1 (which is Nearest Neighbor).
     3 y_test_pred = classifier.predict_labels(dists, k=1)
     4
     5 # Compute and print the fraction of correctly predicted examples
     6 num_correct = np.sum(y_test_pred == y_test)
     7 accuracy = float(num_correct) / num_test
     8 print('Got %d / %d correct => accuracy: %f' % (num_correct, num_test, accuracy))

     Got 137 / 500 correct => accuracy: 0.274000
```

```
k = 1, accuracy = 0.263000
k = 1, accuracy = 0.257000
k = 1, accuracy = 0.264000
k = 1, accuracy = 0.278000
k = 1, accuracy = 0.266000
k = 3, accuracy = 0.239000
k = 3, accuracy = 0.249000
k = 3, accuracy = 0.240000
k = 3, accuracy = 0.266000
k = 3, accuracy = 0.254000
k = 5, accuracy = 0.248000
k = 5, accuracy = 0.266000
k = 5, accuracy = 0.280000
k = 5, accuracy = 0.292000
k = 5, accuracy = 0.280000
k = 8, accuracy = 0.262000
k = 8, accuracy = 0.282000
k = 8, accuracy = 0.273000
k = 8, accuracy = 0.290000
k = 8, accuracy = 0.273000
k = 10, accuracy = 0.265000
k = 10, accuracy = 0.296000
k = 10, accuracy = 0.276000
k = 10, accuracy = 0.284000
k = 10, accuracy = 0.280000
```

```
{1: 0.2656, 3: 0.2496, 5: 0.2732, 8: 0.27599999999999997, 10: 0.2802, 12: 0.2794, 15: 0.275, 20: 0.279, 50: 0.2744000000000003, 100: 0.2616}
```

Nearest Neighbor Classifier

N개의 example이 있을 때 , 시간 복잡도

Train	O(1)	// 한번에 저장
Predict	O(N)	// 매번 N번 비교

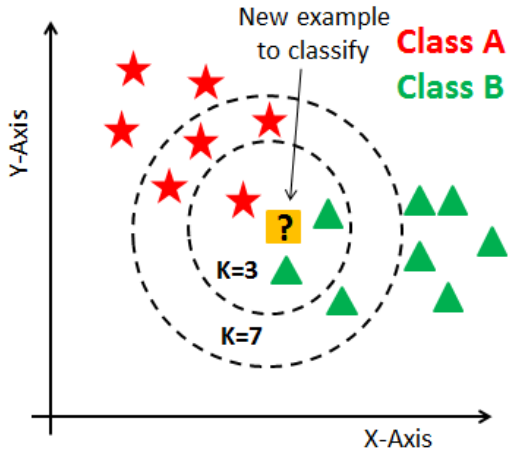
Nearest Neighbor Classifier

Train	FAST	SLOW
Predict	SLOW	FAST

모바일, 웹 low power device에서 사용 하기 위해서는 빠른 성능이 필요

K-Nearest Neighbors

kNN : 새로운 데이터가 주어졌을 때 기존 데이터 중 가장 가까운 k개 이웃의 정보를 기반으로 새로운 데이터를 예측하는 방법



K = 1, 즉 단 1개의 이웃만 보고 예측하는 것이 Nearest Neighbor 방식

https://ichi.pro/ko/k-nearest-neighbors-knn-157240814281359

K-Nearest Neighbors

kNN = Lazy Model

: 모델을 별도로 구축하지 않고 관측치만을 이용

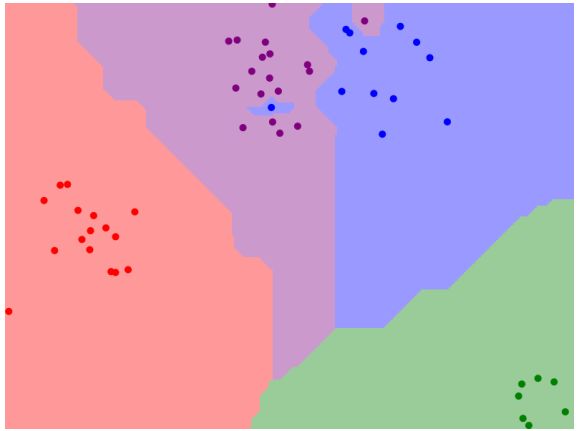
- **kNN의 Hyperparameter**

1. 탐색할 이웃의 수(k)
2. 거리 측정 방법

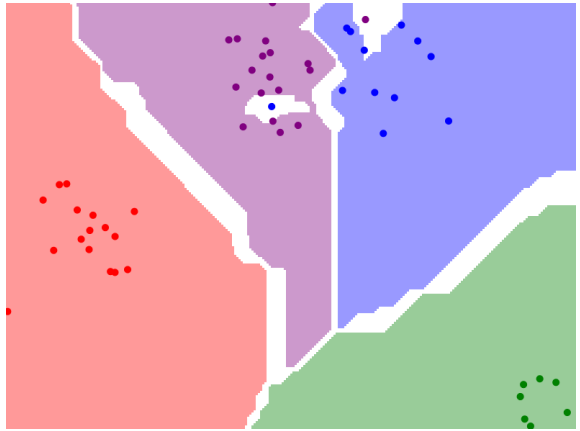
→ k가 너무 작을 경우 overfitting, 너무 클 경우 underfitting

K-Nearest Neighbors

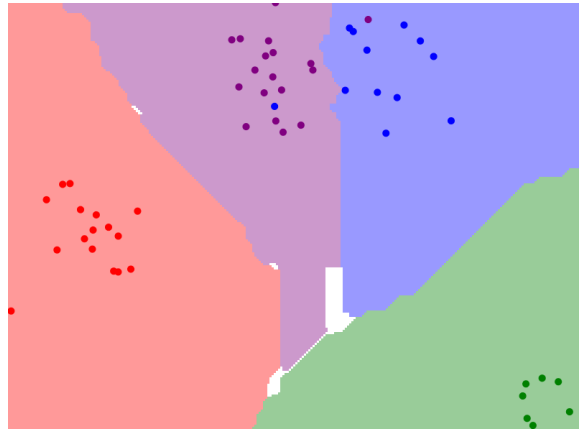
K = 1



K = 2



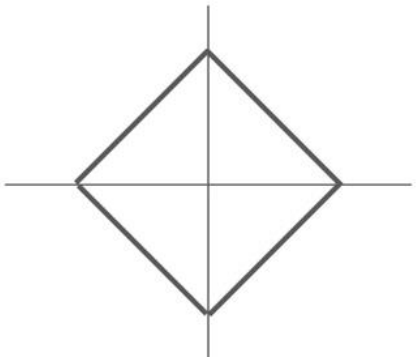
K = 5



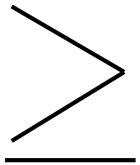
Distance Metric

L1 (Manhattan) distance

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

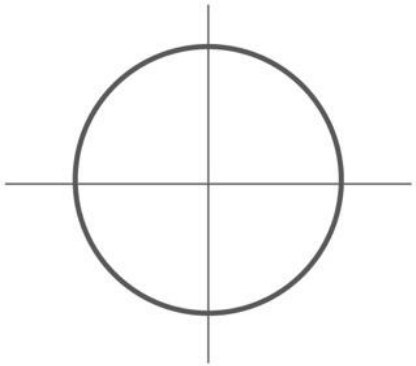


좌표계에 따라 달라짐



L2 (Euclidean) distance

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



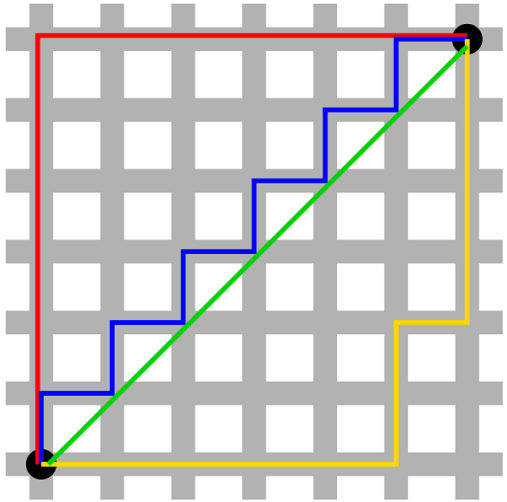
두 점 사이의 직선 거리
좌표계 영향 X

Distance Metric

Why Distance?

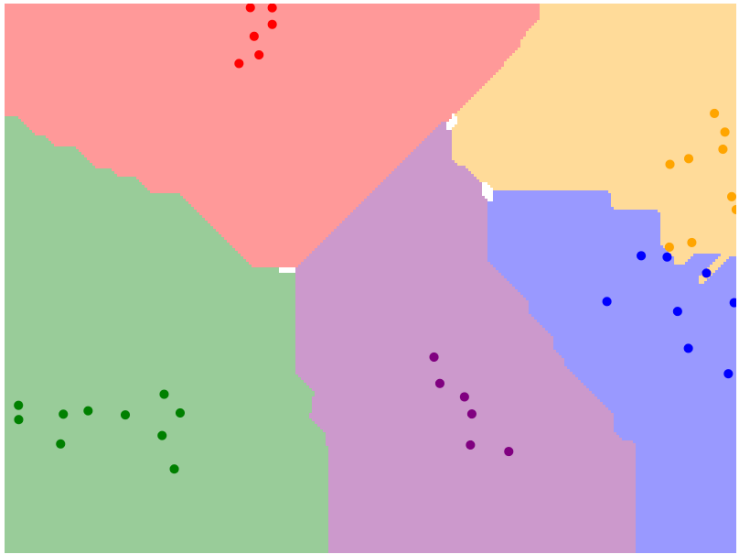
→ 거리는 일종의 유사도(Similarity)와 같은 개념

→ 거리가 가까울수록 특성(feature)들이 비슷하다는 뜻

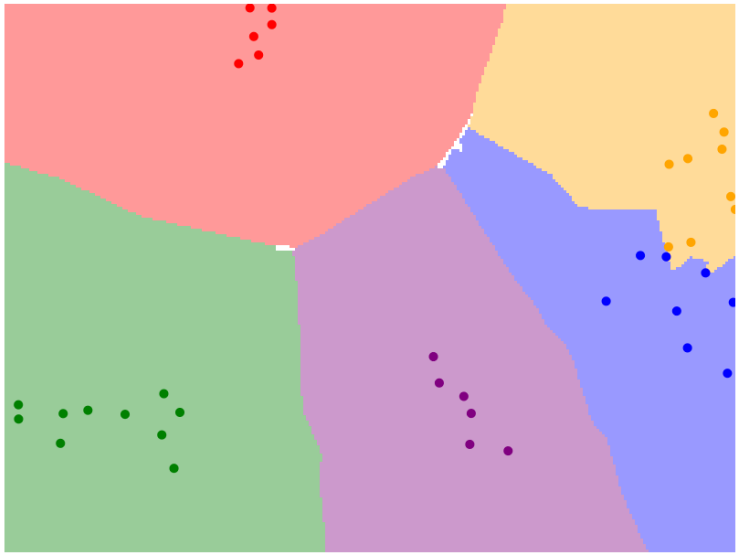


Distance Metric

L1 Metric



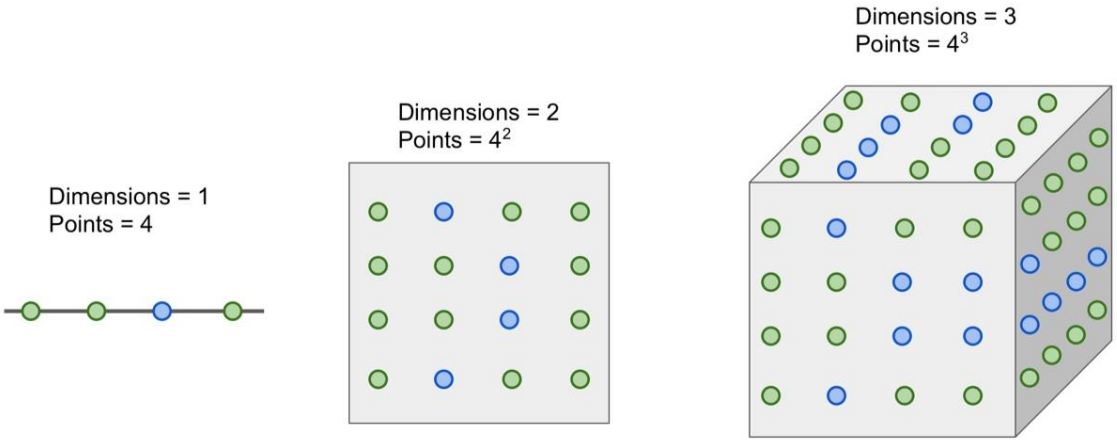
L2 Metric



K-Nearest Neighbors

kNN은 왜 이미지에 사용되지 않을까?

- Test 시간이 오래 소요됨
- Curse of Dimensionality



Exponential Growth

kNN이 잘 작동하려면 data set이 공간 전체를 덮어야 함

Dimension이 높아질수록 데이터 많이 필요

Data Augmentation

Augmentation의 필요성

➔ Training set과 현실의 test set 사이의 괴리감 존재

➔ 임의의 잡음이나 translation을 training set에 가해서 괴리감을 줄이고 성능 향상

➔ 데이터 크기가 작은 경우 데이터셋을 늘리기 위해 사용

Data Augmentation

Flip

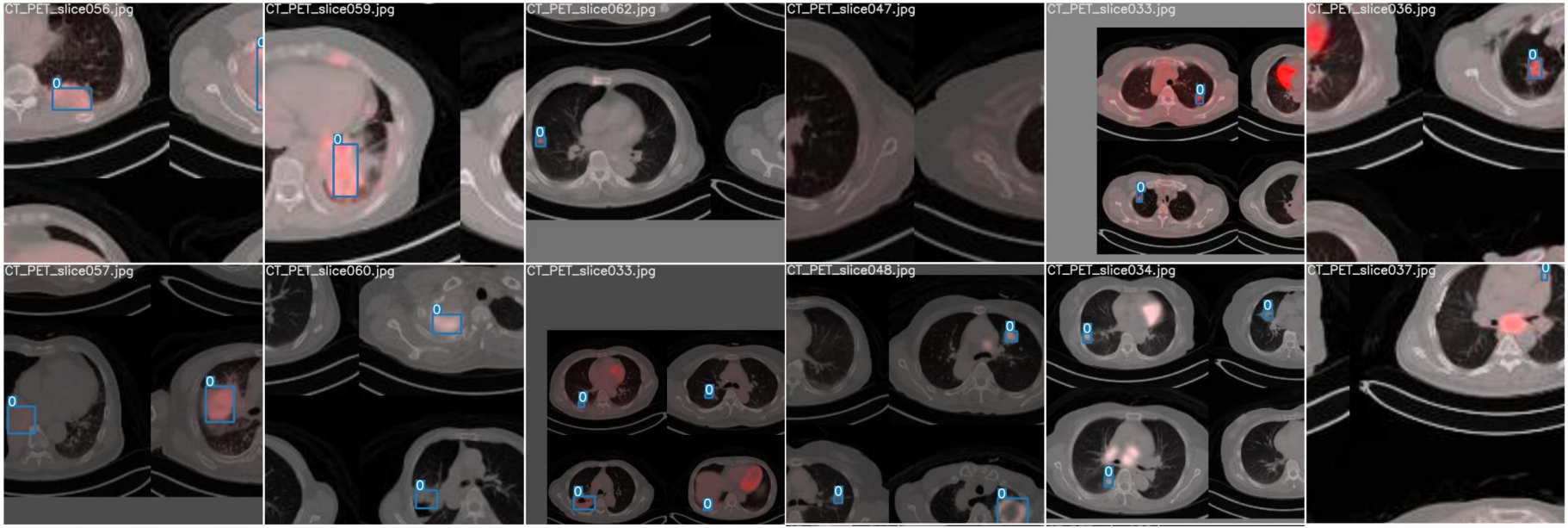
Noise

Contrast

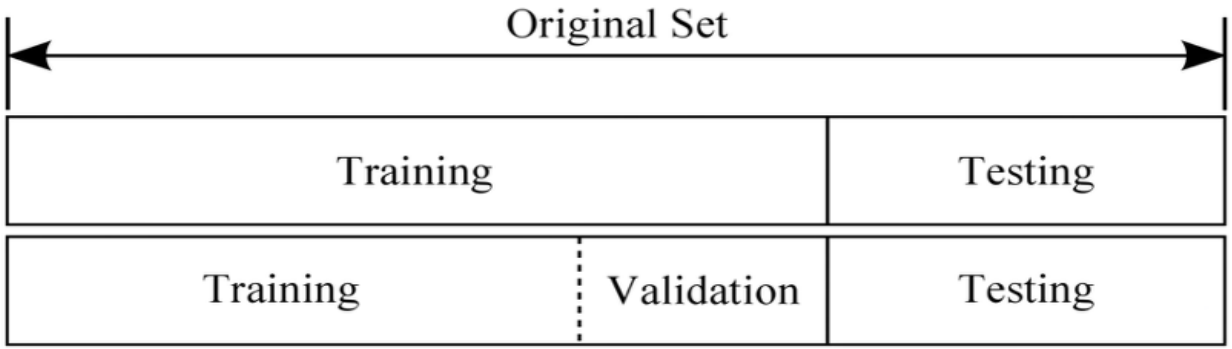
Combination

https://imgaug.readthedocs.io/en/latest/index.html

Data Augmentation



Data Set



Training set: 모델을 학습시키는 data

Validation set: 최적화를 위해 Hyperparameter Tuning에 사용

Test set: 모델의 최종 성능 평가를 위해 사용

Data Set

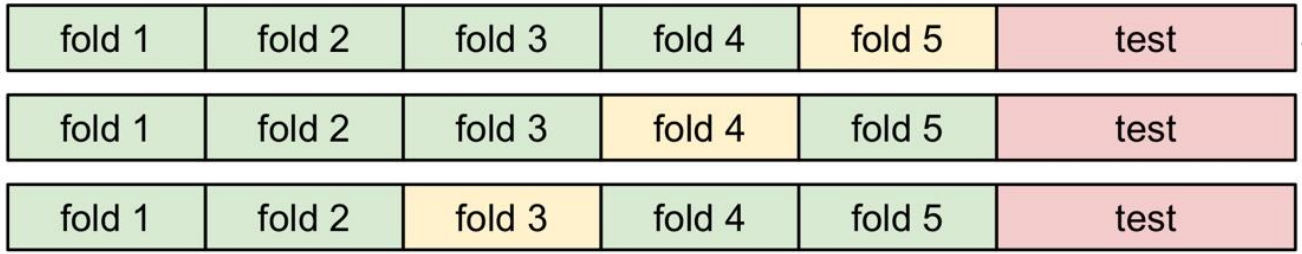
Cross-Validation

- 모든 데이터를 활용함

→ 특정 데이터에 overfitting 되는 것을 방지

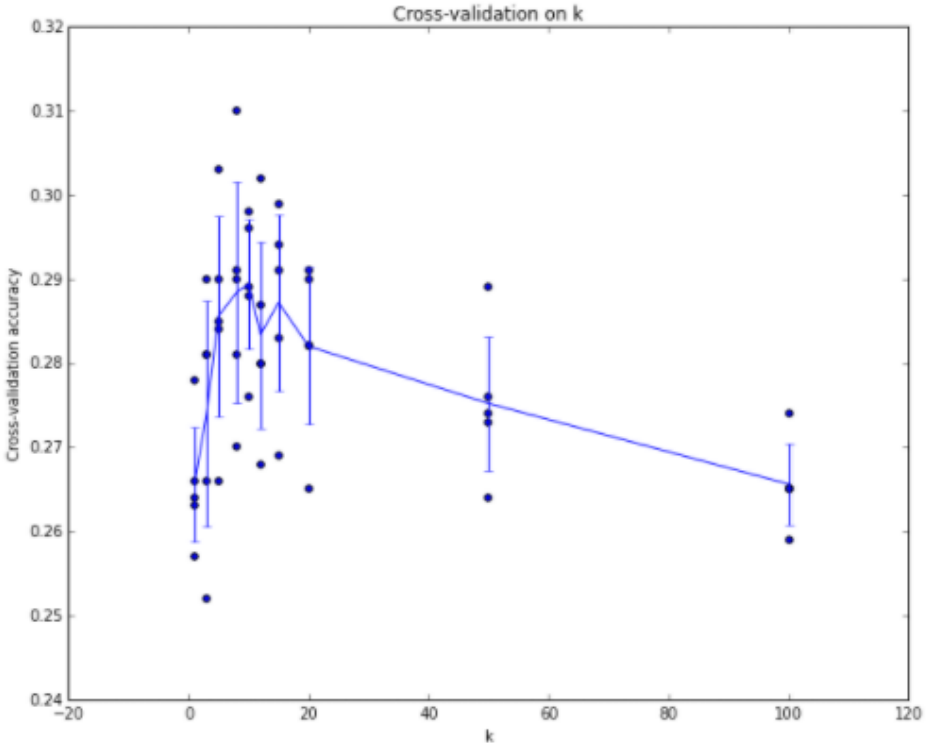
→ 정확도 향상(작은 dataset에 활용)

- 시간 오래 걸림



K-fold
Cross-Validation

Data Set



K-fold Validation

- K 개의 그룹으로 데이터를 나누고 총 K번 검증하고 평균을 냄
- 모델의 성능을 전체적으로 볼 수 있음

Hyper parameters

Hyper parameters: 모델에서 외적인 요소로 데이터 분석을 통해 얻어지는 값이 아니라 사용자가 직접 정하는 값

- 모델의 parameter 값을 측정하기 위해 알고리즘 구현 과정에서 사용
- 경험에 의해 정해지는 경우가 많아 여러 번 수행해보며 최적의 값을 찾음
- Ex) Learning Rate, Error Function, Batch Size

Parameters

Parameters: 모델 내부에서 확인이 가능한 변수로 데이터를 통해 산출 가능한 값

- 학습할 때 모델에 의해 요구되는 값들
- 모델의 능력을 결정하며 주로 데이터로부터 학습되고 사용자가 정하지 않음
- 알고리즘의 최적화 과정에서 정해짐
- Ex) Neural Network의 Weight, SVM의 support vector

Parametric Model

- 데이터가 특정 분포를 따른다 가정
- 결정해야하는 파라미터의 종류와 수가 결정되어 있음
- 우선 모델의 형태를 결정하고 모델의 파라미터를 학습을 통해 발전시키는 방식
- Training data에 대해 우리가 가지고 있는 정보를 요약하여 W에 저장
- Test 할 때 실제 training data를 사용할 필요 없음 (⟷ kNN)

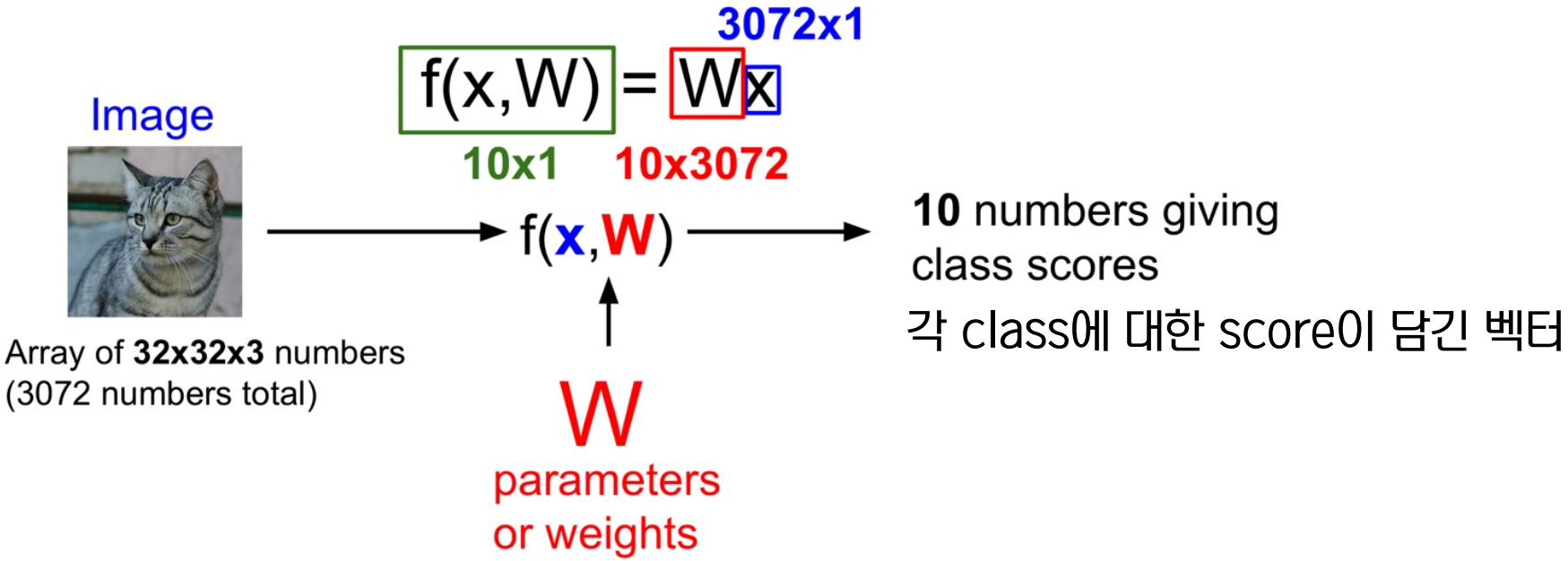
https://process-mining.tistory.com/131

Non – Parametric Model

- 데이터가 특정 분포를 따른다는 가정 없음 (더 flexible)
- 학습에 따라 튜닝할 파라미터가 명확하게 정해져 있지 않음
- Data에 대한 사전지식이 전혀 없을 때 유용
- 더 큰 데이터를 필요로 하고 학습에 시간이 오래 걸림

Linear Classifier

이미지의 pixel values 들을 하나의 긴 column vector로 만든 것

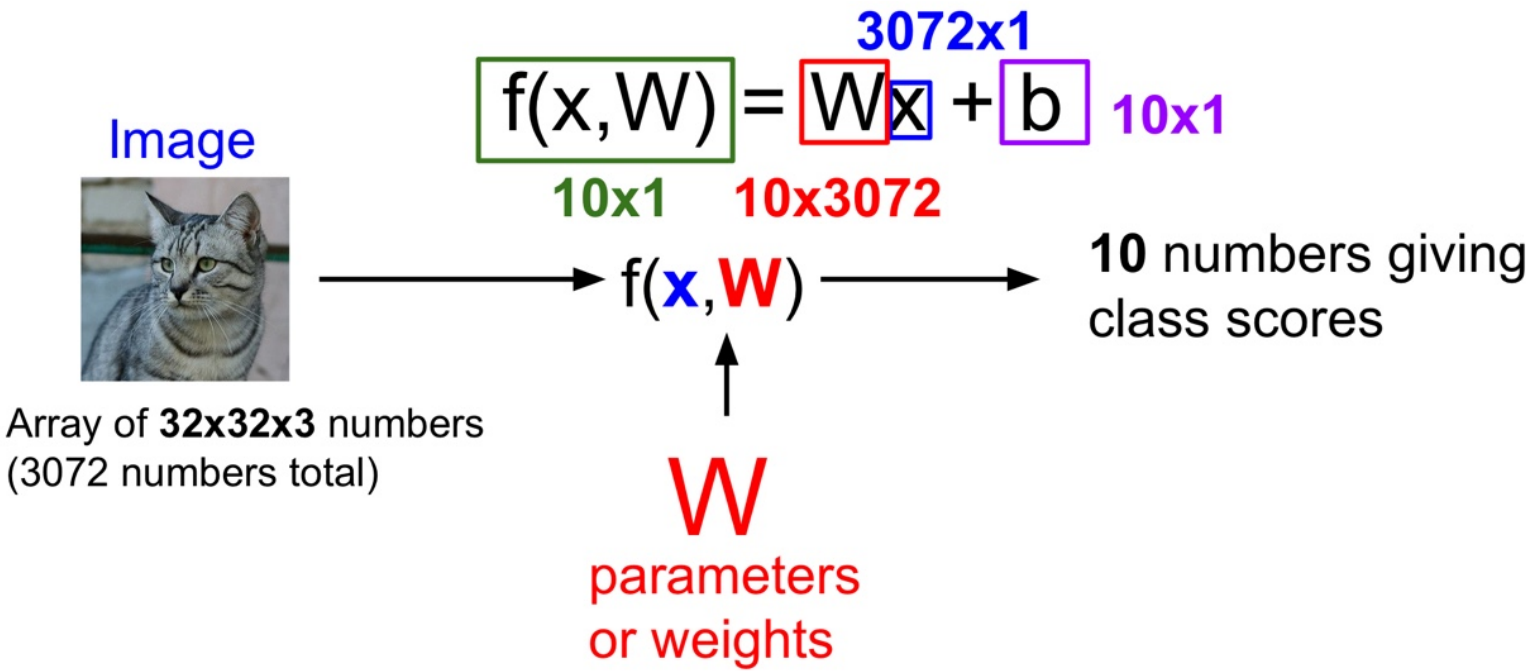


각 class에 대한 score이 담긴 벡터

Training data에 대한 정보가 요약된 벡터

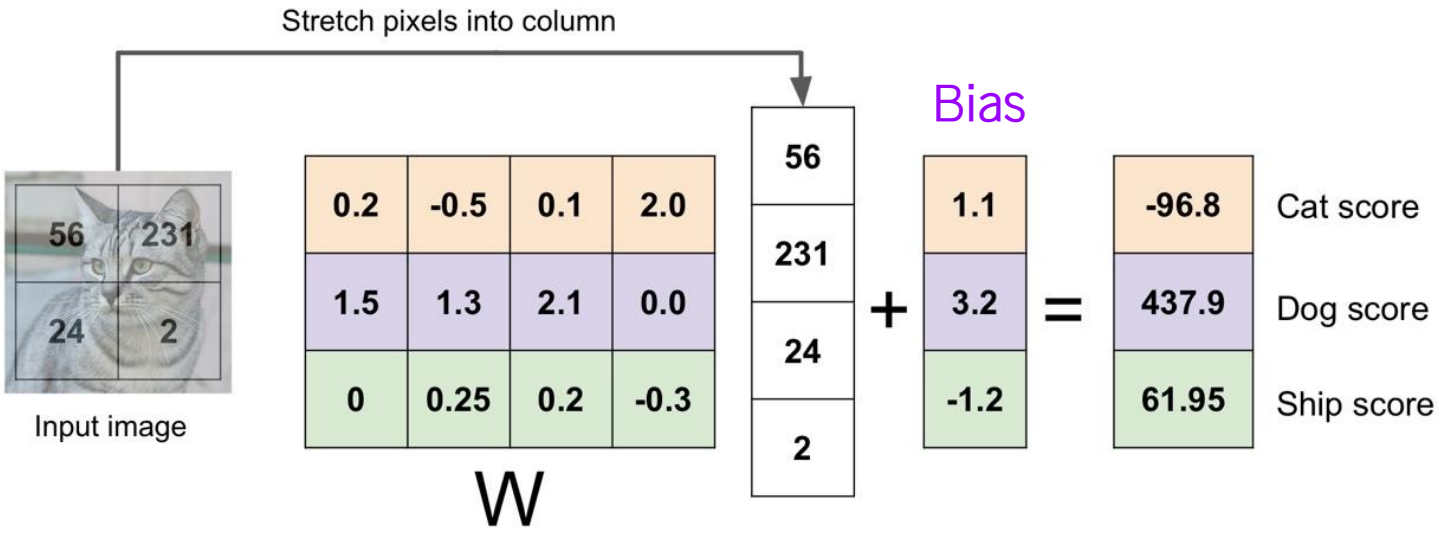
Linear Classifier

Bias: Training data와 연관이 없고 어떤 class에 대한 선호도를 데이터와 독립적으로 제공



Linear Classifier

Ex) 4개의 픽셀, 3개의 classes 갖는 이미지를 가정

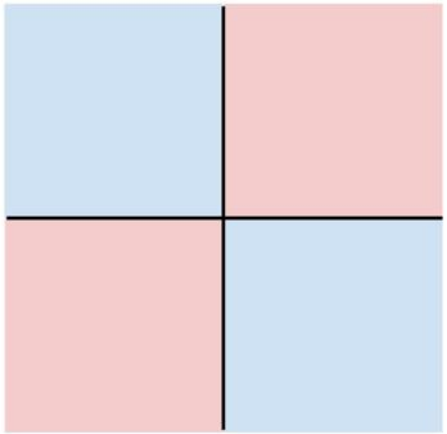


Linear Classifier

Hard Cases

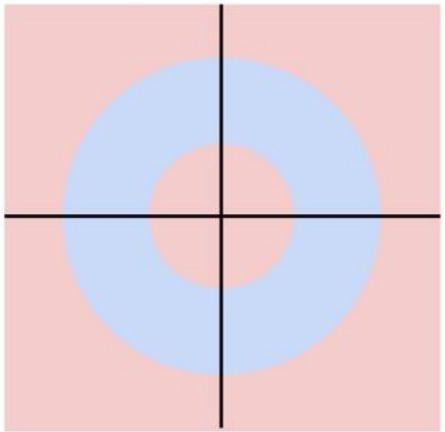
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



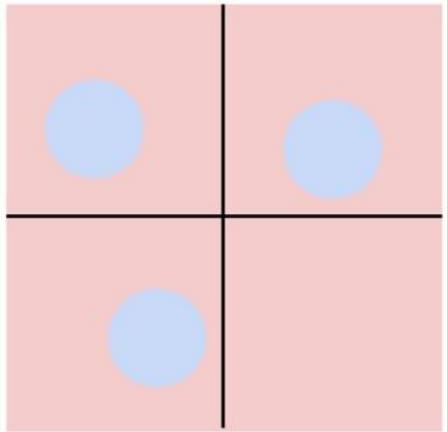
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



감사합니다

Q&A