Image Classfification pipeline

Week2 민소연,안서연

Index

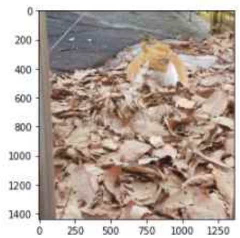
Image classification

01 Semantic Gap

Input image

낙엽위에 고양이가 앉아 있다


```
plt.imshow(image_color)
plt.show()
```



이미지 픽셀을 0~255 사이의 숫자로 구성된 격자로 인식

```
image_color
```

```
array([[[156, 151, 145],
        [158, 153, 147],
        [162, 155, 147],
        ...,
        [189, 172,  94],
        [178, 161,  83],
        [179, 162,  82]],

       [[157, 152, 146],
        [159, 154, 148],
        [163, 156, 148],
        ...,
        [198, 181, 103],
        [201, 184, 104],
        [197, 180,  98]],

       [[157, 152, 146],
        [159, 154, 148],
        [162, 155, 147],
        ...,
        [212, 195, 113],
        [212, 195, 113],
        [205, 189, 104]],

       ...,

       [[ 95,  82,  74],
        [ 92,  79,  71],
        [ 93,  80,  72],
        ...,
        [117,  98,  83],
        [115,  97,  83],
        [113,  95,  81]],

       [[ 92,  79,  71],
        [ 94,  81,  73],
        [ 96,  83,  75],
        ...,
        [117,  98,  83],
        [114,  96,  82],
        [111,  93,  79]],

       [[ 90,  77,  69],
        [ 94,  81,  73],
        [ 98,  85,  77],
        ...,
        [114,  95,  80],
        [113,  95,  81],
        [111,  93,  79]]], dtype=uint8)
```

```
image_color.shape
```

```
(1439, 1362, 3)
```

1439x1362x3

(각 원소는 R,G,B 의 3개 숫자로 구성)

?

EWHA EURON

02 Challenges

Viewpoint variation (관점 변화)

Illumnination (밝기)

Deformation(변형)

알고리즘이 변화에 robust(이상치/에러값에 둔감) 해야함

02 Challenges

Occlusion(폐색)

This image is CC0 1.0 public domain

Background Clutter

This image is CC0 1.0 public domain

Intraclass variation

This image is CC0 1.0 public domain

알고리즘이 변화에 robust(이상치/에러값에 둔감) 해야함

02 Challenges

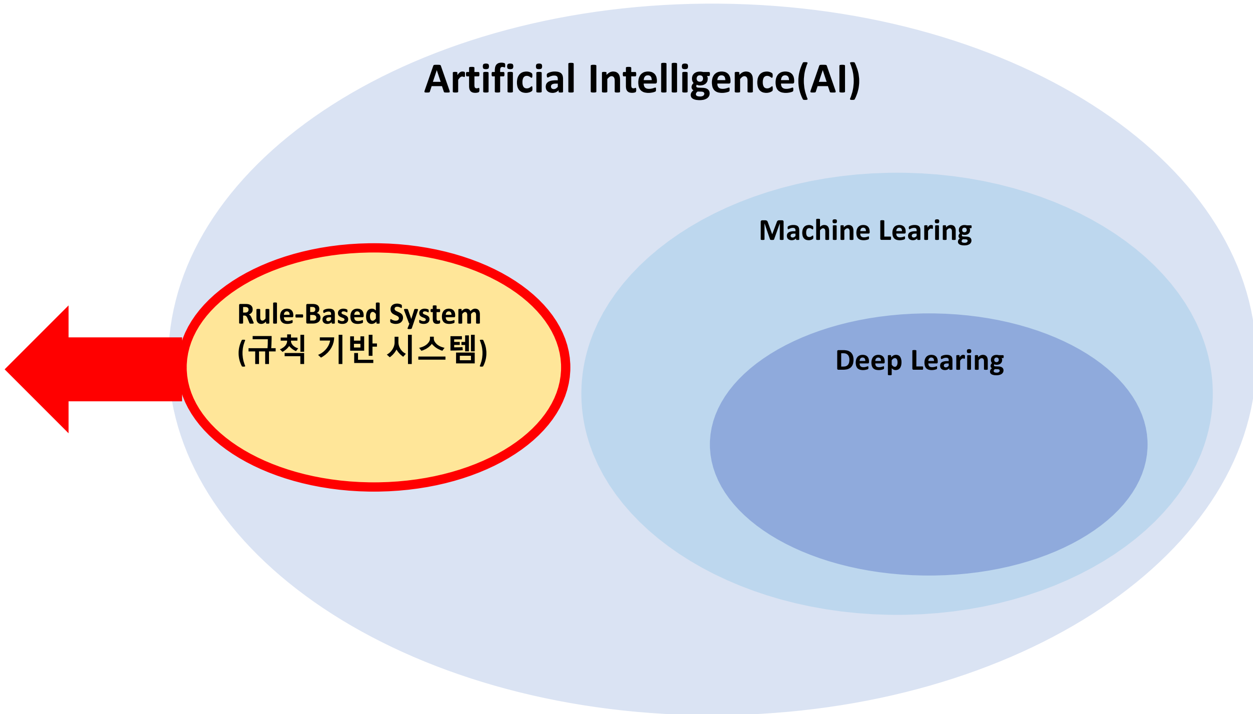
규칙 기반 방법으로 분류하기

Input image

```
def classify_cat(image):
  cat_score=0
  if image.color in ["orange","yellow","brown","black"]:
     cat_score+=1;
  if image.shape == "triangle":
      cat_score+=1;
  if image.size< MAX:
      cat_score+=1;
        ...
  return class_label
```

Hard-coding 기반 rules (1 function)

Output label



모든 라벨에 대해
규칙을 만들어야 하는 어려움

Difficult...

Data-Driven Approach

01 Data-Driven Approach

(Image – Label) Dataset 수집

모든 라벨이 있는 이미지를 수집

Train

머신 러닝 활용 classifier에 데이터 모두 학습, 기억

Evaluate

새로운 이미지로 classifier 성능 평가

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

2 function

- 안정성 및 확장성 (다른 객체에도 적용 가능)
- 사람이 학습하는 것과 비슷

02 Dataset

오픈소스 CV dataset

- COVID-19 X-Ray Dataset(V7) (폐 x-ray)
  (https://www.v7labs.com/open-datasets/covid-19- chest-x-ray-dataset)
- CIFAR-10 & CIFAR-100 (32x32 사물 사진)
  (https://www.cs.toronto.edu/~kriz/cifar.html)
- ImageNet (대규모 이미지 데이터셋)
- (https://image-net.org/)
- Kinetics-700(사람 행동과 관련된 대규모 동영상 데이터)
  (https://deepmind.com/research/open-source/kinetics)
- MNIST (28x28 손글씨 사진)
  (http://yann.lecun.com/exdb/mnist/)
- LSUN (large-scale 장면)
  (https://www.yf.io/p/lsun)
- IMDB-Wiki(얼굴 인식)
  (https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/)
- MS COCO (대규모 객체탐지 /세그멘테이션/ Key-point 탐지/captioning 데이터셋)
  (https://cocodataset.org/#download)
- Labeled Faces in the Wild(얼굴 인식)
  (http://vis-www.cs.umass.edu/lfw/)

02 Dataset

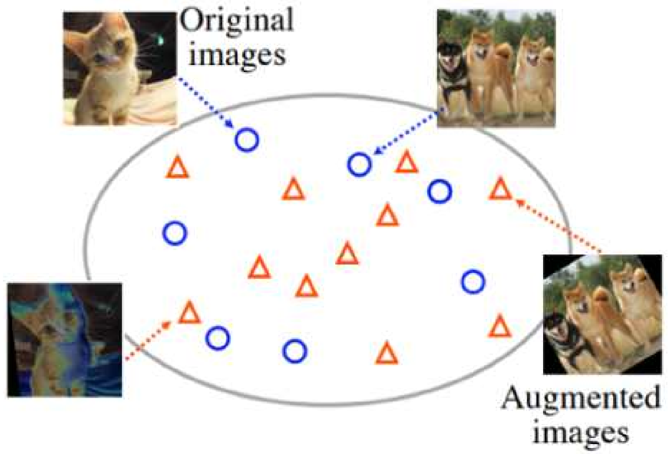
오픈소스 CV dataset

- Cityscapes (city scene)
  (https://www.cityscapes-dataset.com/)
- LabelMe-12-50k (객체 탐지/인식)
  (http://www.ais.uni-bonn.de/download/datasets.html)
- Places2(365-Standard) (scene 인식)
  (http://places2.csail.mit.edu/download.html)
- VisualGenome(이미지 구조를 언어로 연결)
  (https://visualgenome.org/api/v0/api_home.html)
- Stanford Dogs (강아지)
  (http://vision.stanford.edu/aditya86/ImageNetDogs/main.html)
- Stanford Cars (자동차)
  (http://ai.stanford.edu/~jkrause/cars/car_dataset.html)
- CelebFaces (유명인 얼굴 인식)
  (https://www.kaggle.com/jessicali9530/celeba-dataset)
- Face Mask Detection (마스크 인식)
  (https://www.kaggle.com/andrewmvd/face-mask-detection)
- Fire and Smoke Dataset(Fire,smoke)
  (https://www.kaggle.com/dataclusterlabs/fire-and-smoke-dataset)

03 Data Augmentation

Data Augmentation이 필요한 이유

- **데이터셋을 늘려** overfitting **문제 해결**
- **원본 이미지에 각종 변환**(translation)**을 적용하여 개수 증강**



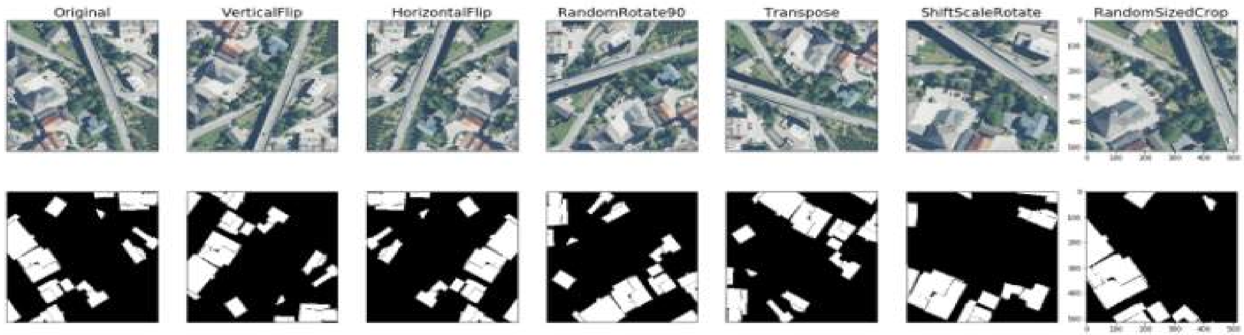
그림과 같이 비어있는 training data point**를 채움**

03 Data Augmentation 기법

Image Transformation 을 통해 데이터를 증강

1. Pixel-Level Transforms
   - Blur,Jitter,Noise

2. Spatial-Level Transforms
   - Flip,Rotation



04 Nearest Neighbor Classifier(NN)

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

1. 모든 데이터-라벨을 기억

2. 가장 비슷한 라벨(Nearest Neighbor)을 예측

04 Nearest Neighbor Classifier(NN)

```python
def train(images, labels):
  # Machine learning!
  return model
```

1. 모든 데이터-라벨을 기억

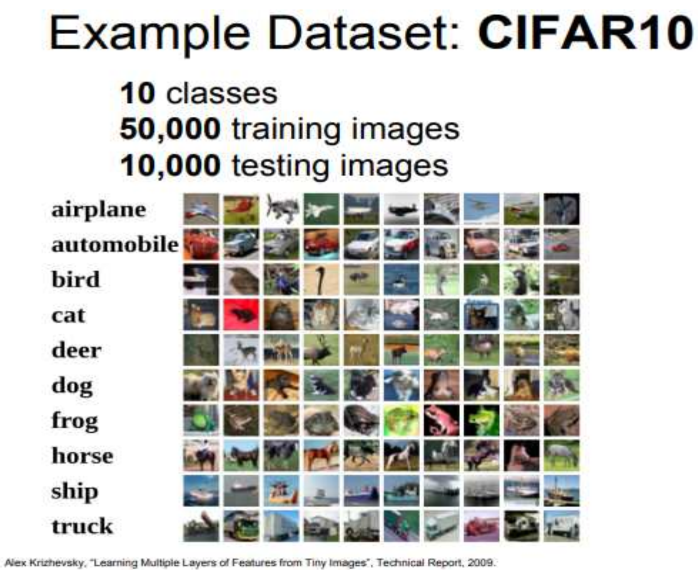
Example Dataset: **CIFAR10**

10 classes
50,000 training images
10,000 testing images

Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

04 Nearest Neighbor Classifier(NN)

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```



2. 가장 비슷한 라벨(Nearest Neighbor)을 예측

- Distance Metric에서 L1 사용

04 Nearest Neighbors(NN)

Distance Metric to compare images

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

$$\left| \overset{\text{test image}}{\begin{array}{|c|c|c|c|} \hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array}} - \overset{\text{training image}}{\begin{array}{|c|c|c|c|} \hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{array}{|c|c|c|c|} \hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline \end{array}} \xrightarrow{\text{add}} 456$$

04 Nearest Neighbors(NN)

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

짧고 간결함, but..

N example이 있을때 시간복잡도

Train: O(1) < Predict O(N)

```
Two loop version took 41.686833 seconds
One loop version took 40.690964 seconds
No loop version took 0.597274 seconds
```

복습과제와 같이 predict 함수에서 L1을 구현하는데 루프가 생겨 복잡해 질 때 소요 시간이 크게 증가하는 것을 알 수 있음

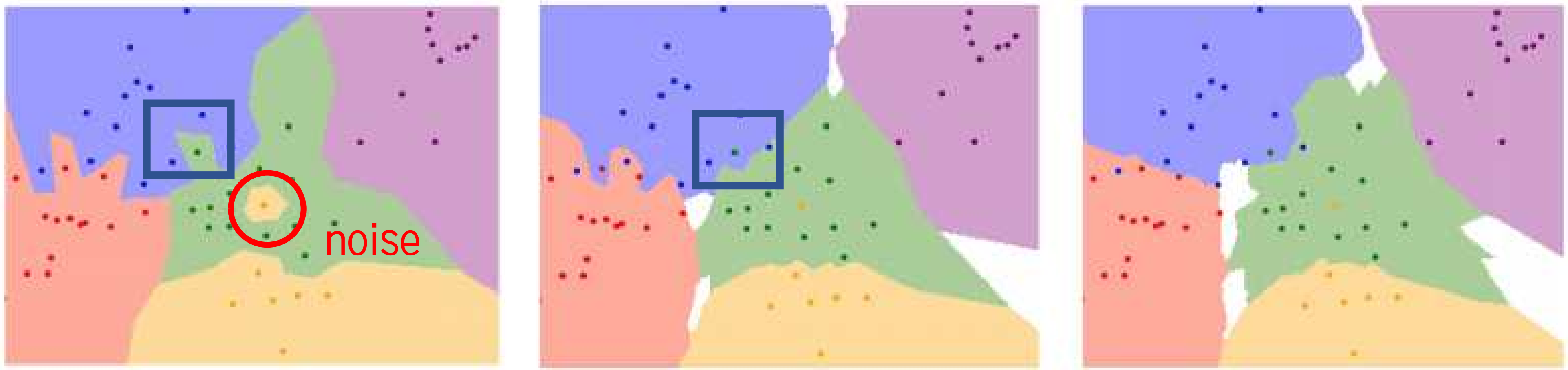
05 K-Nearest Neighbors(KNN)

K-Nears Neighbors: 새로운 데이터의 라벨을 예측할때, 가장 비슷한 기존 데이터 k개의 라벨을 majority vote(다수결)하여 라벨을 예측하는 방법

K=5

05 K-Nearest Neighbors(KNN)

K-Nears Neighbors: 새로운 데이터의 라벨을 예측할때, 가장 비슷한 기존 데이터 k개의 라벨을 majority vote(다수결)하여 라벨을 예측하는 방법



K=1

K=3

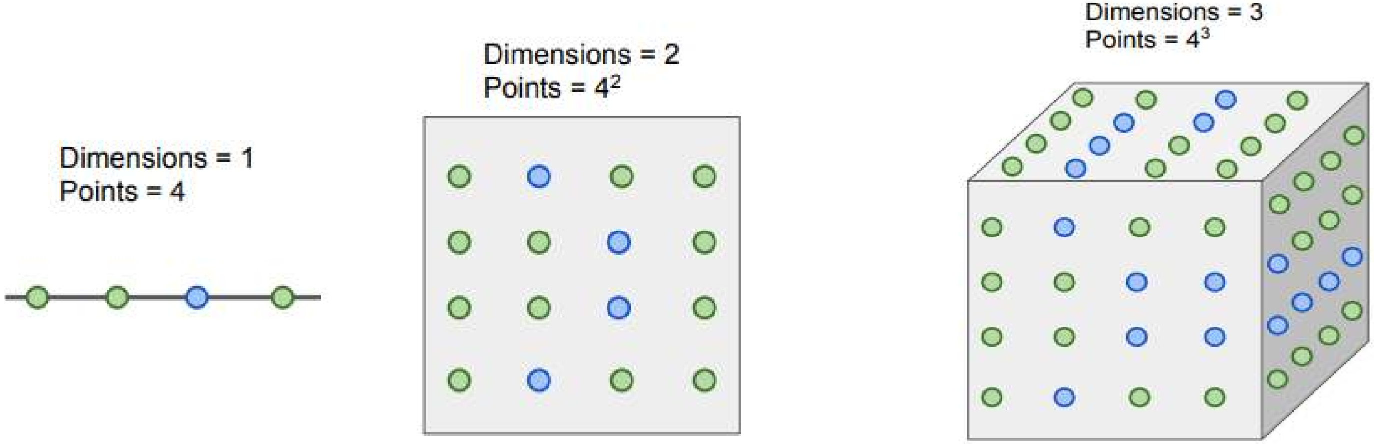
K=5

Smooth boundary

05 K-Nearest Neighbors(KNN)

K-Nears Neighbors 이 이미지에 사용되지 않는 이유

1. Test 시간이 오래 소요
2. 픽셀에 적용되는 Distance Metrics(L1,L2)가 유용하지 않음
   (서로 다른 이미지가 같은 거리를 가질 수 있음)
3. Curse of dimensionality
   - K-NN이 잘 동작하려면 공간을 충분히 덮는 points(데이터)가 필요한데, 차원이 증가할수록 필요한 데이터가 기하급수적으로 증가함



EWHA EURON

Parameter Approach

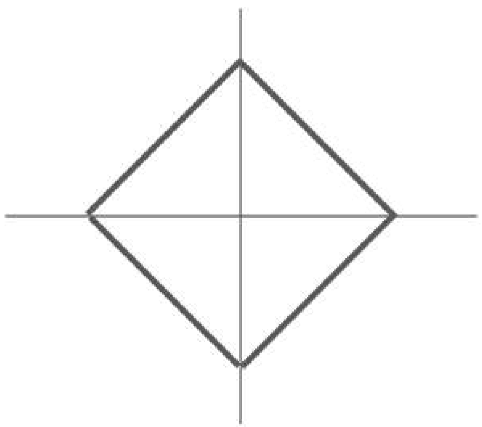
EWHA
EURON

01 Distance Metric (L1, L2)

#1 About L1 distance, L2 distance

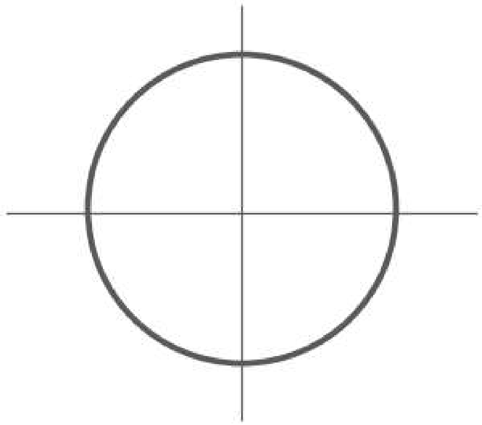
L1 (Manhattan) distance

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



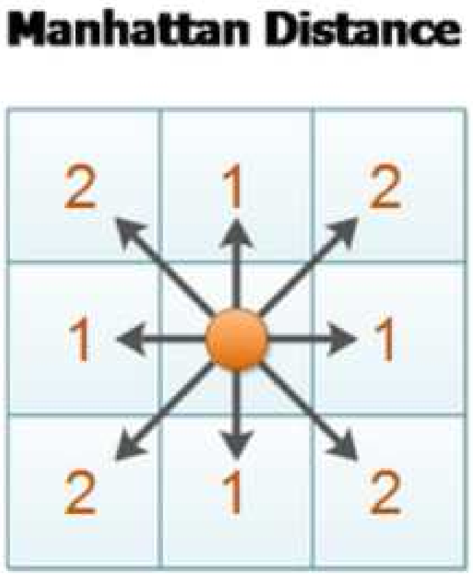
L2 (Euclidean) distance

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



01 Distance Metric (L1, L2)

#1 L1 (Manhattan) distance



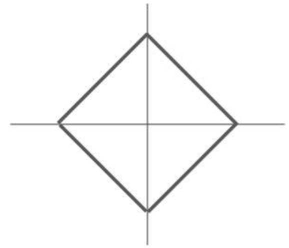
$$|x_1 - x_2| + |y_1 - y_2|$$

-> Stepped path 형태

- Outlier 대응에 강하다
- Domain에 따라 변화율이 일정하다
- Regularization, Regression에 쓰인다.
- Perceptual loss를 위한 VGG network에서 Feature간의 거리 차이를 계산하는데 쓰인다.

L1 (Manhattan) distance

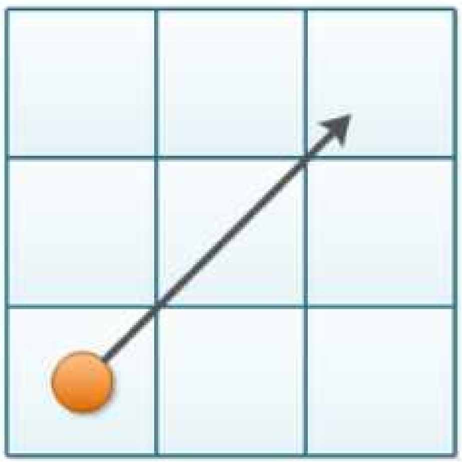
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



01 Distance Metric (L1, L2)

#1 L2 (Euclidean) distance

Euclidean Distance



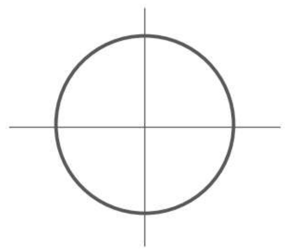
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

-> Straight line 형태

- L1에 비해 Outlier 대응에 강하지 않다
- x가 변할 때 그 기울기가 정답에 가까울 수록 완만 해진다
- Regression에서 사용되는 mean square error에 쓰인다

L2 (Euclidean) distance

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



01 Distance Metric (L1, L2)

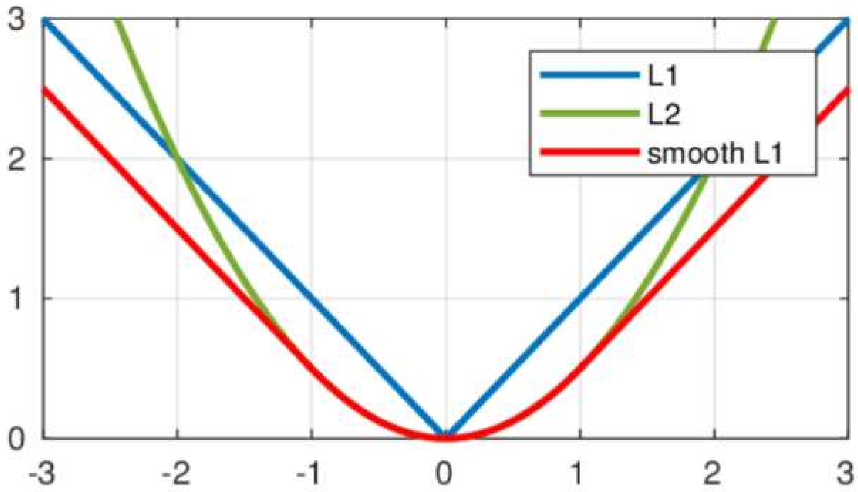
#1 L1 distance VS L2 distance

특정 벡터가 개별적인 의미를 가지고 있다면
-> L1 distance

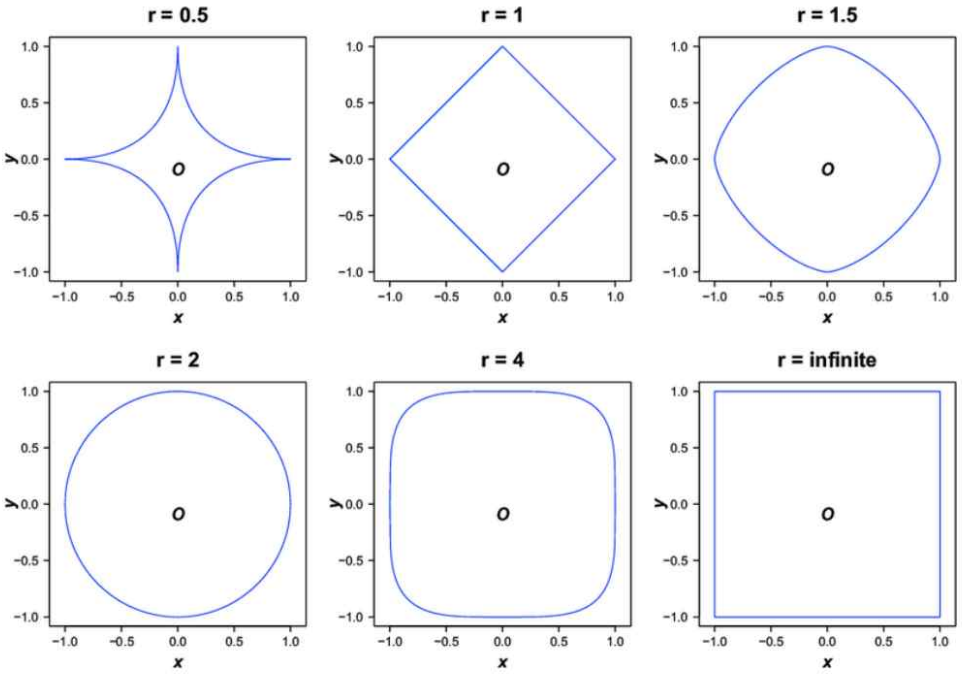
일반적인 벡터 요소들의 의미를 모르거나 의미가 없다면
-> L2 distance

01 Distance Metric (L1, L2)

#1 Others



$$D(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^{1/p} \right)^{p}$$



L1은 outlier에 강하고 L2는 정답에 가까워질수록 변화율이 완만하다는 장점이 있기에 이 둘의 특성을 융합하여 Smooth L1 loss 방식이 있다.

L1 (Manhattan) distance와 L2 (Euclidean) distance을 일반화 한 형태로 Lp라고 표현한다.

01 Distance Metric (L1, L2)

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

$\xrightarrow{\text{add}}$ 456

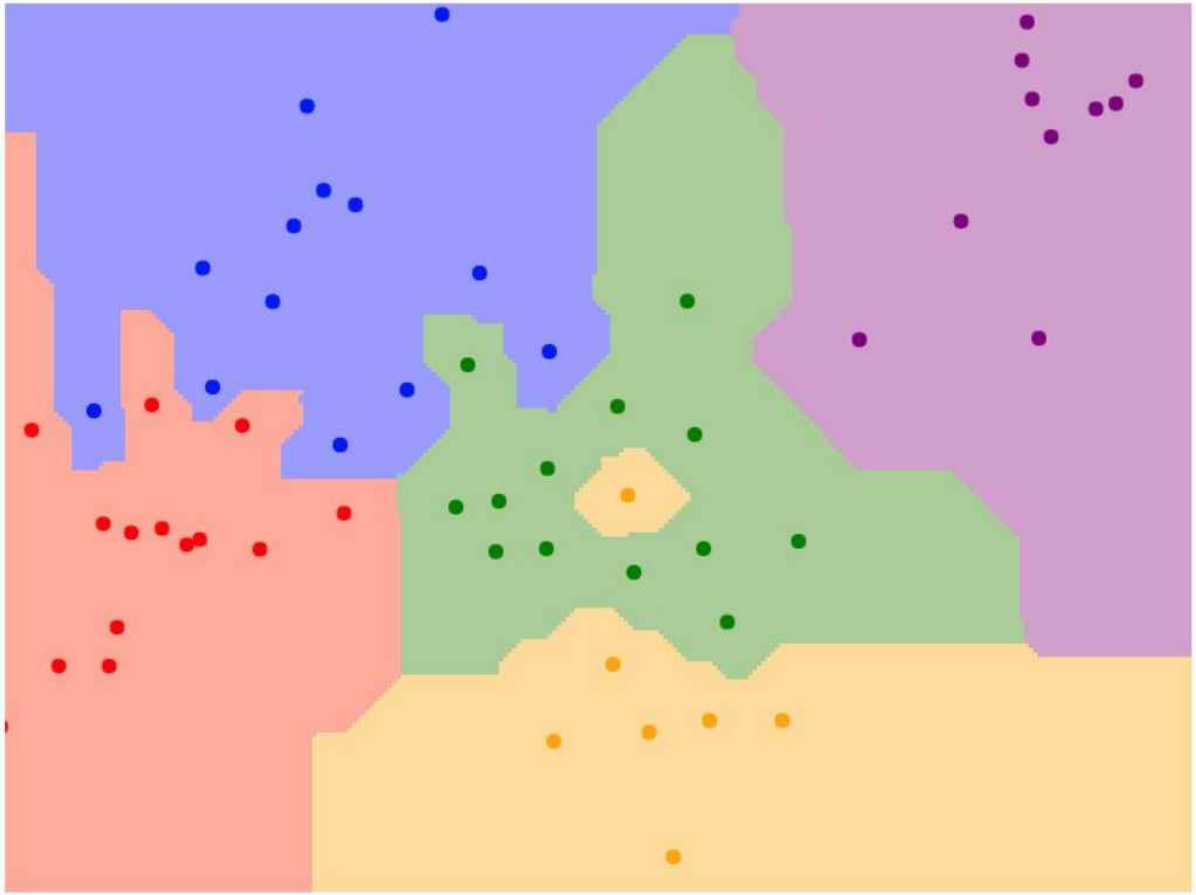
Pixelated test Image와 training image의 차이를
구해서 그 모든 차들의 합을 구하는 형태로 사용된다.

차이가 일정 기준보다 크면 두 이미지는 다른 객체에 대한 이미지인 것으로,
차이가 0에 가까울 수록 비슷한 객체에 대한 이미지인 것으로 한다.

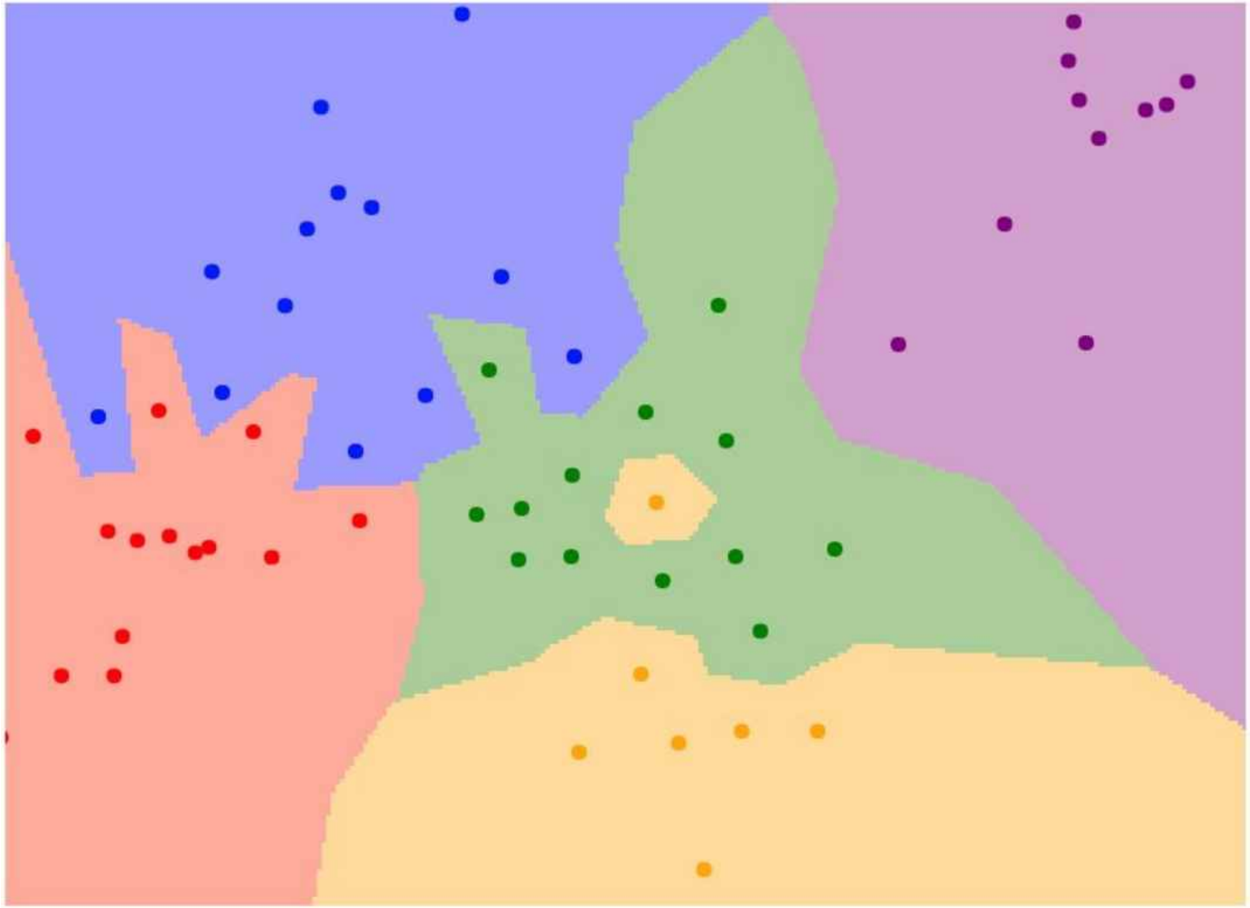
01 Distance Metric (L1, L2)

L1 (Manhattan) distance

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



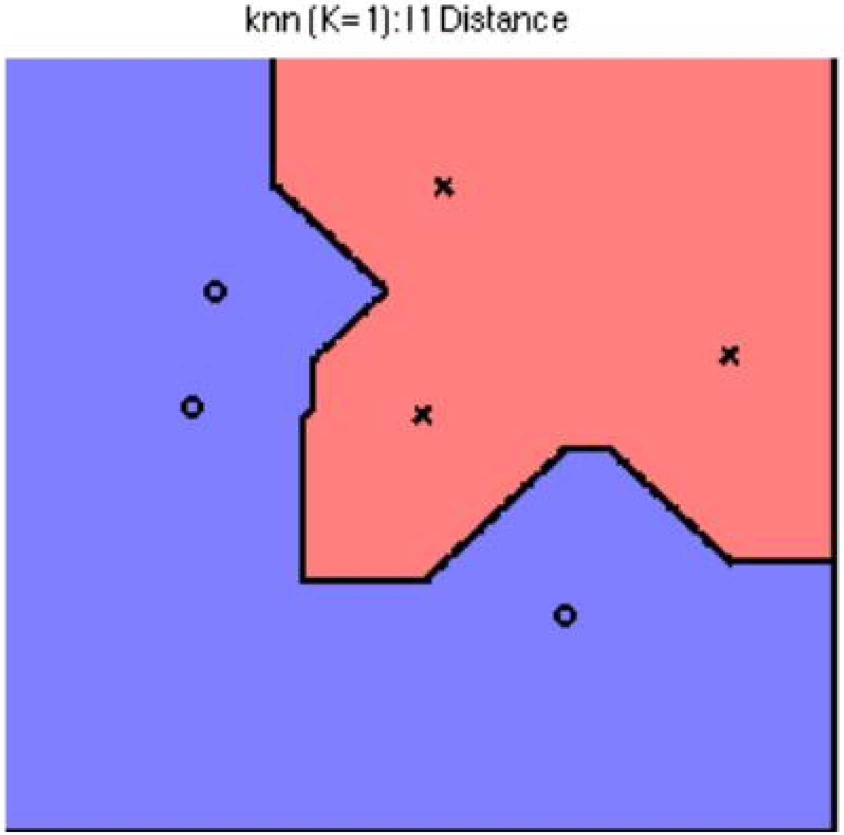
01 Distance Metric (L1, L2)

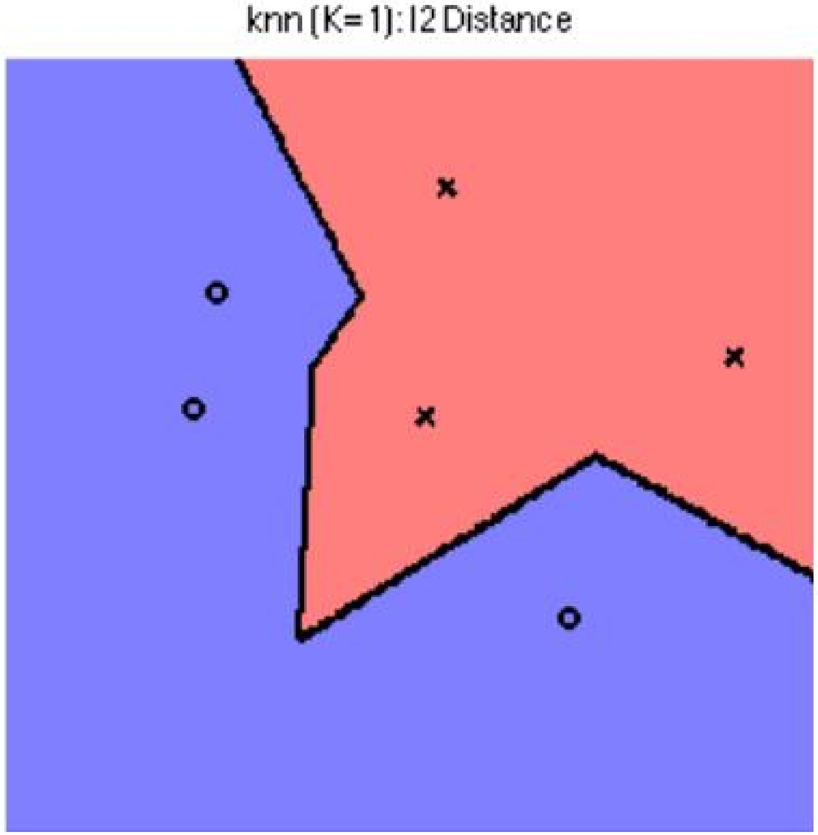


L2 (Euclidean) distance

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

01 Distance Metric (L1, L2)





02 Hyperparameter

#01 What is Hyperparameter and What's the best?

What is the best value of **k** to use?
What is the best **distance** to use?

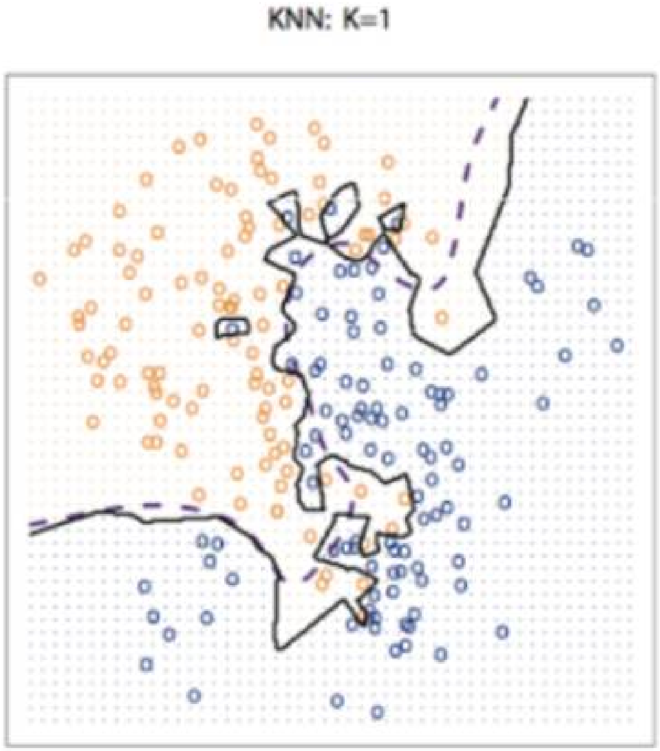
These are **hyperparameters**: choices about the algorithm that we set rather than learn

Very problem-dependent.
Must try them all out and see what works best.

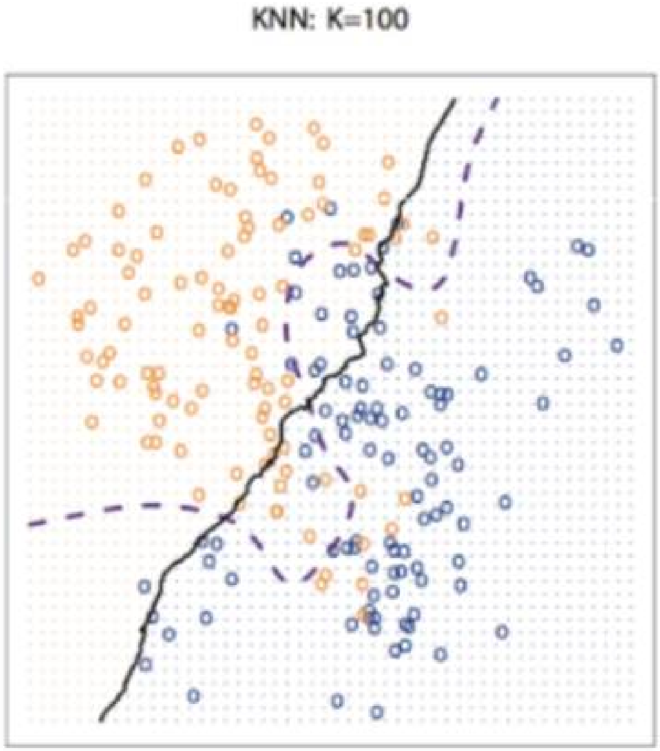
EWHA EURON

02 Hyperparameter

#01 What is Hyperparameter and What's the best?



오버피팅 (Overfitting)



언더피팅 (Underfitting)

02 Hyperparameter

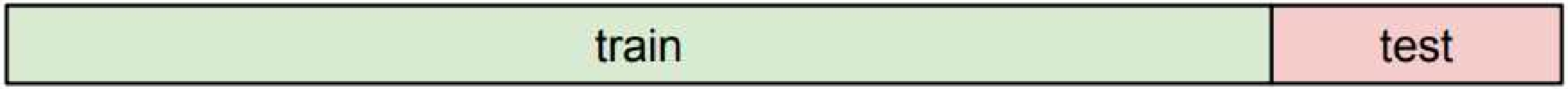
Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

- What we want is more of a good prediction than a good classification.

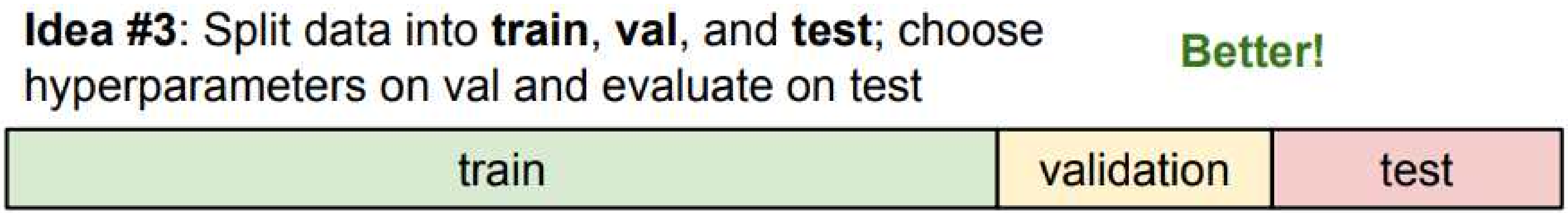
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data



- Even if the results performed as a test set are good, performance may deteriorate when new data is received.

02 Hyperparameter



- The prediction accuracy of completely new data can be measured by creating a validation set, training with a train set, modifying the parameter, and then checking the performance with a test set at the end.

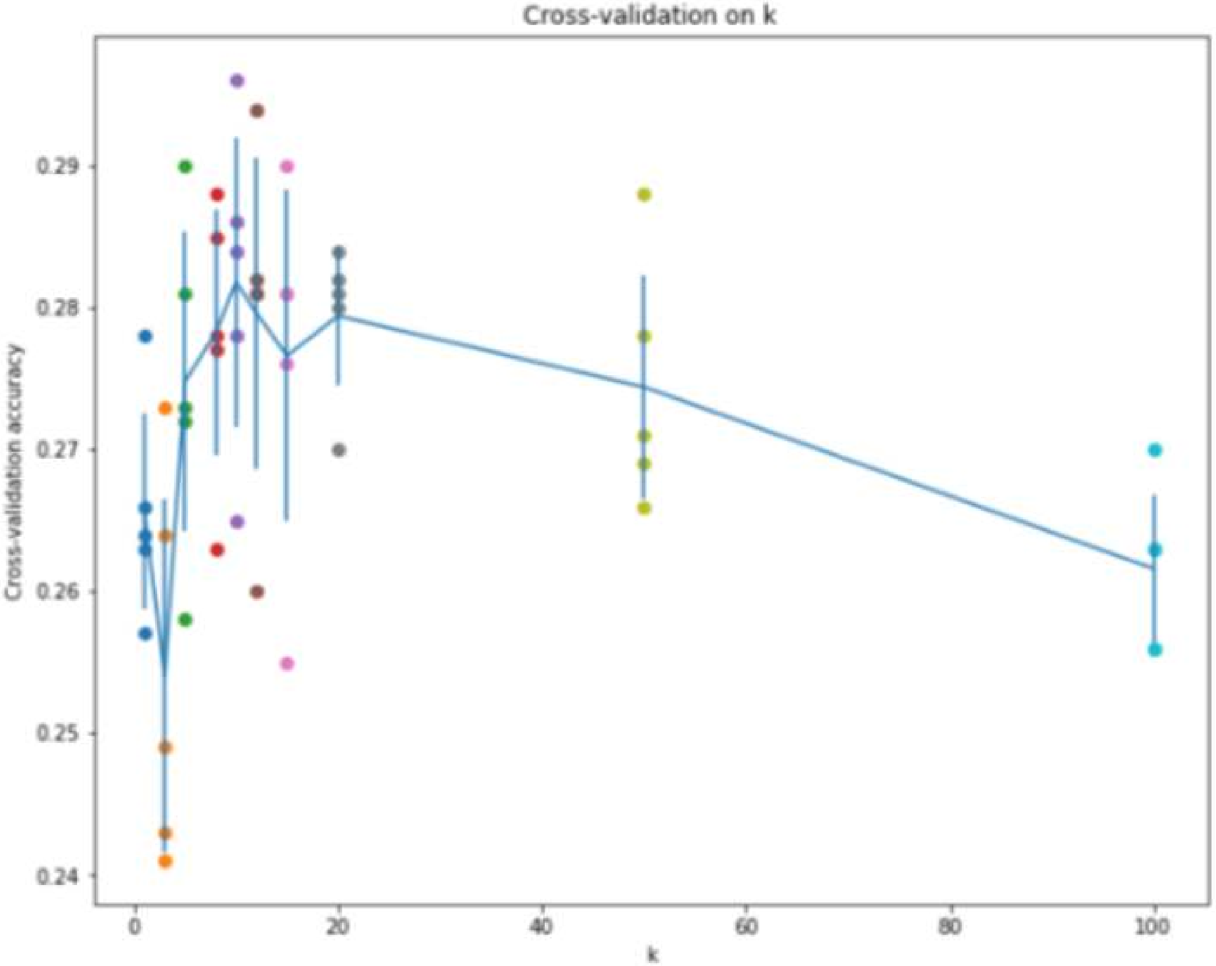
02 Hyperparameter

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

02 Hyperparameter



연산량이 너무 많아서
딥러닝에서는 자주 쓰이지는 않으나
정확한 K를 찾기에 유용한 방식

03 KNN Summary

- In Image classification,
  we start with a "training set" of images and labels,
  and must predict labels on the "test set"

- The kNN classifier predicts labels based on nearest training examples

- Distance metric and K are hyperparameters

- Choose hyperparameters using the validation set
  -> only run on the test set once at the very end!

04 Parametric Approach

#1 Basic Idea of parametric Approach

There is a set of “Fixed Parameters” that uses to determine a probability model that is used in Machine Learning.

Parametric methods are those methods for which we priory knows that the population is normal, or if not then we can easily approximate it using a normal distribution which is possible by invoking the Central Limit Theorem.

Parameters for using the Normal Distribution

- Mean
- Standard Deviation

04 Parametric Approach

#1 Basic Idea of parametric Approach

Eventually, the classification of a method to be parametric is completely depends on the presumptions that are made about a population.

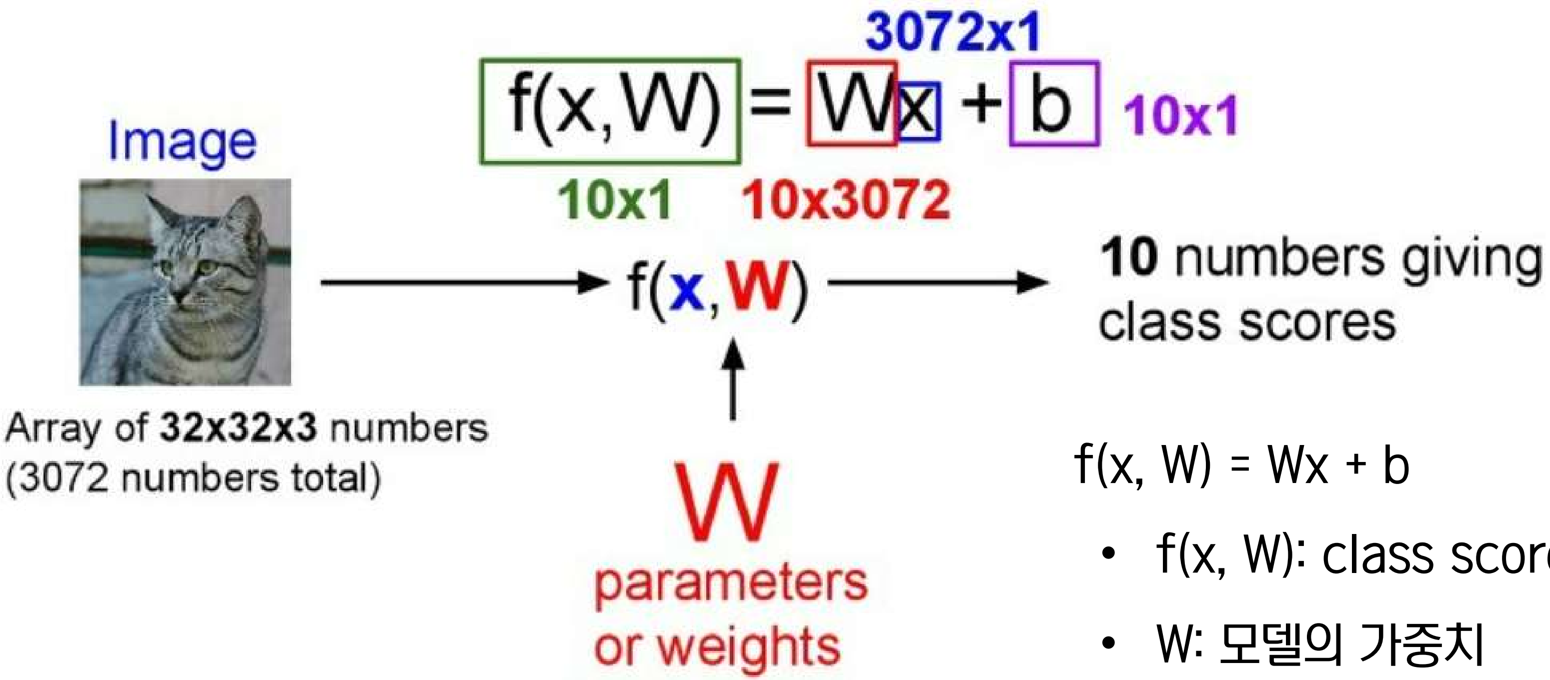
There are many parametric methods available some of them are:

- Confidence interval used for – population mean along with known standard deviation
- The confidence interval is used for – population means along with the unknown standard deviation.
- The confidence interval for population variance.
- The confidence interval for the difference of two means, with unknown standard deviation.

Linear Classification

EWHA EURON

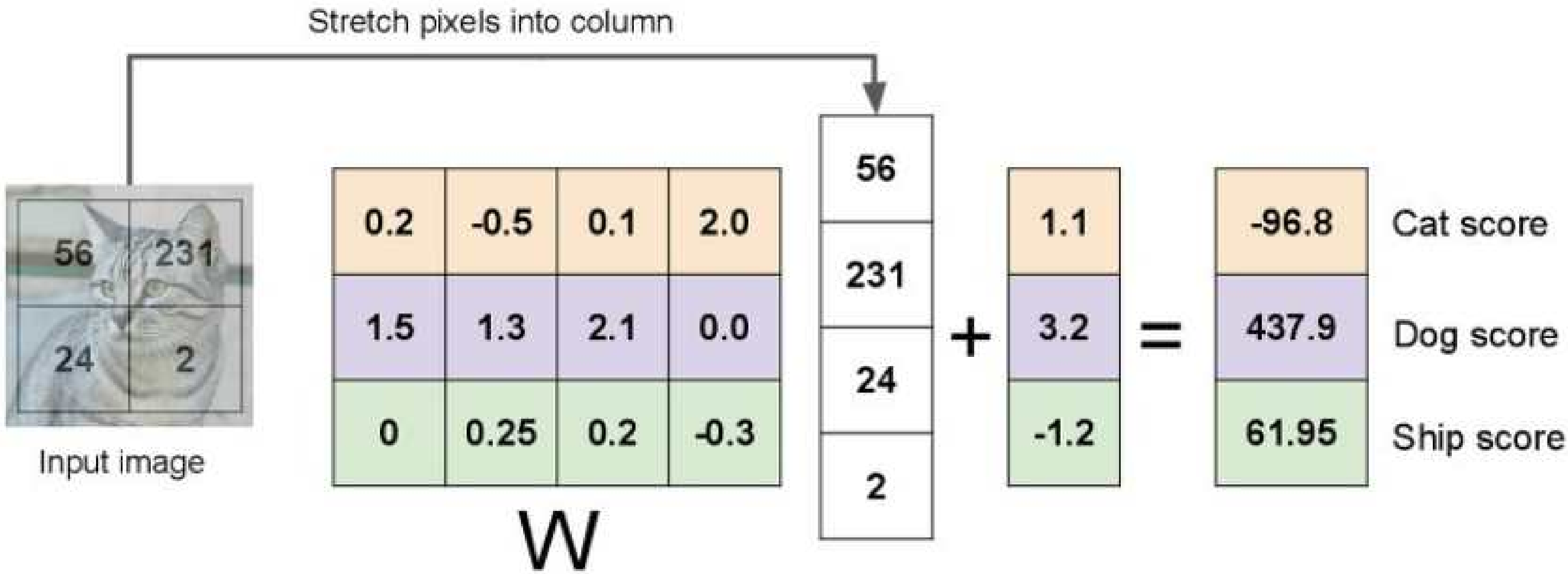
01 Linear Classification



f(x, W) = Wx + b

- f(x, W): class score를 반환
- W: 모델의 가중치
- x: data
- b: bias (학습과는 무관한 데이터의 일부 클래스에 대한 선호도)

01 Linear Classification

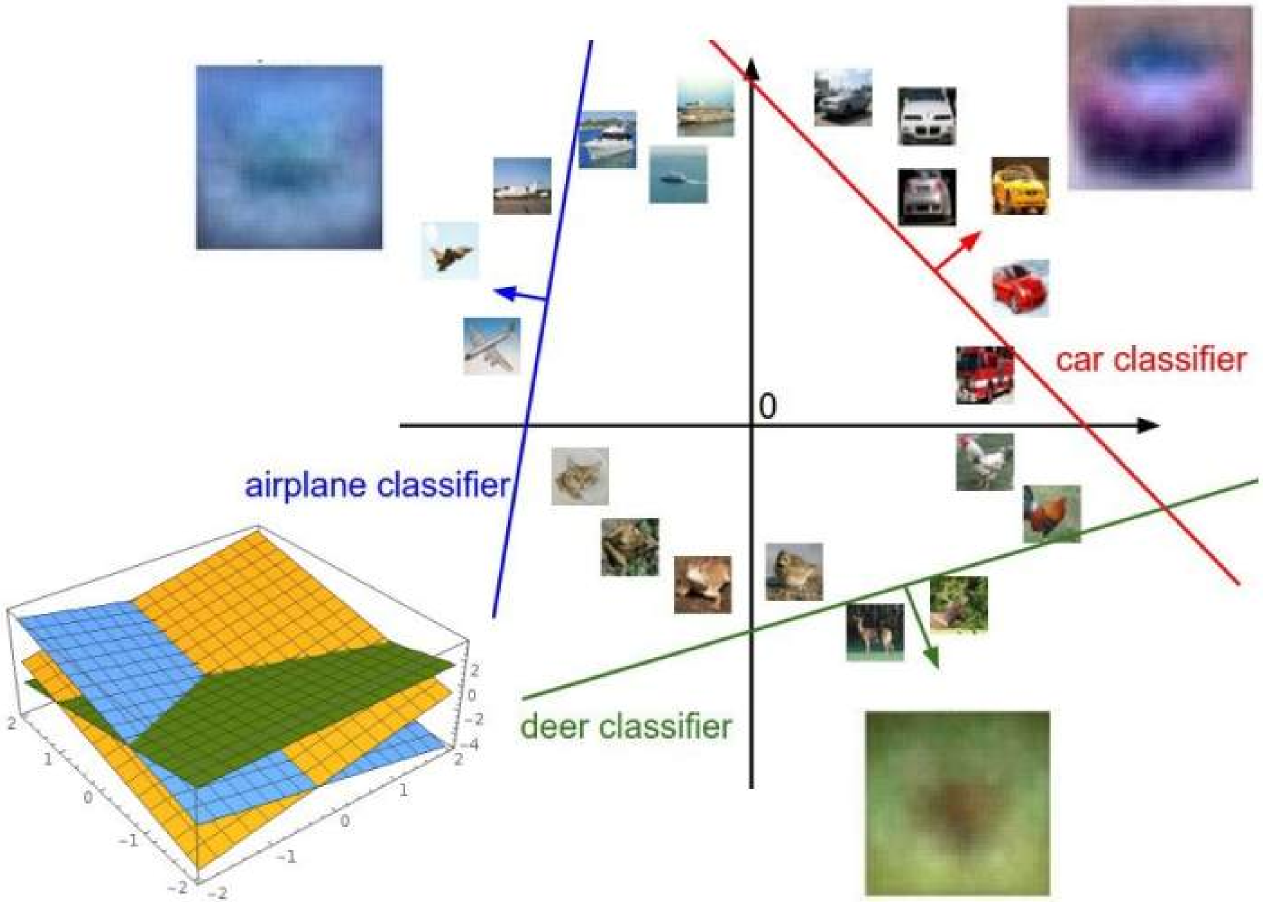


01 Linear Classification

$$f(x,W) = Wx + b$$

Example trained weights of a linear classifier trained on CIFAR-10

01 Linear Classification



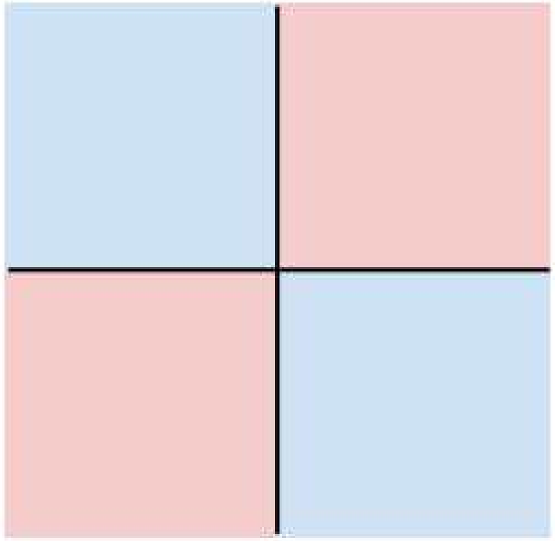
f(x, W) = Wx + b

- f(x, W): class score를 반환
- W: 모델의 가중치
- x: data
- b: bias

02 Hard Cases for a Linear Classification

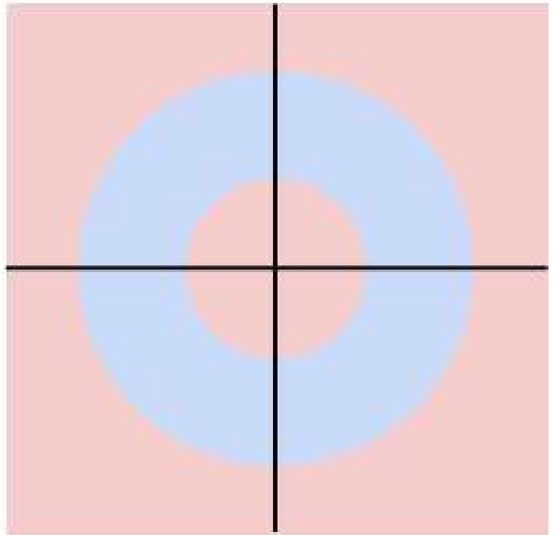
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



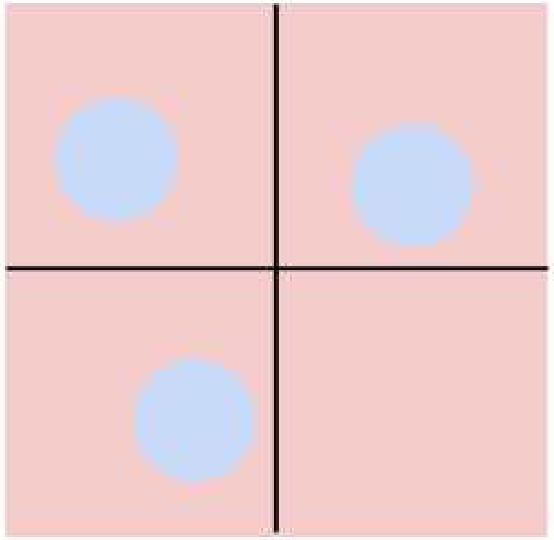
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



03 Linear Classification Summary

So far: Defined a (linear) <u>score function</u> $f(x,W) = Wx + b$

Example class scores for 3 images for some W:

How can we tell whether this W is good or bad?

airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

고양이는 어느 정도는 했지만 정답을 맞추지는 못 했고,

자동차는 임의의 W값을 설정한 상태에서 정답을 맞췄고,

개구리는 완전히 답을 틀렸다.

-> W값을 어떻게 설정해야 정답률(=예측 정확도)를 향상 시킬 수 있을까?

03 Linear Classification Summary

$$f(x,W) = Wx + b$$

Coming up:
- Loss function (quantifying what it means to have a "good" W)
- Optimization (start with random W and find a W that minimizes the loss)
- ConvNets! (tweak the functional form of f)

THANK YOU