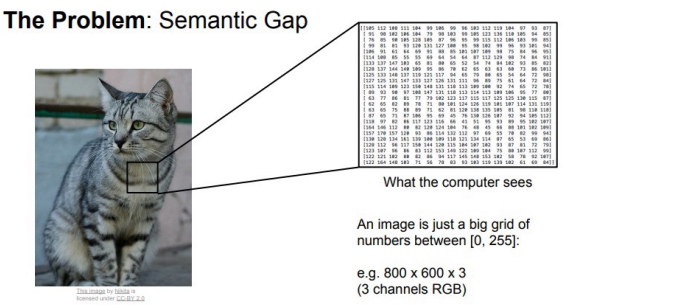
Lecture 2:
Image Classification pipeline

1/10 <CS231n 강의>

Lec. 02 Image Classification

사람과 기계가 이미지를 인식하는게 다르다.

(우리는 이미지 자체) (기계 ⇒ 픽셀로 인식을 함)



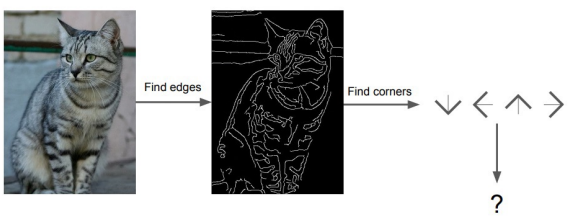
컴퓨터랑 시각 체계가 다르다!!

Challenges: Illumination

우리는 역광이나 이런걸 쉽~게 보고 이게 뭔지를 인식할 수 있다!!

but 기계는 주어진 사진으로만 판단가능 ⇒ 한계

Attempts have been made



물체를 코드로 구현 is not easy

⇒ 다른 방법 사용

왜 not easy??

⇒ 고양이의 edges 같은걸 때려 코드를 짰는데 ! 만약 사진이 뒤집히거나 그러면 기계는 인식 잘못함

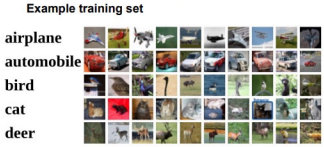
Data-Driven Approach

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

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

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

Example training set



train

⇒ 이미지를 주고 label을 주면 많은 데이터로 학습함

predict

⇒ 새로운 데이터를 주고 이 사진이 무엇인지 인식하게 해보자.

Example Dataset: **CIFAR10**

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

여기 !!

① Nearest Neighbor

↳ 가장 가까운 이미지를 찾음

* 근데 개구리랑 고양이가 같다고 본다 !! (색이 비슷해서...)

Distance Metric to compare images

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

add → 456

거리차

값을 다 더했을 때 합이 작으면 근접함이다!!


```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
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: Train O(1), predict O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

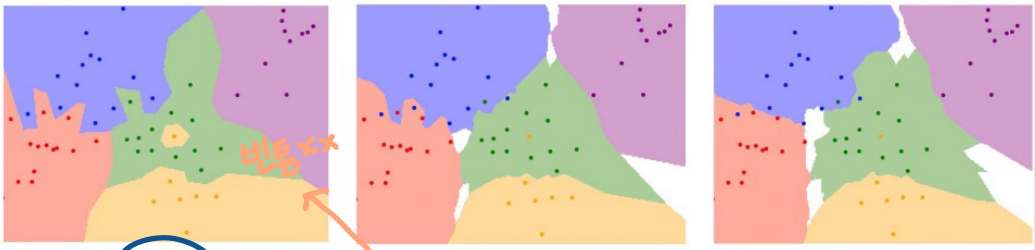
train is fast
test is long time

!! 문제점 !!
사람이 오래걸린다

⇒ 실생활 활용이 어렵다.
학습은 오래걸려도 test는 ↓

K-Nearest Neighbors

Instead of copying label from nearest neighbor,
take **majority vote** from K closest points



면 정확도↓ but 경분 good

KNN (k-최근접 이웃)

k=1
k=3 → □ → □□□
k=5

3번째까지 고려하겠다!

어떻 알해줘!

K가 늘어날수록 정확도가 ↑↑

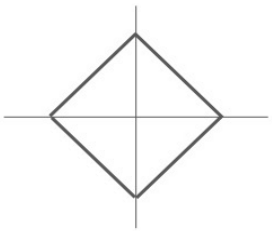
but, 경계가 뚜렷하↓ xx 경계가 모호함.

전부 장.단점 존재!!

K-Nearest Neighbors: Distance Metric

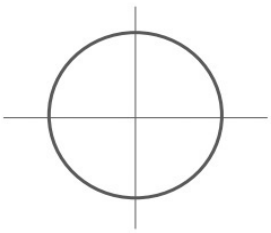
L1 (Manhattan) distance

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



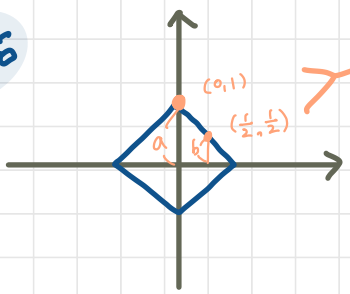
L2 (Euclidean) distance

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



L1 distance (Manhattan)

특이함

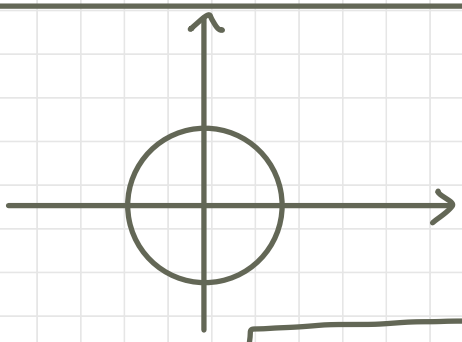


원래는 a와 b 거리가 다르다!

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

but!! Manhattan은 마름모 안에 있는 모든 점의 거리가 같다.

L2 distance (Euclidean)



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

좌표평면 내의 거리계산

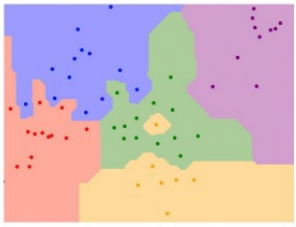
(a_1, a_2) (b_1, b_2) → 점 2개

$$\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$$

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

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



K = 1

L2 (Euclidean) distance

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

K = 1

L1 & L2 → 예시를 보면 쉽다.

L1은 거리가 같지만!

L2는 값이 다르게 나오는걸 알 수 있다.

L1 distance는 크기를 키우거나, 줄이는거 같은 행위가 가능...?

L2 distance는 feature 들에 대해서 각각을 반영하겠다.

ex) 키, 몸무게는 각각 반영 ⇒ L2 distance 사용

L1은 오밀조밀하게 분포 vs. L2는 sharpe 하게 있는거.

Hyperparameters (KNN 에서 K가 하이퍼파라미터)

파라미터 → 모델안에서 모델이 스스로 업데이트 해가는 것

하이퍼 파라미터 → 모델이 학습할 때 이것에 초점을 맞추면 좋겠다 ~~ 이런느낌?
⇒ 사용자가 지정하는 것!
ex) K=1, K=5

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.

k-Nearest Neighbor on images **never used.**

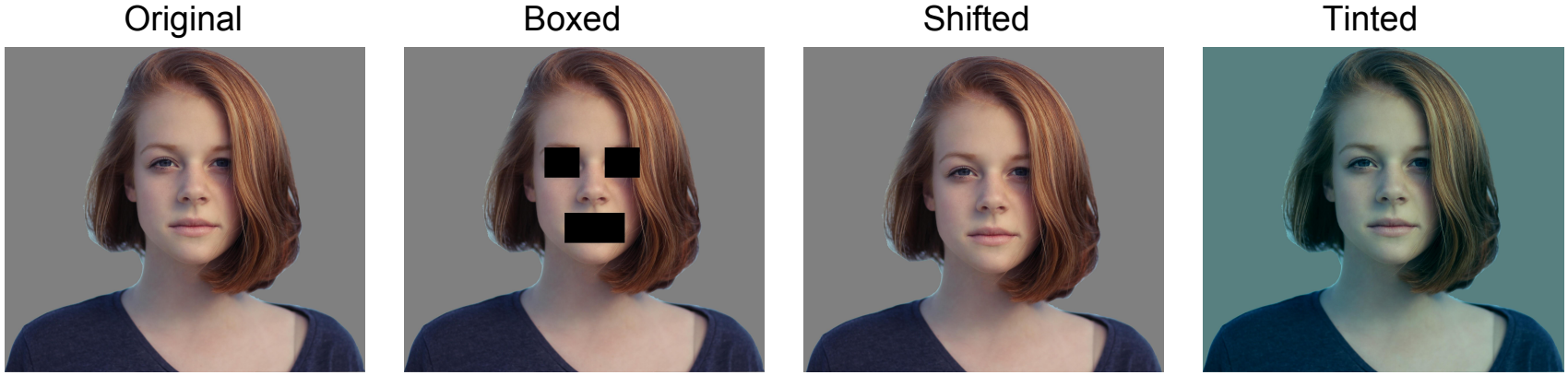
KNN은 이미지 분류에 적합xxx

- Very slow at test time
- Distance metrics on pixels are not informative

① test time이 느리다.. T

② 감, 변형, 위치이동 (픽셀이 조금만 달라도!) 하면 다 다른 이미지로 인식함

⇒ 성능이 좋은 알고리즘은 아니다!!!



(all 3 images have same L2 distance to the one on the left)

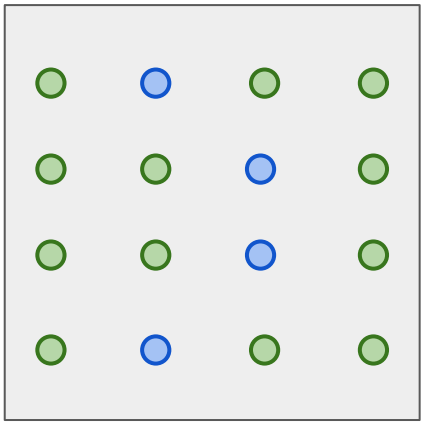
< 차원의 저주 ... ?? > ~> 뒷장에 계속 !!

k-Nearest Neighbor on images **never used.**

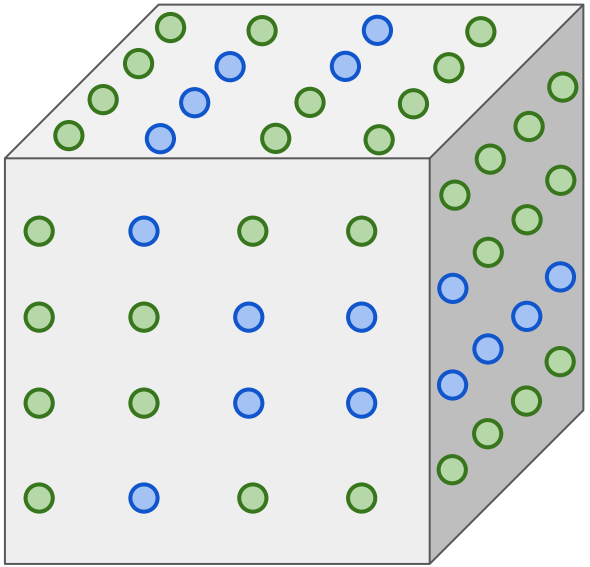
- Curse of dimensionality

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3



< 차원의 저주 >

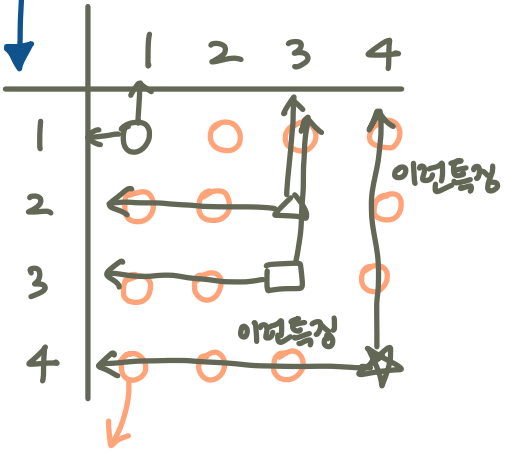
데이터가 가지고 있는게 많아서 (많아서!)
컴퓨터가 학습을 잘 못함.
→ 고려해야 하는 요소가 너무 많다.

data를 학습시킴
⇒ 패턴을 찾기 위함.
차원이 높을수록 ↑↑
패턴을 명확하게 하기 힘들다.

<해결방법>

① feature 에 대한 부분을 좀 더 명확

② PCA (차원축소 → 의미없는 feature을 쳐내는것)



빈 공간이 많으면
모델이 학습하기 어렵다!!

<우리가 보고싶은 data>

이런 데이터를
찾아서 빈공간을
메꾼다.
⇒ 온전한 matric

이게
데이터의 수를
늘리기 위해서 (만개 → 2만개)

K-Nearest Neighbors: Summary <KNN Summary>

(최근접 이웃 알고리즘)

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

The **K-Nearest Neighbors** 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!

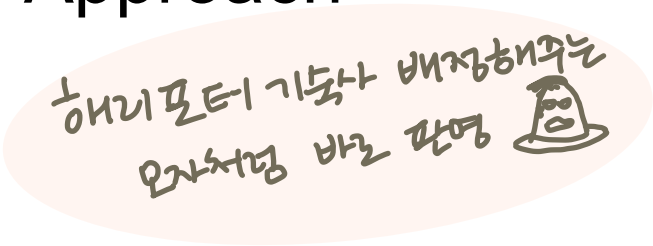
↳ 직선
↳ 선형적인걸 바탕으로 분류를 해낸겠다.

Linear Classification

Neural Network

Linear classifiers

Parametric Approach



Image

Array of **32x32x3** numbers
(3072 numbers total)

f(**x**,**W**)

10 numbers giving class scores

W

파라미터 or 가중치

parameters or weights

Parametric Approach: Linear Classifier

(행렬의 연산으로 사이즈결정)

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

3072x1 (x) · 10x1 (b) · 10x3072 (W) · 10x1 (f(x,W))

W와 x의 내적!!

Image

Array of **32x32x3** numbers
(3072 numbers total)

f(**x**,**W**)

10 numbers giving class scores

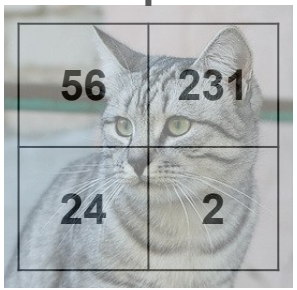
W
parameters or weights

X = 3072x1

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Stretch pixels into column

x를 벡터 사용



Input image

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W

56
231
24
2

x

+

1.1
3.2
-1.2

=

-96.8	Cat score
437.9	Dog score
61.95	Ship score

Interpreting a Linear Classifier

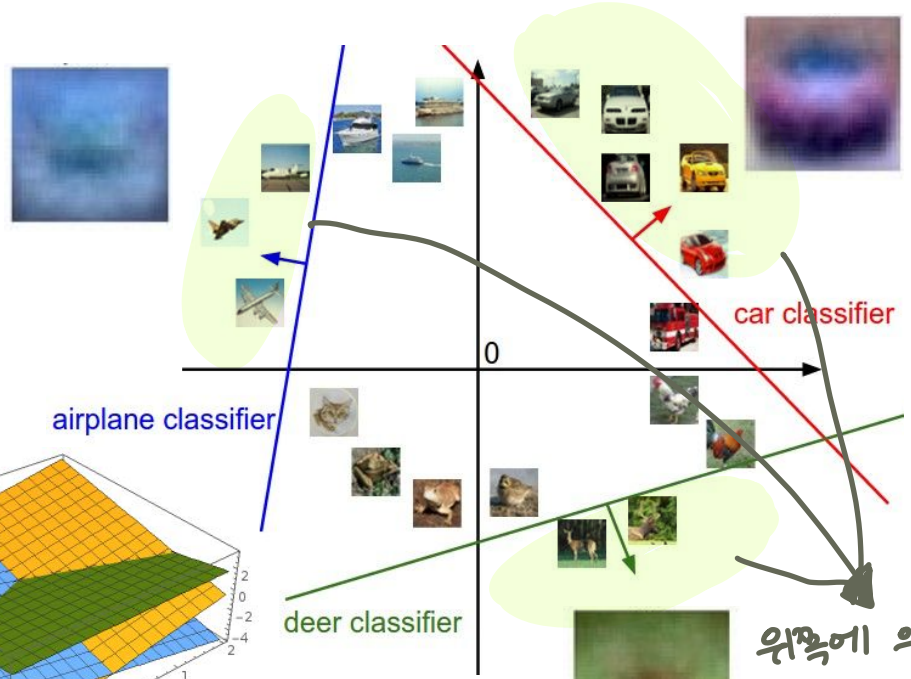
학습할 때 데이터들의 내용이 함축된게 뽑혀 나온다!!

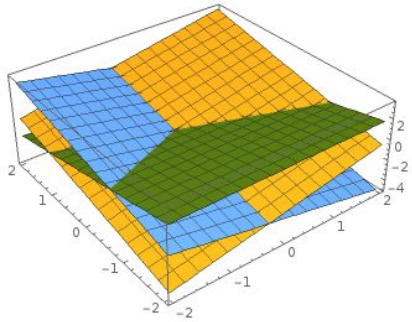
$$f(x,W) = Wx + b$$

Weight 들을 하나의 가중치로 만든다!
~> 하나의 그림판에 각각의 그림을 녹여 내야하니!!

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

Interpreting a Linear Classifier





Plot created using Wolfram Cloud

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

Array of **32x32x3** numbers
(3072 numbers total)

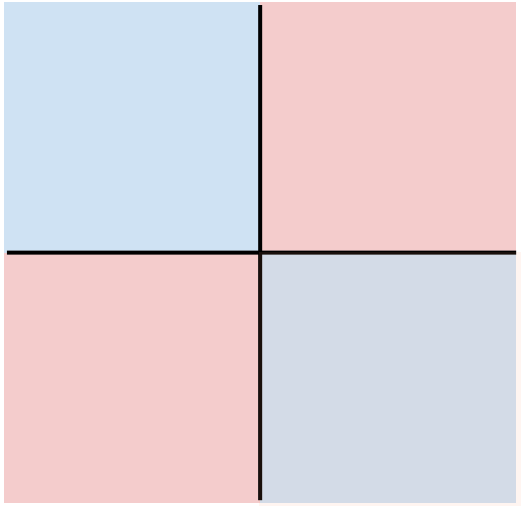
위쪽에 위치한게 잘 분류되었나?? (개별적 접근)

⇒ Linear하게 분류가능하다!!

Hard cases for a linear classifier

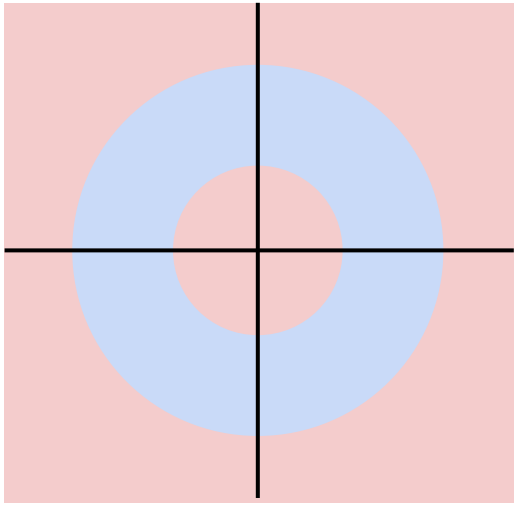
Class 1:
number of pixels > 0 odd

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