



Type something...

01) Image Classification

01_Image Classification

02_Nearest Neighbor Classifier(KNN)

03_Distance Metric

04_Linear Classification

01) Image Classification

01_Image Classification

An image classifier

```
def predict(image):
    # ???
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for
recognizing a cat, or other classes.

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 2 -14

6 Jan 2016

- Image를 분류하는 hard-code algorithm은 없다.

Data-driven approach:

1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

Example training set

```
def train(train_images, train_labels):
    # build a model for images -> labels...
    return model

def predict(model, test_images):
    # predict test_labels using the model...
    return test_labels
```



- 그래서 Data를 학습해서 값을 예측하는 접근을 시도하였다.



02_Nearest Neighbor Classifier(KNN)

First classifier: Nearest Neighbor Classifier

```
def train(train_images, train_labels):
    # build a model for images -> labels...
    return model

def predict(model, test_images):
    # predict test_labels using the model...
    return test_labels
```

Remember all training images and their labels

Predict the label of the most similar training image

- 모든 training image를 비교해서 가장 가까운 것에 label을 사용한다.
- 현재는 아무도 사용하지 않는 알고리즘이다.

03_Dlstance Metric

How do we compare the images? What is the **distance metric**?

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

test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
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

add → 456

- test image - training image의 절대값을 써워준 형태를 사용한다
- L1 distance를 통해 구한 거리 중에서 가장 작은 값이 그 test_image의 label이 된다.



```

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

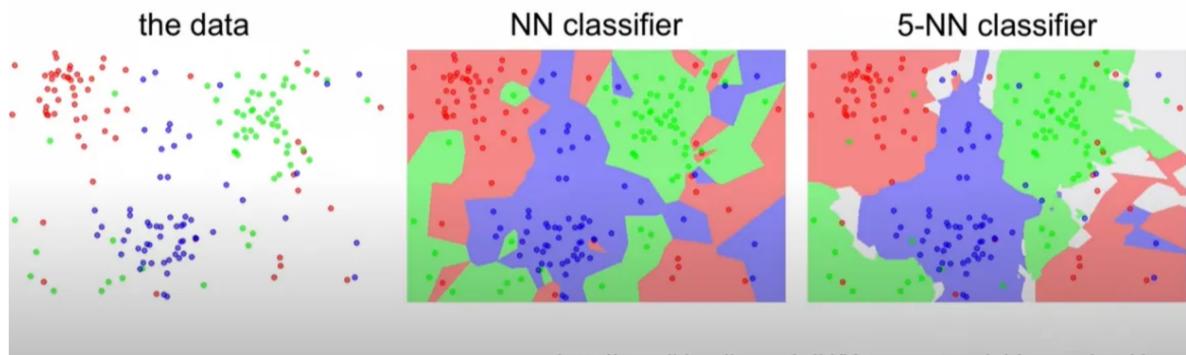
```

Nearest Neighbor classifier

- train 과정이 단순히 데이터를 기억하기만 하기 때문에 시간이 매우 짧다. O(1)
- predict에서는 X를 모두 다 돌아보기 때문에 O(N)
- We want to **Train slow, Predict Fast**

k-Nearest Neighbor

find the k nearest images, have them vote on the label



http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

- K가 커질수록 smooth하다.



Hyperparameters

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

Fei-Fei Li & Justin Johnson & Serena Yeung

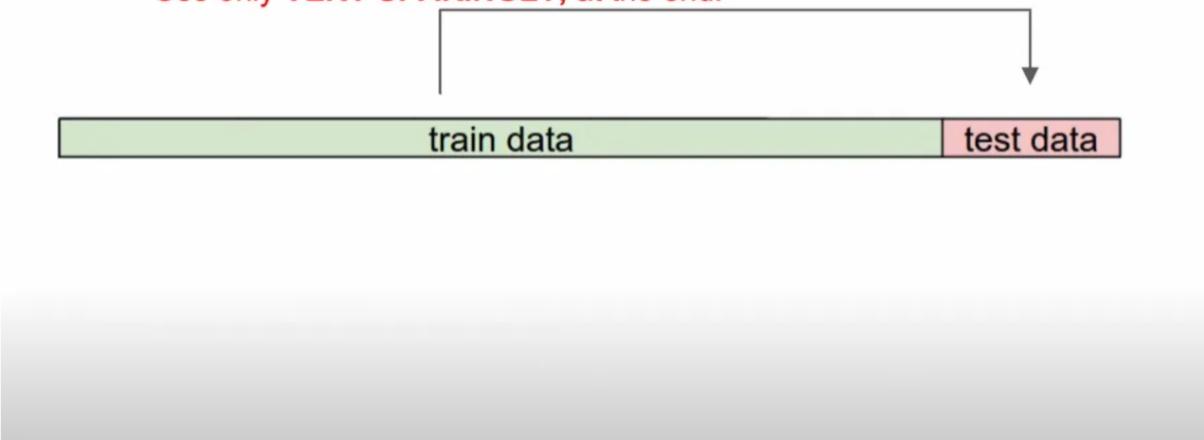
Lecture 2 - 34

April 6, 2017

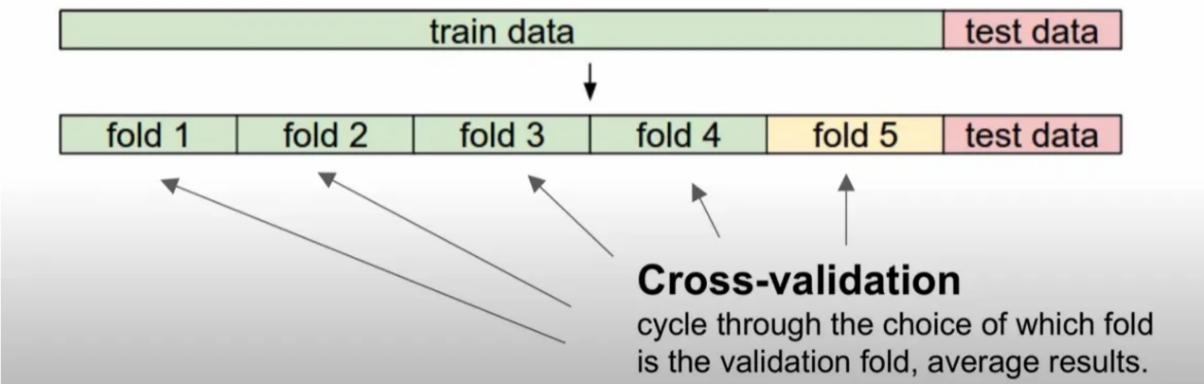
- 가장 최적의 k값, 가장 최적의 distance를 구하는 것들, 그런 것들을 **hyperparameters**라고 한다.

Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY**, at the end.



- test data는 test 때 딱 한번만 사용하는 것이다. 절대로 train 때 사용해서는 안된다.

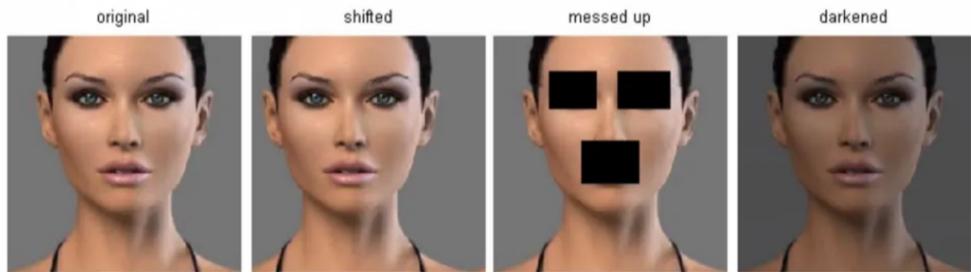


- validation set을 포함한 다음, 각각 하나씩 제외하고 남은 하나로 검토하는 **Cross-validation**을 사용하면 매우 좋다.



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

- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive



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

- 절대로 KNN을 image에 사용해서는 안된다.
- test time이 매우 느리고, 픽셀에 대한 거리가 별로 유용하지 않다.

K-Nearest Neighbors: Summary

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

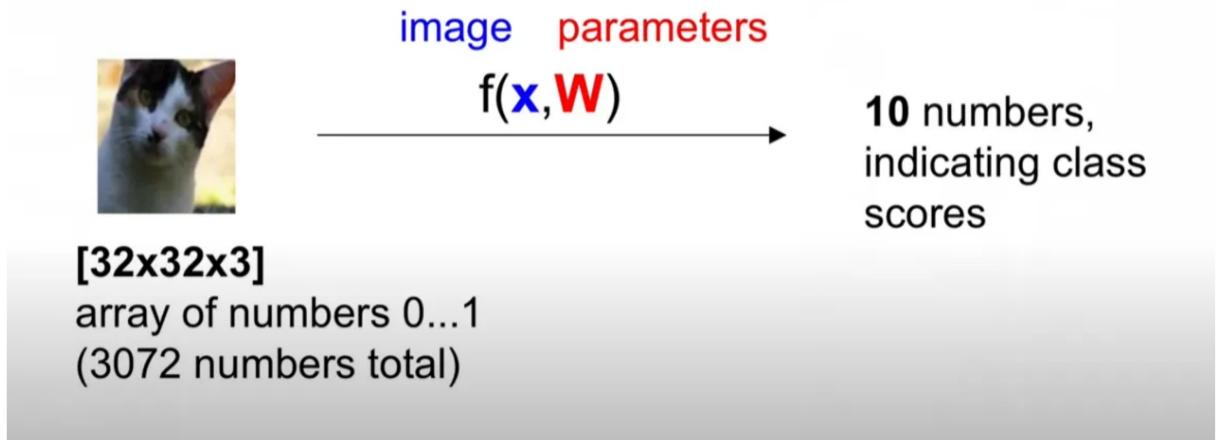
Distance metric and K are **hyperparameters**

Choose hyperparameters using the **validation set**; only run on the test set once at the very end!

<요약>

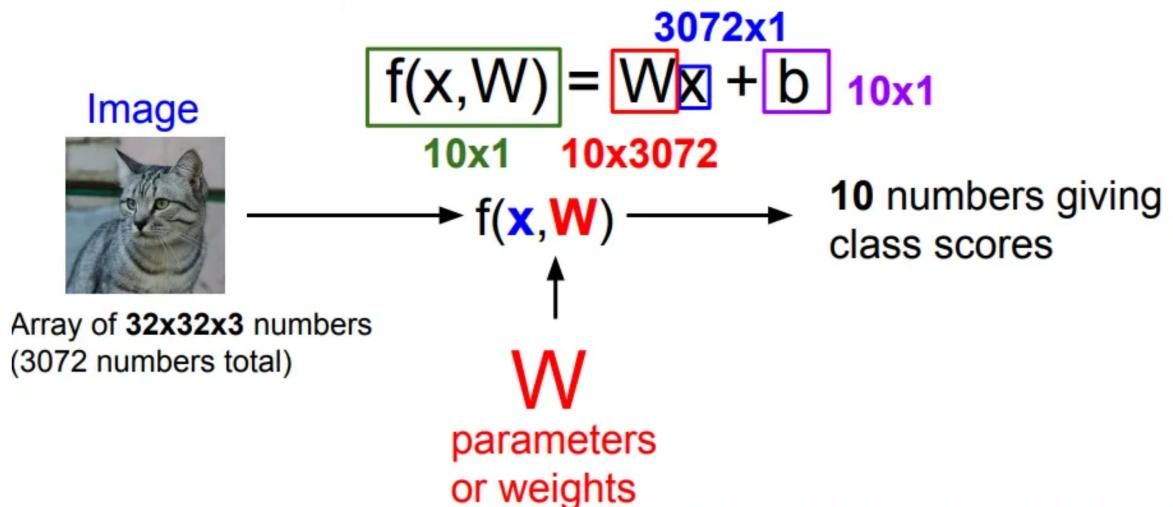
- Image classification에서는 training set과 test set으로 나눈다.
- KNN은 거리를 이용한 predict 방식이다
- Distance metric 과 K값은 모두 hyperparameter이다
- validation set을 사용해서 hyperparameter를 선택한다.

Parametric approach



- $f(x, W)$: image + parameters

Parametric Approach: Linear Classifier



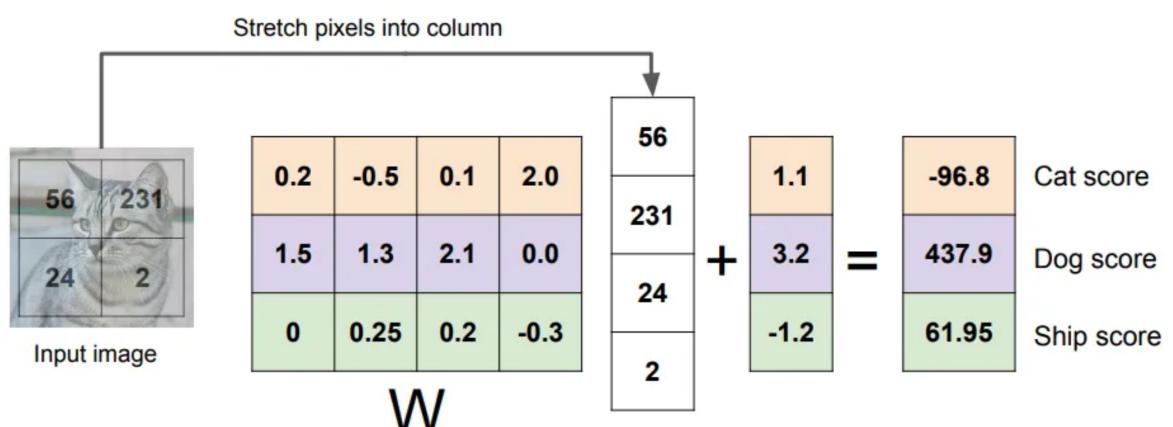
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 54

April 6, 2017

- $f(x, W) = Wx + b$
- $10 \times 1 = 10 \times 3072 * 3072 \times 1 + 10 \times 1$

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



- 예시는 다음과 같다.

