Image Classification pipeline

| 고려대학교 산업경영공학과
| 서덕성

INDEX

- Image classification & **Challenges**
- K-NN based image classification
- Linear image classification

Image classification & Challenges

❖ Image

- Images are represented as 3D arrays of numbers, with integers between [0, 255]. (brightness at a point)
- E.g. 300 X 100 X 3 (3 for 3 color channels RGB)

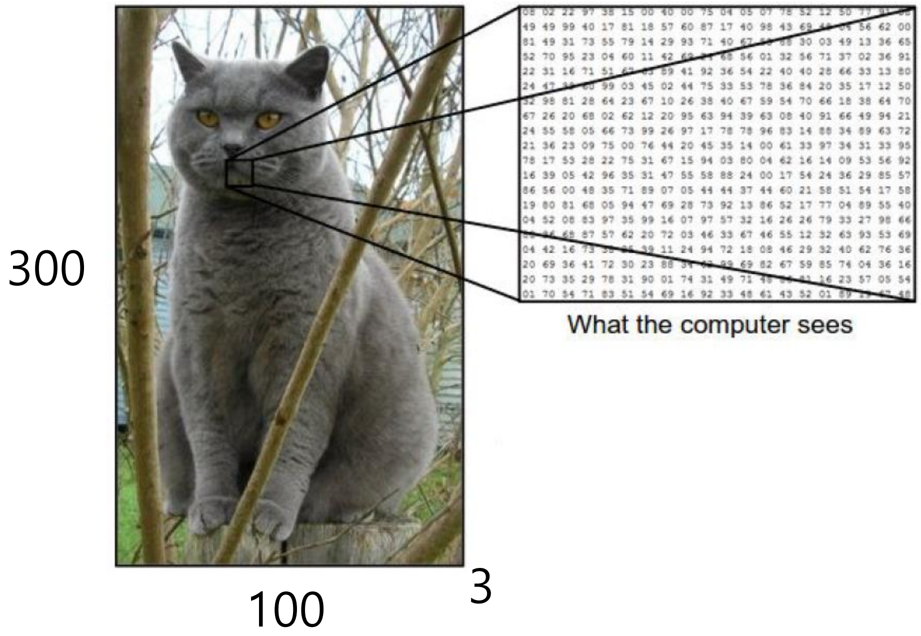
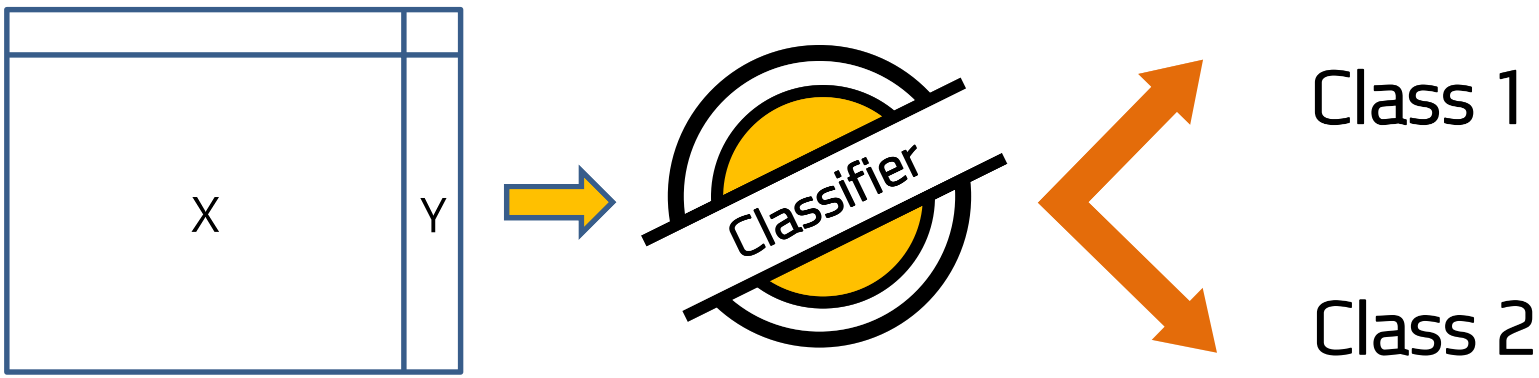


Image classification & Challenges

❖ Image classification

- Core task in Computer Vision
- Image is input data

<structured, unstructured(text, …)>



<Image>

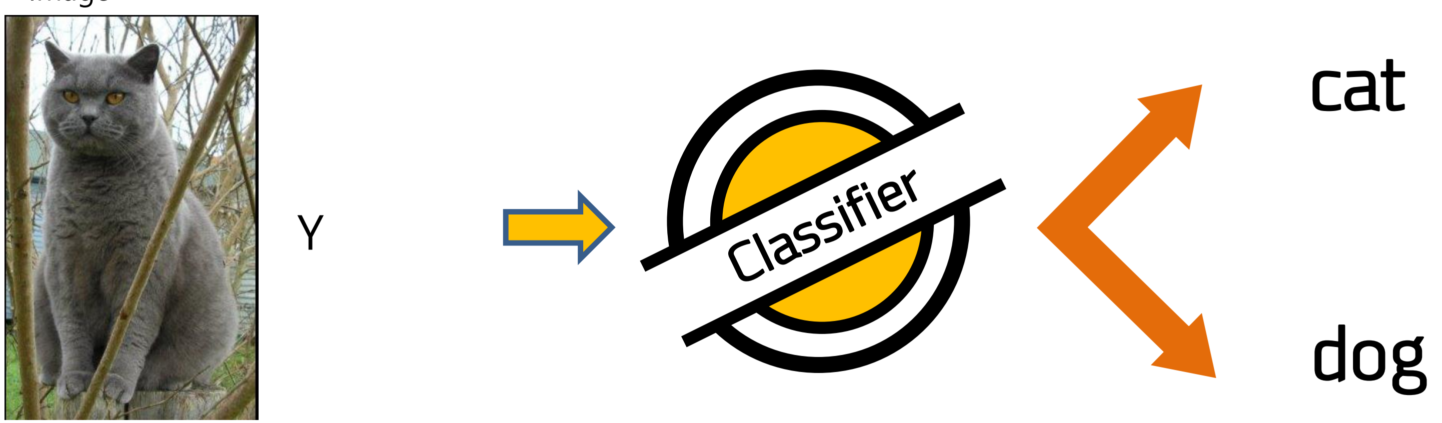


Image classification & Challenges

❖ Challenges

- Viewpoint Variation
- Rotate, zoom in/out

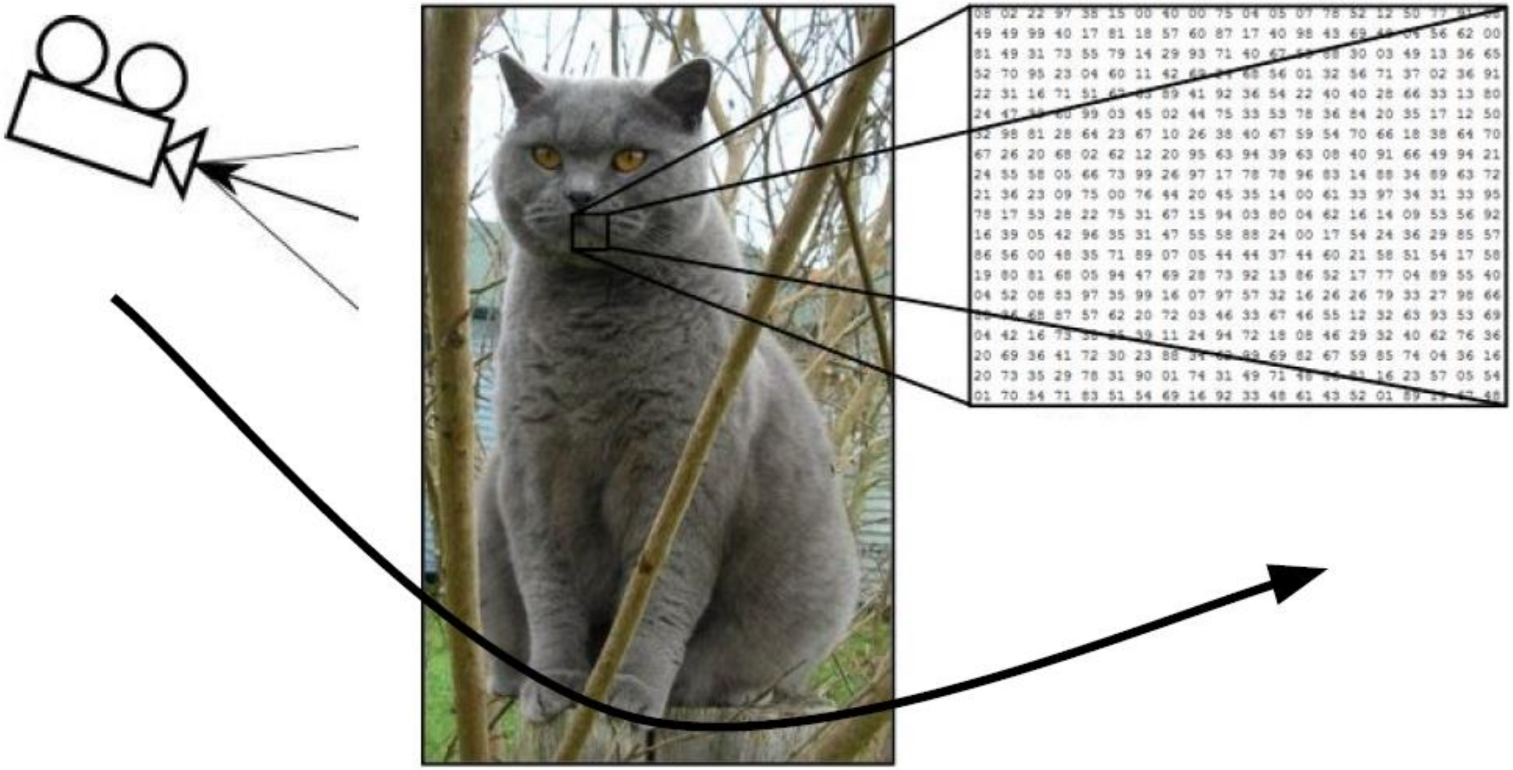


Image classification & Challenges

01

02

03

❖ Challenges

- Illumination

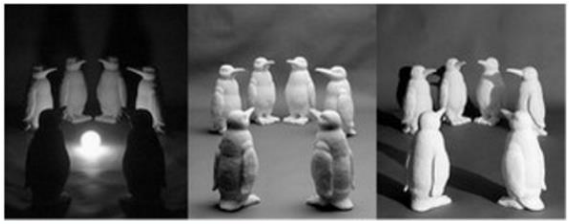


Image classification & Challenges

❖ Challenges

- Deformation

Image classification & Challenges

01

02

03

❖ Challenges

- Occlusion

Image classification & Challenges

01

02

03

❖ Challenges

- Background clutter

Image classification & Challenges

01

02

03

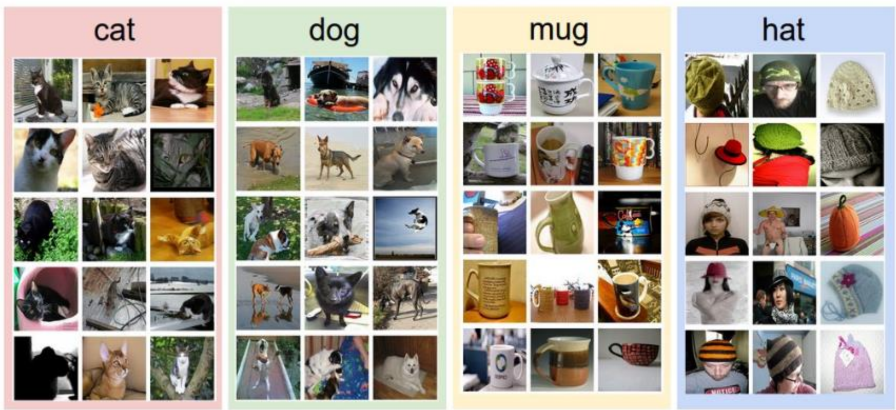
❖ Challenges

- Intra-class variation

Image classification & Challenges

❖ 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




```python
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
```

K-NN based image classification

K-NN based image classification

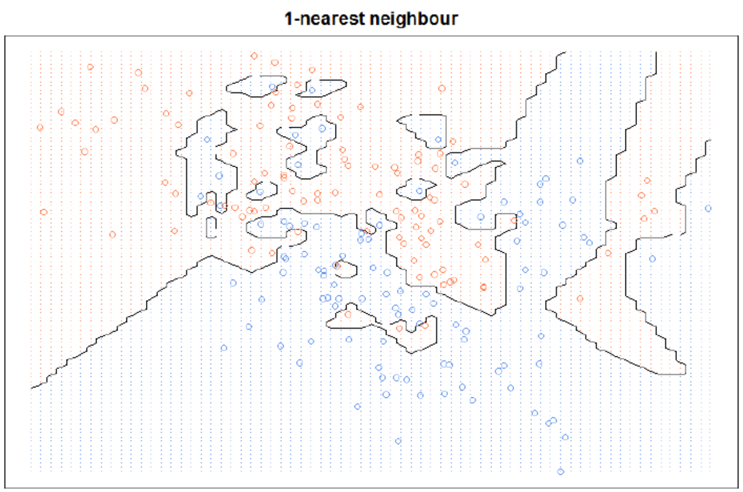
01

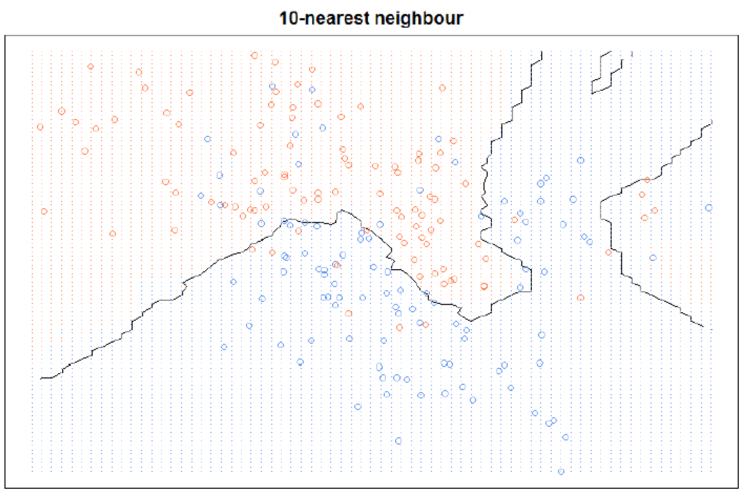
02

03

❖ K-NN

- Lazy model
  - Remember all training data
  - Predict the label of the most similar
- Hyperparameter
  - K
  - Voting
  - Distance metric





K-NN based image classification

❖ Distance metric

- **L1 distance (Manhattan)**
  - $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$
- **L2 distance (Euclidean)**
  - $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$

L1 distance example

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

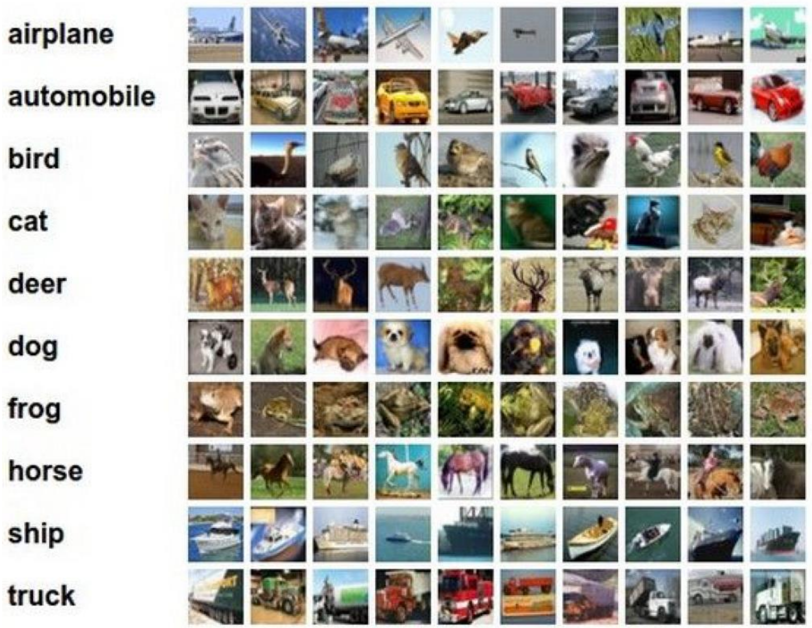
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

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

K-NN based image classification

❖ CIFAR-10

- 10 labels
- 50,000 training images (each image is tiny : 32 X 32)
- 10,000 test images



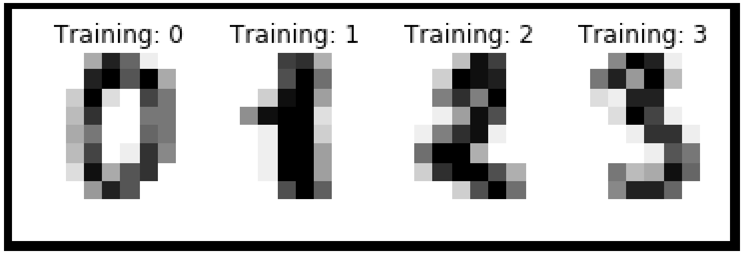
K-NN based image classification

01
02
03

❖ Code (data : hand-written digits _ 1797x8x8 images)

```
# 시각화 모듈 import
import matplotlib.pyplot as plt

# Import datasets and performance metrics
from sklearn import datasets, metrics

# The hand-written digits dataset (8x8 image)
digits = datasets.load_digits()

# Target
images_and_labels = list(zip(digits.images, digits.target))

# Visualization of 4 images
for index, (image, label) in enumerate(images_and_labels[:4]):
    plt.subplot(2, 4, index + 1) # 2 x 4의 subplot 위치를 할당하고, 각 image를 할당해서 그림
    plt.axis('off') #축 없음
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest') #image 파일을 시각화 하는 함수. 타입을 흑백으로.
    plt.title('Training: %i' % label) #title 설정
```

```
In [116]: images_and_labels[1]
Out[116]:
(array([[  0.,   0.,   0.,  12.,  13.,   5.,   0.,   0.],
        [  0.,   0.,   0.,  11.,  16.,   9.,   0.,   0.],
        [  0.,   0.,   3.,  15.,  16.,   6.,   0.,   0.],
        [  0.,   7.,  15.,  16.,  16.,   2.,   0.,   0.],
        [  0.,   0.,   1.,  16.,  16.,   3.,   0.,   0.],
        [  0.,   0.,   1.,  16.,  16.,   6.,   0.,   0.],
        [  0.,   0.,   1.,  16.,  16.,   6.,   0.,   0.],
        [  0.,   0.,   0.,  11.,  16.,  10.,   0.,   0.]]), 1)
```



K-NN based image classification

01

02

03

❖ Code

```
# K-NN classification
import numpy as np
import matplotlib.pyplot as plt
import math

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """X : N x D training data set, Y : N x 1 label"""
        #the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """X : M x D test data set"""
        num_test = X.shape[0] #행 개수(M)
        Ypred = np.zeros(num_test, dtype=self.ytr.dtype) #먼저 0으로 가득 채워둠

        for i in np.arange(len(X)): #모든 test data에 대해
            print(i, '\n')

            distance = np.zeros(len(self.Xtr))
            for j in np.arange(len(self.Xtr)):
                distances = np.sum(np.abs(self.Xtr.iloc[j] - X.iloc[i]), axis=0) #trining data와 L1-norm을 구함
                distance[j] = distances

            min_index = np.argmin(distance) #가장 가까운 1-NN의 index를 구함
            Ypred[i] = self.ytr.iloc[min_index] #가장 가까운 이웃의 label을 적용

        return(Ypred)
```

K-NN based image classification

❖ Code

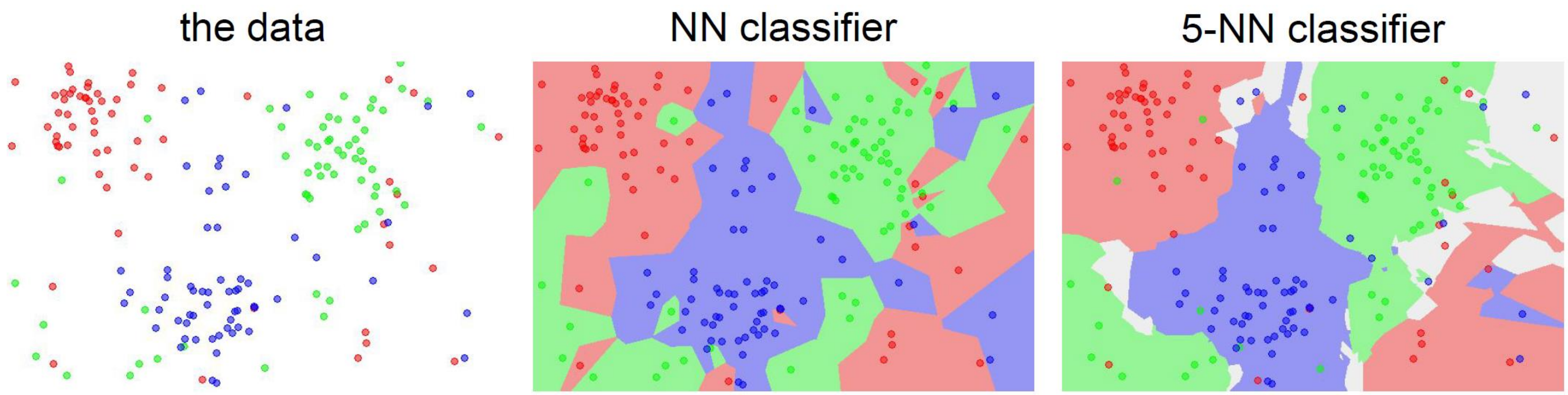
```
#training / test split
# Randomly shuffle the index of nba.
random_indices = np.random.permutation(data.index)
# Set a cutoff for how many items we want in the test set (in this case 1/3 of the items)
test_cutoff = math.floor(len(data)/3)
# Generate the test set by taking the first 1/3 of the randomly shuffled indices.
ts = data.loc[random_indices[1:test_cutoff]]
# Generate the train set with the rest of the data.
tr = data.loc[random_indices[test_cutoff:]]

#Object
nn = NearestNeighbor()

#training (just save)
nn.train(tr.ix[:, 0], tr.ix[:, 1])
Ypred = nn.predict(ts.ix[:, 0])
```

❖ Output

```
array([9, 1, 5, 5, 7, 3, 5, 7, 3, 5, 0, 7, 9, 0, 8, 1, 3, 3, 4, 8, 5, 0, 9,
       4, 7, 2, 8, 6, 8, 4, 9, 9, 0, 1, 5, 8, 0, 3, 9, 1, 7, 1, 6, 0, 7, 1,
       4, 1, 4, 3, 1, 0, 5, 3, 1, 0, 9, 7, 3, 6, 8, 3, 8, 9, 0, 2, 6, 7, 9,
       4, 5, 3, 3, 1, 4, 7, 7, 2, 4, 2, 5, 3, 7, 6, 1, 3, 5, 2, 3, 8, 5, 5,
       0, 5, 9, 0, 2, 5, 3, 6, 5, 3, 7, 2, 0, 2, 3, 2, 4, 6, 7, 6, 1, 4, 3,
       9, 1, 3, 1, 4, 6, 1, 8, 3, 6, 5, 6, 3, 8, 1, 0, 0, 9, 3, 9, 5, 6, 2,
       4, 0, 1, 7, 2, 5, 8, 9, 3, 3, 6, 9, 9, 7, 2, 6, 0, 4, 1, 7, 4, 4, 2,
       3, 6, 5, 0, 6, 1, 5, 6, 3, 6, 6, 4, 9, 9, 9, 7, 7, 1, 0, 7, 0, 0, 3,
       1, 7, 8, 1, 7, 7, 6, 8, 0, 7, 5, 2, 2, 7, 8, 4, 7, 8, 7, 8, 7, 3, 1,
       9, 0, 0, 7, 4, 5, 6, 0, 0, 5, 0, 7, 4, 7, 4, 9, 0, 8, 3, 1, 6, 7, 6,
       6, 5, 2, 8, 0, 7, 5, 9, 5, 3, 4, 4, 2, 8, 0, 9, 1, 5, 3, 5, 8, 1, 9,
       3, 7, 9, 5, 0, 5, 7, 4, 8, 2, 5, 2, 0, 1, 9, 3, 2, 1, 2, 6, 1, 4, 6,
       8, 1, 9, 1, 8, 9, 4, 1, 5, 7, 5, 3, 1, 0, 8, 7, 9, 7, 4, 3, 8, 5, 8,
       7, 4, 1, 5, 0, 6, 0, 6, 6, 6, 7, 9, 5, 0, 7, 4, 3, 1, 4, 7, 3, 6, 6,
       4, 8, 1, 1, 5, 1, 6, 2, 9, 9, 6, 5, 9, 3, 7, 0, 0, 3, 5, 3, 8, 4, 2,
       4, 1, 7, 1, 5, 4, 4, 8, 1, 9, 4, 9, 6, 9, 9, 1, 2, 6, 6, 7, 3, 1, 3,
       1, 7, 2, 5, 7, 4, 6, 7, 1, 4, 3, 1, 0, 7, 6, 3, 6, 7, 6, 0, 3, 9, 3,
       0, 2, 7, 7, 2, 6, 8, 4, 5, 3, 8, 2, 2, 6, 8, 4, 4, 9, 8, 5, 8, 3, 3,
       1, 5, 1, 7, 4, 5, 5, 1, 9, 0, 2, 6, 1, 7, 6, 9, 7, 8, 8, 1, 6, 3, 8,
       2, 9, 7, 8, 1, 6, 7, 3, 3, 6, 1, 0, 9, 8, 7, 1, 1, 6, 0, 7, 9, 5, 1,
       3, 7, 6, 3, 9, 6, 7, 2, 0, 6, 7, 4, 3, 6, 8, 0, 8, 8, 3, 1, 8, 5, 1,
       5, 8, 8, 3, 8, 8, 6, 4, 2, 2, 8, 8, 5, 8, 6, 2, 2, 2, 9, 5, 0, 9, 5,
       0, 1, 8, 4, 4, 1, 2, 2, 4, 9, 6, 2, 9, 4, 2, 8, 5, 6, 9, 4, 0, 3, 4,
       2, 0, 8, 1, 1, 2, 3, 3, 9, 2, 9, 4, 4, 8, 9, 2, 5, 8, 8, 1, 4, 2, 3,
       0, 9, 0, 6, 5, 8, 0, 6, 3, 4, 2, 9, 1, 6, 8, 0, 9, 3, 1, 2, 2, 3, 9,
       4, 6, 9, 4, 8, 7, 9, 4, 3, 0, 7, 4, 1, 4, 6, 4, 6, 0, 9, 7, 8, 4, 8], dtype=int64)
```

K-NN based image classification

❖ Result



- **What is the best (distance and) value of k?**
  - **Use to tune hyperparameters by validation data (Cross validation)**

Training data					Test data
Fold 1	Fold 2	Fold 3	Fold 4 (Validation)	Fold 5	Test data

K-NN based image classification

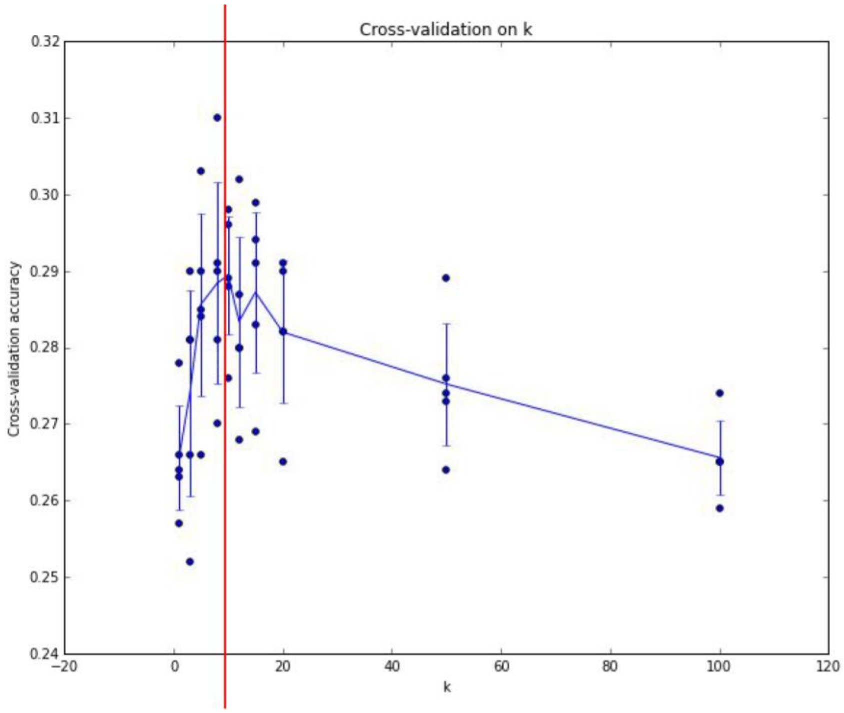
01

02

03

❖ 5-fold cross validation Result

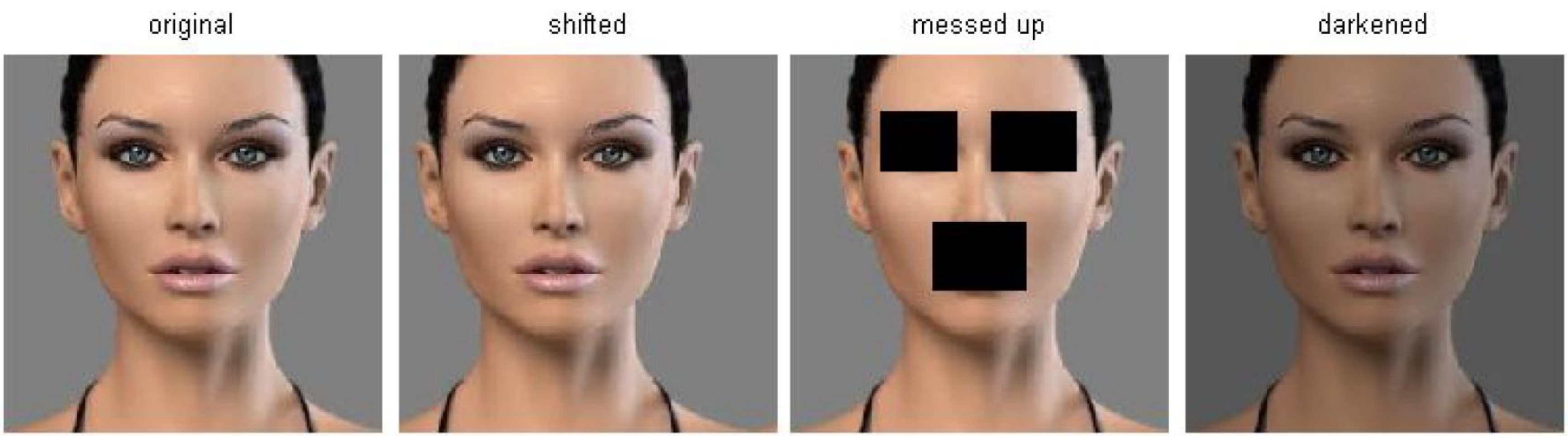
- Each point : single outcome
- Blue line goes through the mean
- Bars indicated standard deviation



K-NN based image classification

❖ Summary

- K_NN on images **NEVER USED**
  - Terrible performance



✓ All 3 images have **same L2 distance** to the one on the left

Linear image classification

Linear image classification

❖ Parametric approach

- 32 X 32 X 3 (=3072) array of numbers 0...1

Image Parameters

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

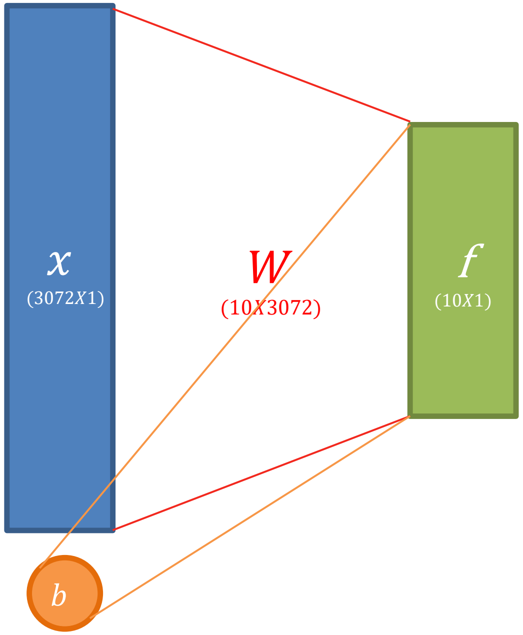
$$f : R^{3072} \rightarrow R^{10}$$

$P(C_1)$
$P(C_2)$
$P(C_3)$
$P(C_4)$
$P(C_5)$
$P(C_6)$
$P(C_7)$
$P(C_8)$
$P(C_9)$
$P(C_{10})$

3072x1

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

10x1 10x3072 10x1

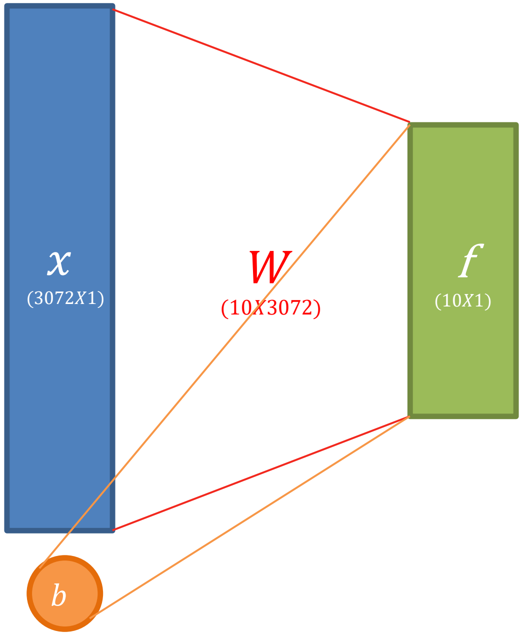


Linear image classification

❖ Parametric approach

- Ex. with 4pixels, and 3 classes(cat/dog/ship)

$$\underset{3x1}{f}(x, W) = \underset{3x4}{W}\overset{4x1}{x} + \underset{3x1}{b}$$



Stretch pixels into single column

$$\underset{W}{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0.0 & 0.25 & 0.2 & -0.3 \end{bmatrix}} \underset{x_i}{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}} + \underset{b}{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}} \longrightarrow \underset{f}{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}} \begin{matrix} \text{cat} \\ \text{dog} \\ \text{ship} \end{matrix}$$

Linear image classification

❖ Interpreting a Linear classifier

- After training, we can make weight to template form.

1. In "Plane" template, R(row), G(row), B(high)
2. "Car" looks like red.

   -> There are many red cars.

   (Yellow car may be predicted to frog)
3. "Horse" have 2 heads.

   -> Almost same as average of each category image.

Linear image classification

❖ Result of Linear classifier

- Example class scores for 3 images, with a random W
  - -> Bad performance
- Need to optimize W
  - Define loss
  - Minimize loss

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

Q & A

01

02

03