D. 기계학습 & 딥러닝 IN OPENCV

1

기계학습(Machine Learning)과 신경망(Neural Network), 딥러닝(Deep Learning

2

기계학습(Machine Learning)

- 인공지능, 머신러닝

인공지능

사고나 학습 등 인간이 가진 지적 능력을 컴퓨터를 통해 구현하는기술

머신러닝, 컴퓨터 비전, 자연어 처리, 로봇 공학 및 그와 관련된 모든 주제를 포괄하는 개념

머신러닝

- 인공 지능의 한 분야로,
- 환경과의 상호작용에 기반한 경험적인 데이터로부터 스스로 성능을 향상시키는 기술 방법
- 컴퓨터에게 배울 수 있는 능력, 즉 코드로 정의하지 않은 동작을 실행하는 능력에 대한 연구 분야

신경망, 서포트 벡터 머신, 결정 트리(Decision tree), 베이지안 신뢰 네트워크, k 최근접 이웃, 자기 조직화 지도, 사례 기반 추론, 인스턴스 기반 학습, 은닉 마르코프 모델, 회귀 기법 등

3

인공신경망(Artificial Neural Network : ANN)

인공지능

사고나 학습 등 인간이 가진 지적 능력을 컴퓨터를 통해 구현하는기술

머신러닝, 컴퓨터 비전, 자연어 처리, 로봇 공학 및 그와 관련된 모든 주제를 포괄하는 개념

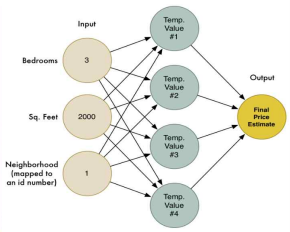
머신러닝

- 인공 지능의 한 분야
- 환경과의 상호작용에
  스스로 성능을 향상
- 컴퓨터에게 배울 수
  동작을 실행하는 능

신경망은
한 Layer(층) 내의 노드들이
다른(다음) Layer(층)의 노드와
연결되는 구조를 가짐

신경망

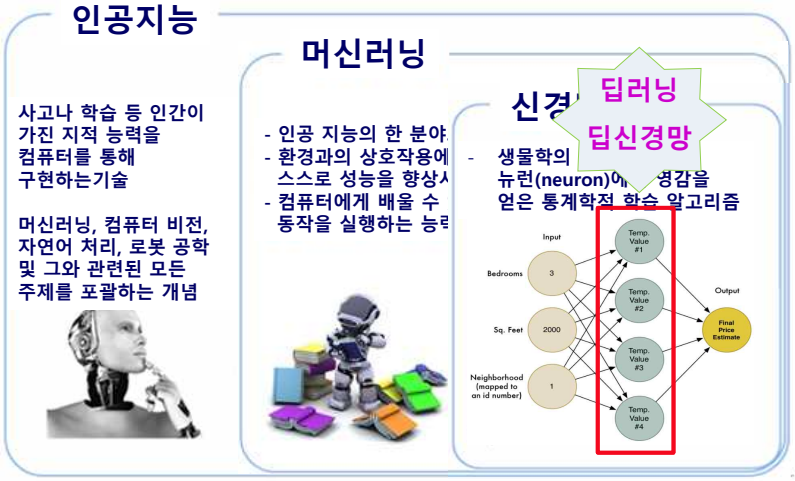
- 생물학의 신경망의 뉴런(neuron)에서 영감을 얻은 통계학적 학습 알고리즘



입력층 (input) 은닉층 (hidden) 출력층 (output)

4

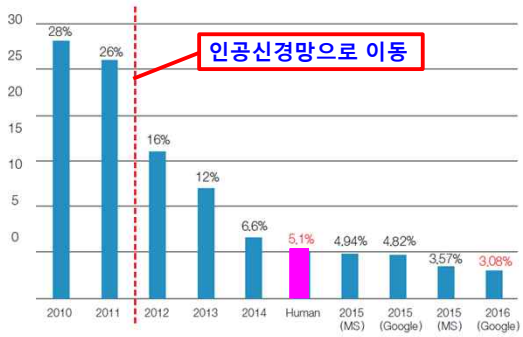
딥러닝/딥신경망(Deep Learning / Neural Network)



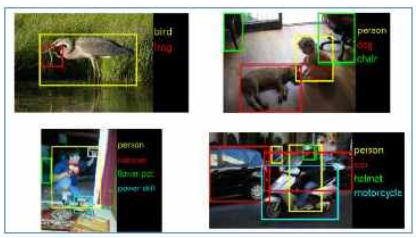
5

딥러닝 비교 사례

- 사진 내 물체 인식
  - ILSVRC : ImageNet이 제공하는 1000여 카테고리로 분류된 100만개의 이미지를 인식하여 그 정확도를 겨루는 대회
  - 2012년 Alex Krizhevsky 등은 computer vision(CV, 컴퓨터 시각) 기법을 사용하지 않고, 기존 CV 전문가들과 큰 격차로 1위 차지(AlexNet)
  - 2015년 이후 '인간'보다 인식율 향상, 상위 결과는 모두 딥러닝 적용



ILSVRC 인식 오류율(= 1 - 인식률)



6

K-Means Clustering

7

K-means 알고리즘

- **K-means 알고리즘**(K-means clustering algorithm)
  - 주어진 데이터를 k개의 클러스터로 묶는 알고리즘
  - 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작
  - 이 알고리즘은 자율(비지도, Unsupervised) 학습의 일종
    - 레이블이 달려 있지 않은 입력 데이터에 레이블을 달아주는 역할을 수행

① **임의의 k개 중심을 선정합니다. (초기치 중심을 설정해줘야 합니다)**
- 랜덤하게 각 샘플의 중심이라고 간주할 위치를 임의로 선택
- 초기 중심점을 어떻게 선정하냐에 따라 결과값이 달라지게 됨

② **모든 데이터에 대하여 가장 가까운 중심을 선택하여 이동합니다.**
- 중심점 간의 거리의 중간으로 영역을 분할

③ **각 군집에 대해 새로운 중심을 다시 계산합니다.**
- 중심을 이동하고 클래스에 해당하는 영역을 다시 선정

④ **중심이 변경되면 ②~③ 과정을 반복합니다.**
- 중심이 바뀌지 않을 때 까지 반복

⑤ **중심이 변경되지 않으면 종료합니다.**

8

K-means 알고리즘

- **cv2.kmeans(data, K, bestLabels, criteria, attempts, flags, centers=None) -> retval, label, centers**
  - 입력
    - data : 입력 데이터
    - K : 클러스터 수
    - criteria : 반복 종료 기준
      - OpenCV 함수에서는 종료 기준을 선정할 수 있음
      - cv.TERM_CRITERIA_EPS - 지정된 정확도인 엡실론에 도달하면 알고리즘 반복을 중지
      - cv.TERM_CRITERIA_MAX_ITER - 지정된 반복 횟수 max_iter 후에 알고리즘을 중지
      - max_iter - 최대 반복 횟수를 지정
      - 엡실론 - 필요한 정확도
    - attempts: 다른 초기 레이블을 이용해 반복 실행할 횟수
    - flags :이 플래그는 초기 중심을 구하는 방법을 지정
      - 일반적으로 cv2.KMEANS_PP_CENTERS 및 cv2.KMEANS_RANDOM_CENTERS를 사용
  - 출력
    - `retval` : 각 데이터에서 해당 중심까지의 거리 제곱의 합, 군집 내 데이터들의 응집도(compactness)
    - `label` : 각 요소가 '0', '1'로 표시된 레이블 배열
    - centers : 클러스터 중심의 배열

9

Color Quantization + K-means

- **Color Quantization(색상 양자화) + K-means**
  - 이미지의 색상 수를 줄이는 프로세스
    - K-means 알고리즘을 이용하여 이미지의 색상을 `K`개로 줄임
  - 색상을 이용하여 영상을 분할하는 방식

10

Color Quantization(색상 양자화) + K-means

```
import numpy as np
import cv2

img = cv2.imread('home.jpg')
Z = img.reshape((-1,3))
Z = np.float32(Z) # convert to np.float32

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0) # 반복 종료 조건
K = 8
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)

# Now convert back into uint8, and make original image
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))

cv2.imshow('res2',res2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

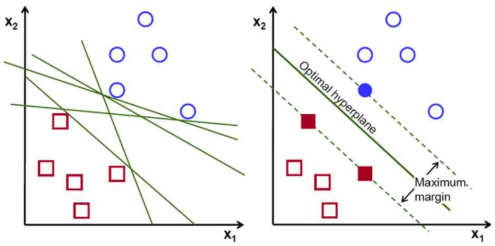
11

SVM + HOG

12

SVM

- SVM(Support Vector Machine)
  - 두 그룹으로 나뉜 학습 데이터를 받아서 두 그룹의 영역으로 나누는 선을 찾는 방식의 분류 알고리즘
  - 데이터와의 간격이 최대가 되는 직선이 최적의 선



13

SVM

- **cv2.ml.SVM_create() -> retval**
  - retval: cv2.ml_SVM 객체

- **cv.ml_SVM.trainAuto(samples, layout, responses) -> retval**
  - samples: 학습 데이터 행렬. numpy.ndarray. shape=(N, d), dtype=numpy.float32.
  - layout: 학습 데이터 배치 방법. cv2.ROW_SAMPLE 또는 cv2.COL_SAMPLE.
  - responses: 각 학습 데이터에 대응되는 응답(레이블) 벡터. numpy.ndarray. shape=(N,) 또는 (N, 1). dtype=numpy.int32 또는 numpy.float32.
  - retval: 학습이 정상적으로 완료되면 True

- **cv.ml_SVM.predict(sample) -> retval, result**
  - sample: 테스트 데이터
  - result : 분류 결과. results[0][0]에 응답(레이블)

14

SVM

```
nsample = 25
a = np.random.randint(0,158,(nsample,2)) # 0~158 구간 임의의 수 nsample x 2 생성
b = np.random.randint(98, 255,(nsample,2)) # 98~255 구간 임의의 수 nsample x 2 생성
traindata = np.vstack((a, b)).astype(np.float32) # a, b를 병합, nsample x 2의 임의의 수 생성
responses = np.zeros((nsample*2, 1), np.int32)
responses[:nsample], responses[nsample:] = 0 , 1

color = [(128, 128, 255), (128, 255, 128)]
color2 = [(0, 0, 128), (0, 128, 0)]
image = np.zeros((255, 255, 3)).astype("uint8")
```

15

SVM

```
svm = cv2.ml.SVM_create()
svm.trainAuto(traindata, cv2.ml.ROW_SAMPLE, responses) # SVM 알고리즘 객체 생성 및 훈련

for y in range(255): # (0,0) ~ (254,254)의 모든 점에 대한 예측
    for x in range(255):
        test = np.array([[x, y]], dtype=np.float32)
        ret, results = svm.predict(test)
        image[y,x] = color[int(results[0][0])]

for i in range(traindata.shape[0]):
    x = int(traindata[i, 0])
    y = int(traindata[i, 1])
    l = responses[i][0]
    cv2.circle(image, (x, y), 3, color2[l], -1, cv2.LINE_AA)
```

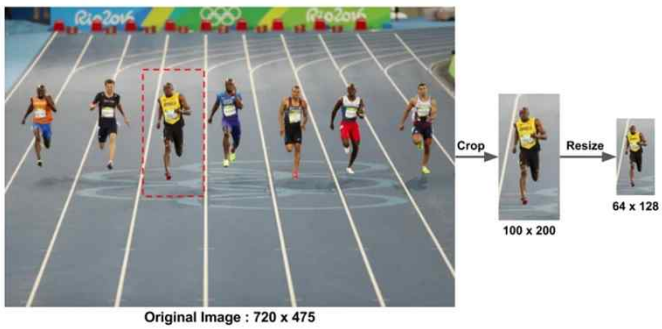
16

HOG

- HOG(Histogram of Oriented Gradient)
  - 방향 그래디언트 히스토그램
  - 서있는 사람 영상 전체적인 형태에 주목하여, 그래디언트를 구하고, 그 크기와 방향 성분을 사용하여 사람이 서있는 형태에 대한 특징 벡터를 추출

① 보행자 검출을 위한 영상은 기본적으로 64x 128 크기의 영상을 사용

② 입력 영상에서 그래디언트를 계산

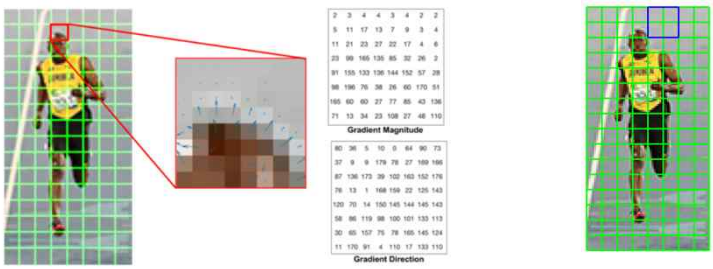


17

HOG

③ 그래디언트 방향 성분은 0~180으로 하고, 입력 영상은 8x8 크기 단위의 실(cell)로 분할

- 보행자 영상(64x 128)은 8x16개의 셀을 가짐
- **각 셀로부터 그래디언트 방향 성분에 대한 히스토그램을 계산**
- **각 방향 성분을 20도 단위로 구분** => 셀마다 9방향으로 분류된 그래디언트 방향 히스토그램을 계산 : 9개의 값



④ 인접한 네 개의 셀을 묶어 블록(block)이라고 함 : 4 × 9 = 36개의 특징값

- 보행자 영상(64x 128 => 8x16셀) 에서 7x15블록이 존재 : 7 × 15 × 36 = 3780개의 특징값

⑤ 블록 단위로 계산된 HOG 특징을 이용하여 보행자인지, 아닌지를 분류 => SVM을 사용

18

HOG

- **cv2.HOGDescriptor() -> retval**
  - retval: cv2.HOGDescriptor 객체
- **cv2.HOGDescriptor.setSVMDetector(svmdetector) -> None**
  - svmdetector: 선형 SVM 분류기를 위한 계수
    - cv2.HOGDescriptor_getDefaultPeopleDetector()
    - cv2.HOGDescriptor_getDaimlerPeopleDetector()
- **cv2.HOGDescriptor.detectMultiScale(img) -> foundLocations, foundWeights**
  - 입력 영상을 조금씩 축소하면서 다양한 크기에 대해서 검출
  - img: 입력 영상. cv2.CV_8UC1 또는 cv2.CV_8UC3.
  - foundLocations: (출력) 검출된 사각형 영역 정보
  - foundWeights: (출력) 검출된 사각형 영역에 대한 신뢰도

19

HOG

```
# default 디텍터를 위한 HOG 객체 생성 및 설정
hogdef = cv2.HOGDescriptor()
hogdef.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cap = cv2.VideoCapture('./images/walking.avi')

print('Toggle Space-bar to change mode.')
while cap.isOpened():
    ret, img = cap.read()
    if not ret: break

   found, _ = hogdef.detectMultiScale(img) # default 디텍터로 보행자 검출
    for (x,y,w,h) in found:
            cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,255))

    cv2.putText(img, 'Detector:%s'%('Default' if mode else 'Daimler'), \
                    (10,50 ), cv2.FONT_HERSHEY_DUPLEX,1, (0,255,0),1)
    cv2.imshow('frame', img)
    key = cv2.waitKey(1)
```

20