KNN

양유진

한성대학교 HANSUNG UNIVERSITY

유튜브 주소: https://youtu.be/LTKK_bJ3FIM

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

01 KNN이란?

KNN 정의

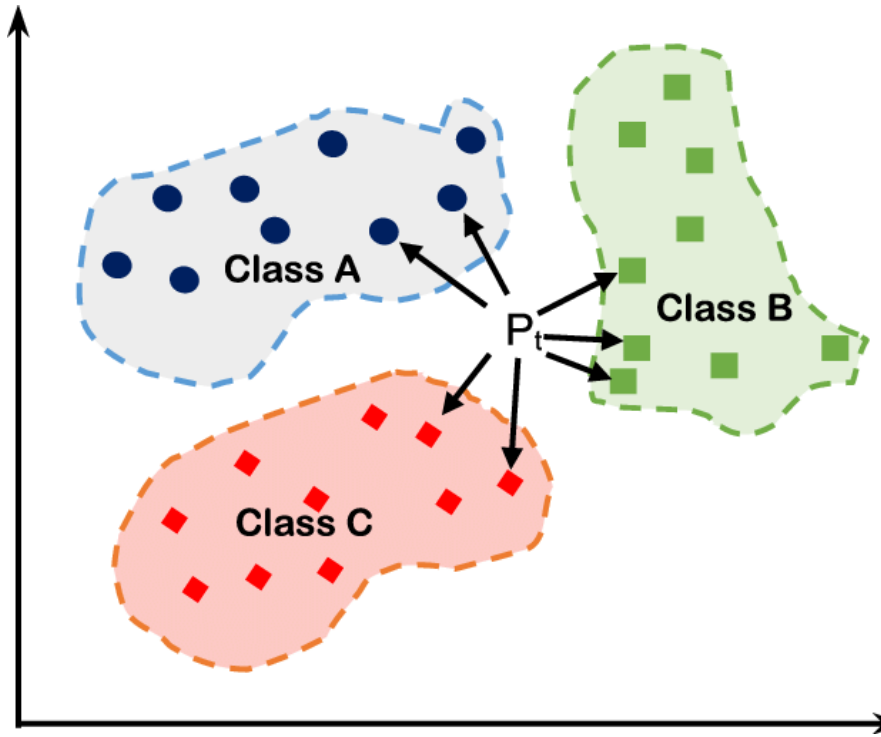
KNN(K-Nearest Neighbor) : K - 최근접 이웃법

- 지도 학습 중 '분류'에 속하는 알고리즘
- 새로운 데이터가 들어왔을 때, 데이터 그룹 중 어떤 그룹에 속하는지 분류해주는 알고리즘
- 새로운 데이터와 기존 데이터 비교하여 가장 유사하게 판단된 기존 데이터로 분류함.
- Ex) 이미지 분류, 음악 장르 분류 등

KNN 단계

[1단계] 새로운 데이터와 가장 인접한 데이터 k개 선정

[2단계] k개의 데이터가 가장 많이 속한 분류 선택



(출처: https://www.researchgate.net/figure/Example-on-KNN-classifier_fig1_331424423)

1) k 값 너무 작은 경우:

데이터 하나의 영향력 높아짐
→ overfitting 발생

2) k 값 너무 큰 경우:

분류 경계면 단조로워짐
→ underfitting 발생

KNN 장/단점

[장점]

1. 보통 **높은 정확도**
2. 가장 가까운 상위 k개의 데이터만 활용

→ 오류 데이터 비교 대상에서 제외

→ **오류 데이터**가 **결과**에 **영향 끼치지 않음**

3. 기존 데이터 기반 → **데이터에 대한 가정 없음**
→ 구현 간단함

[단점]

1. 새로운 데이터와 기존의 모든 데이터 비교

→ 데이터 많을수록 처리시간 증가

→ **속도 느려짐**

2. 데이터량과 메모리 사용량 비례

→ **고용량 메모리 필요**로 함.

3. 차원(벡터) 크기 큰 경우, 변수간 거리 멀어짐

→ **정확도 감소**

02 실습(1)

실습환경

1. 아나콘다(Anaconda) prompt 설치링크

2. Python (3.9.5) 버전변경

3. 설치해야 할 모듈 설명 링크

conda install [모듈이름]

face_recognition [필수]

4. Github 코드 (examples)

사진 속 얼굴 찾기 (1) 코드

```python
from PIL import Image
import face_recognition

# 이미지 파일 숫자 배열(numpy)로 가져옴
image = face_recognition.load_image_file("img01.jpg")  이미지 로드(숫자배열로)

'''
기본 HOG 기반 모델을 사용하여 이미지에서 모든 면을 찾는다.

*HOG 기반 모델이란? (Histogram of Oriented Gradient)
: 대상 영역을 일정 크기의 셀로 분할하고, 각 셀마다 edge 픽셀(gradient magnitude가 일정 값 이상인 픽셀)들의
방향에 대한 히스토그램을 구한 후 이들 히스토그램 bin 값들을 일렬로 연결한 벡터이다.

이 방법은 꽤 정확하지만 GPU를 사용하지 않고, CNN 모델만큼 정확하진 않다.
cnn모델을 사용한 코드: find_faces_in_picture_cnn.py
'''

face_locations = face_recognition.face_locations(image)  얼굴 위치값 찾기

print("사진에서 얼굴이 {}개 발견되었습니다.".format(len(face_locations)))

for face_location in face_locations:

    # 이미지의 각 얼굴 위치값 출력
    top, right, bottom, left = face_location
    print("얼굴 픽셀 위치 Top: {}, Left: {}, Bottom: {}, Right: {}".format(top, left, bottom, right))

    # You can access the actual face itself like this:
    face_image = image[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)  Numpy배열 → 이미지로 변환
    pil_image.show() #화면에 보여줌
```

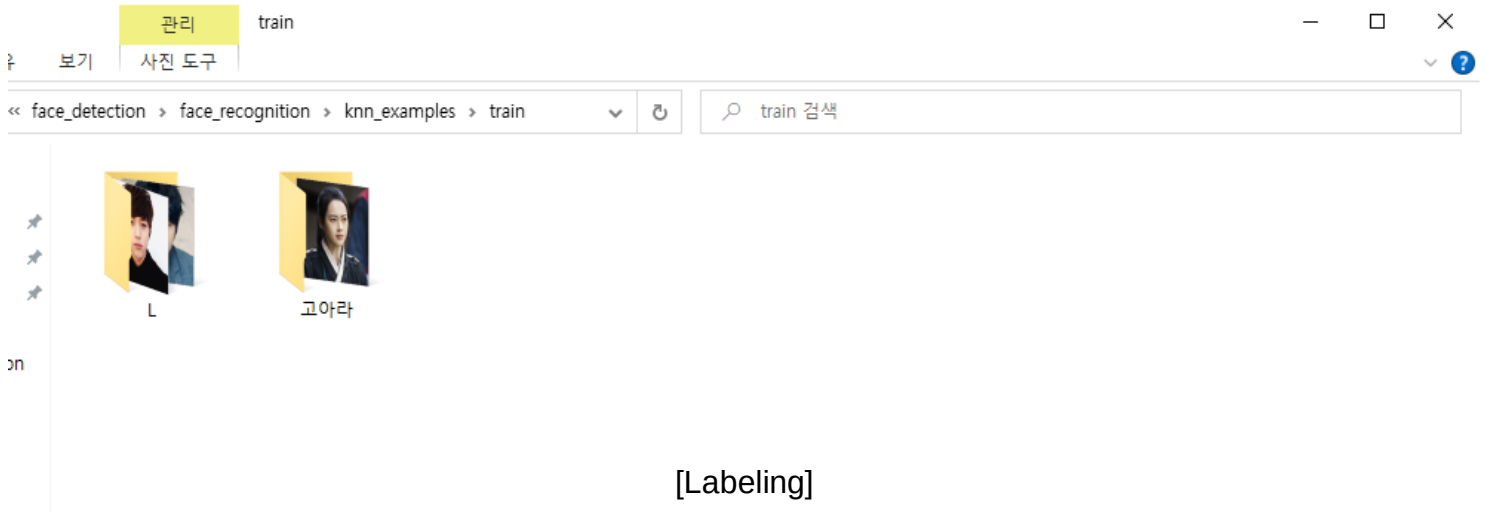
사진 속 얼굴 찾기 (2) 결과

[img01.jpg]

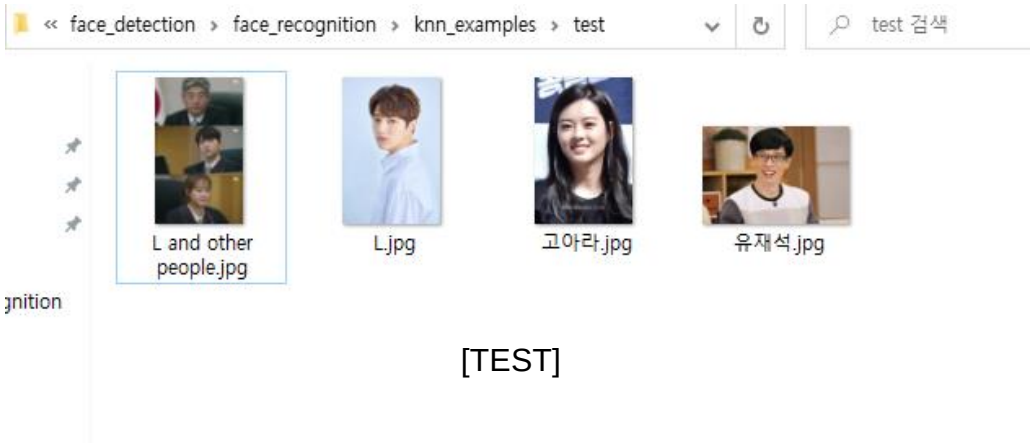
```
사진에서 얼굴이 5개 발견되었습니다.
얼굴 픽셀 위치 Top: 105, Left: 254, Bottom: 179, Right: 328
얼굴 픽셀 위치 Top: 80, Left: 113, Bottom: 155, Right: 187
얼굴 픽셀 위치 Top: 211, Left: 315, Bottom: 274, Right: 377
얼굴 픽셀 위치 Top: 122, Left: 432, Bottom: 184, Right: 494
얼굴 픽셀 위치 Top: 121, Left: 519, Bottom: 196, Right: 594
```

03 실습(2)

사진 속 얼굴 분류 (KNN 모델사용) (1) Labeling&코드_import



[Labeling]



[TEST]

```
import math
from sklearn import neighbors
import os
import os.path
import pickle
from PIL import Image, ImageDraw, ImageFont
import face_recognition
from face_recognition.face_recognition_cli import image_files_in_folder

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
```

사진 속 얼굴 분류 (1) 코드_train함수

```
def train(train_dir, model_save_path=None, n_neighbors=None, knn_algo='ball_tree', verbose=False):
    훈련 자료가 있는 디렉토리   knn 모델 경로   K값: 가중치 부여할 인접값 수
    X = []
    y = []

    # 훈련셋에 있는 사람들 훈련함. Loop through each person in the training set
    for class_dir in os.listdir(train_dir):return: 훈련 디렉토리 내의 파일 이름
        if not os.path.isdir(os.path.join(train_dir, class_dir)):경로 생성(train_dir/class_dir)
            continue
```

사진 속 얼굴 분류 (1) 코드_train함수

[문자].(jpg/jpeg/png) 형태 찾으면 그 이름 반환

```python
def image_files_in_folder(folder):
    return [os.path.join(folder, f) for f in os.listdir(folder) if re.match(r'.*\.(jpg|jpeg|png)', f, flags=re.I)]
```

face_recognition_cli.py

```python
        #현재 사용자에 대한 각 훈련 이미지 반복 Loop through each training image for the current person
        for img_path in image_files_in_folder(os.path.join(train_dir, class_dir)):
            image = face_recognition.load_image_file(img_path)
            face_bounding_boxes = face_recognition.face_locations(image)

            if len(face_bounding_boxes) != 1:
                # If there are no people (or too many people) in a training image, skip the image.
                if verbose:
                    print("Image {} not suitable for training: {}".format(img_path, "Didn't find a face"
                        if len(face_bounding_boxes) < 1 else "Found more than one face"))
            else:
                # Add face encoding for current image to the training set
                X.append(face_recognition.face_encodings(image, known_face_locations=face_bounding_boxes)[0])
                y.append(class_dir)
```

얼굴 위치값(top, right, bottom, left) 반환

```python
def load_image_file(file, mode='RGB'):
    im = PIL.Image.open(file)
    if mode:
        im = im.convert(mode)
    return np.array(im)
```

Open한 이미지 파일을 numpy array 형태로 return

api.py

사진 속 얼굴 분류 (1) 코드_train함수

```python
    # KNN 분류기의 가중치에서 사용한 인접 항목수 결정 Determine how many neighbors to use for weighting in the KNN classifier
    if n_neighbors is None:
        n_neighbors = int(round(math.sqrt(len(X))))
        if verbose:
            print("Chose n_neighbors automatically:", n_neighbors)

    # KNN 분류기 생성 및 훈련 Create and train the KNN classifier
    knn_clf = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors, algorithm=knn_algo, weights='distance')
    knn_clf.fit(X, y)

    # Save the trained KNN classifier
    if model_save_path is not None:
        with open(model_save_path, 'wb') as f:
            pickle.dump(knn_clf, f)

    return knn_clf
```

리스트X 길이의 제곱근 반올림한 값(정수형)

KNN 분류기 생성

KNN 분류기 훈련

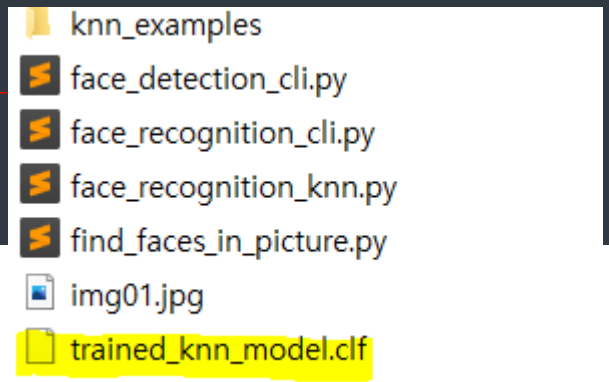


사진 속 얼굴 분류 (2) 코드_predict함수

테스트 셋 경로 knn 분류기 객체 훈련된 KNN 모델 경로 얼굴 분류 임계값 값이 클수록 잘못 분류할 가능성이 커짐

```python
def predict(X_img_path, knn_clf=None, model_path=None, distance_threshold=0.6):

    if not os.path.isfile(X_img_path) or os.path.splitext(X_img_path)[1][1:] not in ALLOWED_EXTENSIONS:
        raise Exception("Invalid image path: {}".format(X_img_path))

    if knn_clf is None and model_path is None:
        raise Exception("Must supply knn classifier either thourgh knn_clf or model_path")

    # Load a trained KNN model (if one was passed in)
    if knn_clf is None:
        with open(model_path, 'rb') as f:
            knn_clf = pickle.load(f)
```

사진 속 얼굴 분류 (2) 코드_predict함수

```python
    # 파일에서 이미지 파일 로드하고 얼굴 위치값 찾음. Load image file and find face locations
    X_img = face_recognition.load_image_file(X_img_path)
    X_face_locations = face_recognition.face_locations(X_img)

    # 만약 이미지에서 얼굴이 발견되지 않으면 빈 결과값이 반환됨. If no faces are found in the image, return an empty result.
    if len(X_face_locations) == 0:
        return []

    # 테스트 이미지에서 얼굴들에 대한 인코딩 찾음. Find encodings for faces in the test iamge
    faces_encodings = face_recognition.face_encodings(X_img, known_face_locations=X_face_locations)

    # KNN 모델 사용하여 테스트 얼굴에서 가장 적합한 일치 항목(이름)을 찾음. Use the KNN model to find the best matches for the test face
    closest_distances = knn_clf.kneighbors(faces_encodings, n_neighbors=1)
    are_matches = [closest_distances[0][i][0] <= distance_threshold for i in range(len(X_face_locations))]

    # 클래스 예측하고 임계값 안에 없는 분류 제거. Predict classes and remove classifications that aren't within the threshold
    return [(pred, loc) if rec else ("unknown", loc) for pred, loc, rec in zip(knn_clf.predict(faces_encodings), X_face_locations, are_matches)]
```

샘플 거리 계산. index 반환함

```
8004 95e3 1800 0000 0000 008c 2173 6b6c
6561 726e 2e6e 6569 6768 626f 7273 2e5f
636c 6173 7369 6669 6361 7469 6f6e 948c
144b 4e65 6967 6862 6f72 7343 6c61 7373
6966 6965 7294 9394 2981 947d 9428 8c0b
6e5f 6e65 6967 6862 6f72 7394 4b02 8c06
7261 6469 7573 944e 8c09 616c 676f 7269
7468 6d94 8c09 6261 6c6c 5f74 7265 6594
```

trained_knn_model.clf 中 일부

사진 속 얼굴 분류 (3) 코드_show_prediction_labels_on_image 함수

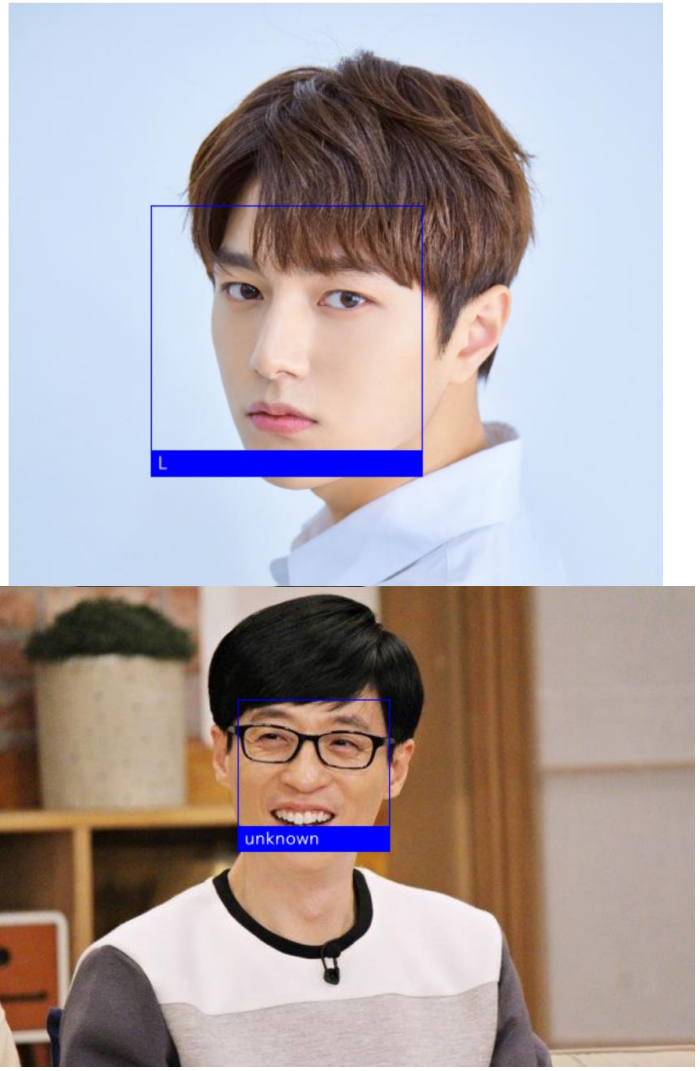
(test 데이터 경로, 예측 결과)

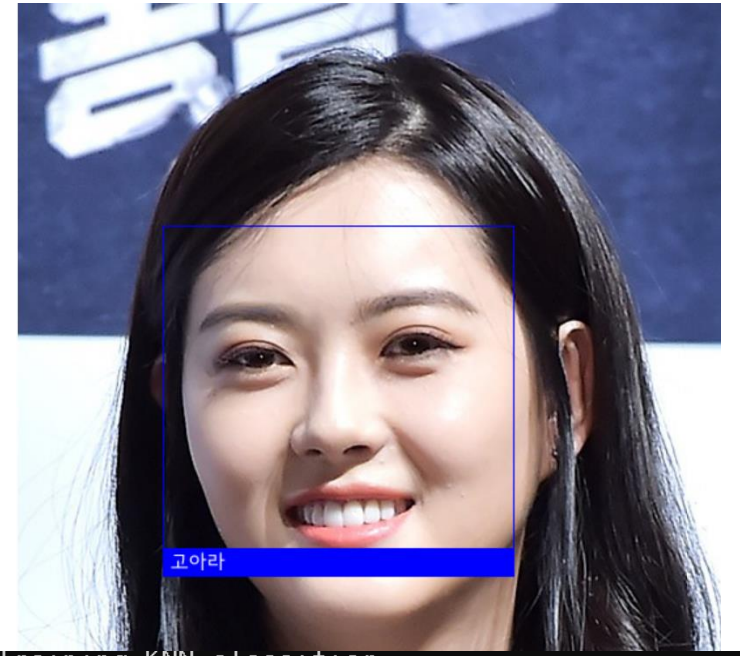
```python
def show_prediction_labels_on_image(img_path, predictions):

    pil_image = Image.open(img_path).convert("RGB")
    draw = ImageDraw.Draw(pil_image)
    font = ImageFont.truetype("fonts/SeoulNamsanM.ttf", 15) #한글 표현할 수 있는 폰트 사용해야 깨지지 않음.

    for name, (top, right, bottom, left) in predictions:
        # Pillow 모듈 사용하여 얼굴에 box 그려줌 Draw a box around the face using the Pillow module
        draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))

        # 한글 출력시에는 utf-8로 변환 안해줘도 됨.
        # name = name.encode("UTF-8")

        # 얼굴 밑에 이름(레이블) 달아줌 Draw a label with a name below the face
        text_width, text_height = draw.textsize(name)
        draw.rectangle(((left, bottom - text_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))
        draw.text((left+6, bottom - text_height - 8), name, font=font, fill=(255, 255, 255, 255))

    # Pillow 문서에 따라 메모리에서 그려주는 라이브러리 제거 Remove the drawing library from memory as per the Pillow docs
    del draw

    # 이미지 결과 보여줌 Display the resulting image
    pil_image.show()
```

사진 속 얼굴 분류 (4) 코드_main

```
if __name__ == "__main__":
    print("Training KNN classifier...")
    classifier = train("knn_examples/train", model_save_path="trained_knn_model.clf", n_neighbors=2)
    print("Training complete!")

    # STEP 2: 훈련된 분류기 사용하여 unkown 이미지 예측하기
    for image_file in os.listdir("knn_examples/test"):
        full_file_path = os.path.join("knn_examples/test", image_file)

        print("Looking for faces in {}".format(image_file))

        # 훈련된 분류기 모델을 사용하여 이미지에서 모든 사람 찾기
        # 분류기 파일 이름 또는 분류기 모델 인스턴스를 전달할 수 있음.
        predictions = predict(full_file_path, model_path="trained_knn_model.clf")

        # 콘솔에 결과 출력. Print results on the console
        for name, (top, right, bottom, left) in predictions:
            print("- Found {} at ({}, {})".format(name, left, top))

        # 이미지에 결과 중첩하여 표시함. Display results overlaid on an image
        show_prediction_labels_on_image(os.path.join("knn_examples/test", image_file), predictions)
```

사진 속 얼굴 분류 (모델사용O) (2) 결과






```
Training KNN classifier...
Training complete!
Looking for faces in L and other people.jpg
- Found 고아라 at (239, 655)
- Found unknown at (184, 81)
- Found L at (253, 354)
Looking for faces in L.jpg
- Found L at (117, 167)
Looking for faces in 고아라.jpg
- Found 고아라 at (111, 171)
Looking for faces in 유재석.jpg
- Found unknown at (225, 96)
```

감사합니다