CHAPTER 10
영상 분할 및 특징 처리

10.3 K-최근접 이웃 분류기

1

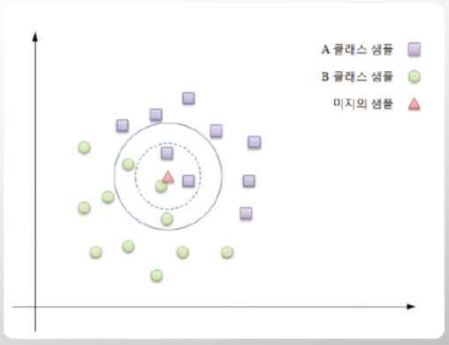
10.3.1 k-최근접 이웃 분류기의 이해

- 최근접 이웃 알고리즘
  - 기존에 가지고 있는 데이터들을 일정한 규칙에 의해 분류된 상태에서 새로운 입력 데이터의 종류를 예측하는 분류 알고리즘
  - 학습 클래스의 샘플들과 새 샘플의 거리가 가장 가까운(nearest)클래스로 분류
    - '가장 가까운 거리'
      - 미지의 샘플과 학습 클래스 샘플간의 유사도가 가장 높은 것을 의미
      - 유클리드 거리(euclidean distance), 해밍 거리(hamming distance),차분 절대값

2

10.3.1 k-최근접 이웃 분류기의 이해

- k-최근접 이웃 분류(k-Nearest Neighbors: k-NN)
  - 학습된 클래스들에서 여러 개(k)의 가까운 이웃을 선출하고 이를 이용하여 미지의 샘플들을 분류하는 방법
  - k가 3일 경우
    - 미지 샘플 주변 가장 가까운 이웃 3개 선출
    - 이 중 많은 수의 샘플을 가진 클래스로 미지의 샘플 분류
    - A클래스 샘플 2개, B클래스 샘플 1개 → A클래스 분류
  - k가 5일 경우
    - 실선 큰 원내에 있는 가장 가까운 이웃 5개 선출
    - 2개 A 클래스, 3개 B 클래스 → B클래스로 분류



3

10.3.2 k-NN을 위한 KNearest 클래스의 이해

예제 10.3.1 임의 좌표 생성 – 04.kNN_exam.py

```
01  import numpy as np, cv2
02
03  def draw_points(image, group, color):
04      for p in group:
05          pt = tuple(p.astype(int))                    # 정수 원소 튜플
06          cv2.circle(image, pt, 3, color, cv2.FILLED)
07
08  nsample = 50                                         # 그룹당 학습 데이터 수
09  traindata = np.zeros((nsample*2, 2), np.float32)     # 전체 학습 데이터 행렬
10  label = np.zeros((nsample*2, 1), np.float32)         # 레이블 행렬 생성
11
12  cv2.randn(traindata[:nsample], 150, 30)              # 정규분포 랜덤 값 생성
13  cv2.randn(traindata[nsample:], 250, 60)
14  label[:nsample], label[nsample:] = 0, 1              # 레이블 기준값 지정
15
16  K = 7
17  knn = cv2.ml.KNearest_create()                       # kNN 클래스로 객체 생성
18  knn.train(traindata, cv2.ml.ROW_SAMPLE, classlable)  # 학습 수행
19
```

반지름 3의 원 표시 – 점으로 표시

반환 행렬

행렬 원소에 정규분포를 따르는 임의값 지정

두 그룹에 레이블 값 지정

4

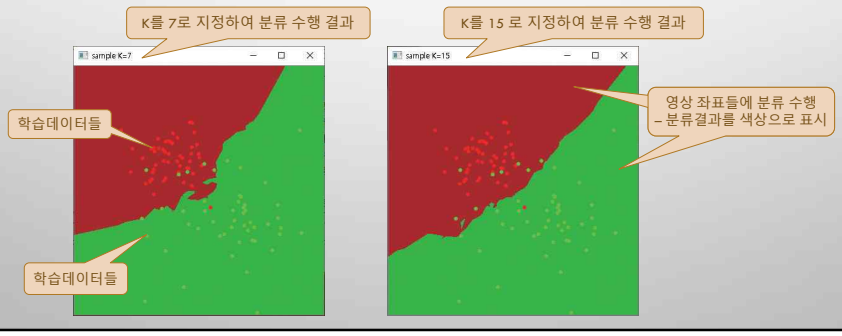
10.3.2 k-NN을 위한 KNearest 클래스의 이해

```
20  points = [(x, y) for y in range(400) for x in range(400) ]      # 검사 좌표 리스트 생성
21  ret, resp, neig, dist = knn.findNearest(np.array(points, np.float32), K)   # 분류 수행
22
23  colors = [(0, 180, 0) if p else (0, 0, 180) for p in resp]
24  image = np.reshape(colors, (400, 400, 3)).astype('uint8')        # 3채널 컬러
25
26  draw_points(image, traindata[:nsample], color=(0, 0, 255))
27  draw_points(image, traindata[nsample:], color=(0, 255, 0))
28  cv2.imshow("sample K="+ str(K), image)
29  cv2.waitKey(0)
```

분류된 좌표들에 색지정

좌표들을 400x400 크기 영상으로 형태 변경

- 실행결과



5

10.3.3 MNIST 데이터 사용

- MNIST(Modified National Institute of Standards and Technology) 데이터셋
  - 필기 숫자 영상으로 구성된 대형 데이터베이스
  - 다양한 영상처리 시스템을 학습하기 위해 일반적으로 사용
  - 다운로드 http://deeplearning.net/data/mnist/mnist.pkl.gz

6

10.3.3 MNIST 데이터 사용

예제 10.3.2 MNIST 데이터 다운로드 및 시각화 – 05.mnist_kNN_train.py

```
01 import cv2, numpy as np
02 import pickle, gzip, os
03 from urllib.request import urlretrieve                # 웹사이트 링크 다운로드 함수
04 import matplotlib.pyplot as plt
05
06 def load_mnist(filename):                             # MNIST 데이터셋 다운로드 함수
07     if not os.path.exists(filename):                  # 현재 폴더에 파일 없으면 다운
08         print("Downloading")
09         link = "http://deeplearning.net/data/mnist/mnist.pkl.gz"
10         urlretrieve(link, filename)                   # 다운로드
11     with gzip.open(filename, 'rb') as f:
12         return pickle.load(f, encoding='latin1')      # pickle 모듈로 파일에서 로드함
13
14 def graph_image(data, lable, title, nsample):        # 데이터 시각화 함수
15     plt.figure(num=title, figsize=(10, 10))
16     rand_idx = np.random.choice(range(data.shape[0]), nsample)  # 데이터 번호 랜덤 생성
17     for i, id in enumerate(rand_idx):
18         img = data[id].reshape(28, 28)                # 1행 행렬을 영상 행태로 변경
19         plt.subplot(6, 4, i+1), plt.axis('off'), plt.imshow(img, cmap='gray')
20         plt.title("%s: %d" % (title, lable[id]))      # 서브플롯 타이틀
21         plt.tight_layout()
22
```

압축해제위한 모듈

mnist 데이터셋 다운주소

전체 데이터중 랜덤하게 nsample개 선택

7

10.3.3 MNIST 데이터 사용

오타

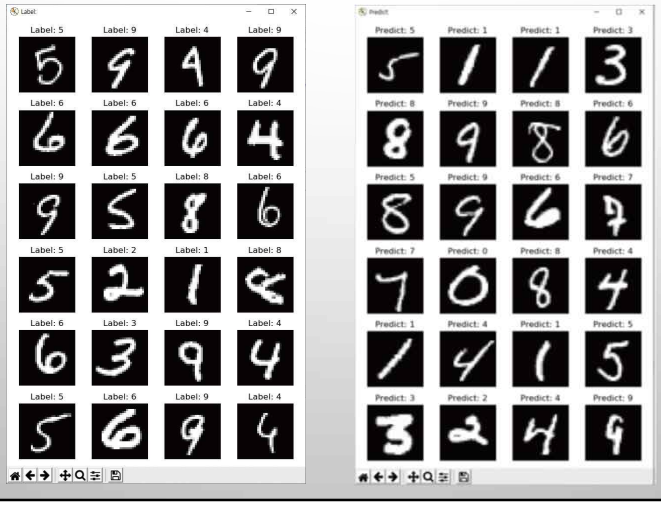
```
23 train_set, valid_set, test_set = load_mnist("mnist.pkl.gz") # MNIST 데이터셋 다운
24 train_data, train_label = train_set
25 test_data, test_label = test_set
26 ## MNIST 로드 데이터 크기 확인
27 print("train_set=", train_set[0].shape)              # 학습 데이터셋
28 print("valid_set=", valid_set[0].shape)              # 검증 데이터셋
29 print("test_set=", test_set[0].shape)                # 테스트 데이터셋
30
31 print("training...")
32 knn = cv2.ml.KNearest_create()
33 knn.train(train_data, cv2.ml.ROW_SAMPLE, train_label)  # k-NN 학습 수행
34
35 nsample = 100
36 print("%d 개 predicting..." % nsample )
37 _, resp, _ , _ = knn.findNearest(test_data[:nsample], k=5)      # k-NN 분류 수행
38 accur = sum(resp.flatten() == test_label[:nsample])             # 분류 정확도 측정
39
40 print('정확도=', accur / nsample * 100, '%')
41 graph_image(train_data, train_label, 'label', 24)     # 데이터 영상으로 그리기
42 graph_image(test_data[:nsample], resp, 'predict', 24)
43 plt.show()                                            # 2개 그림 동시에 열기
```

8

10.3.3 MNIST 데이터 사용

- 실행결과

```
Run: 05.mnist_kNN_train
C:\Python\python.exe D:/source/chap10/05.mnist_kNN_train.py
train_set= (50000, 784)
valid_set (10000, 784)
test_set (10000, 784)
training...
100개 predicting...
정확도= 98.0 %
```



9

단원 요약

- 최근접 이웃 알고리즘은 기존에 가지고 있는 데이터들을 일정한 규칙에 의해 분류된 상태에서 새로운 데이터의 종류를 예측하는 분류 알고리즘이다. 새로운 미지의 샘플이 입력될 때, 학습 클래스의 샘플들과 새 샘플의 거리가 가장 가까운 클래스로 분류한다.

- 최근접 이웃 방법 중에서 가장 많이 사용되는 것은 학습된 클래스들에서 여러 개의 가까운 이웃을 선출하고 이를 이용하여 미지의 샘플들을 분류하는 방법이다. 이러한 분류 과정을 k-최근접 이웃 분류라고 한다.

- OpenCV에서는 최근접 이웃 분류를 위해 ml 라이브러리의 KNearest_create() 함수로 kNN 객체(ml_KNearest)를 만들어서 객체 내부 메서드인 train()로 학습을 한다. 또한, findNearest() 메서드로 분류를 수행한다.

10