

Open_cv07

OpenCV의 ML

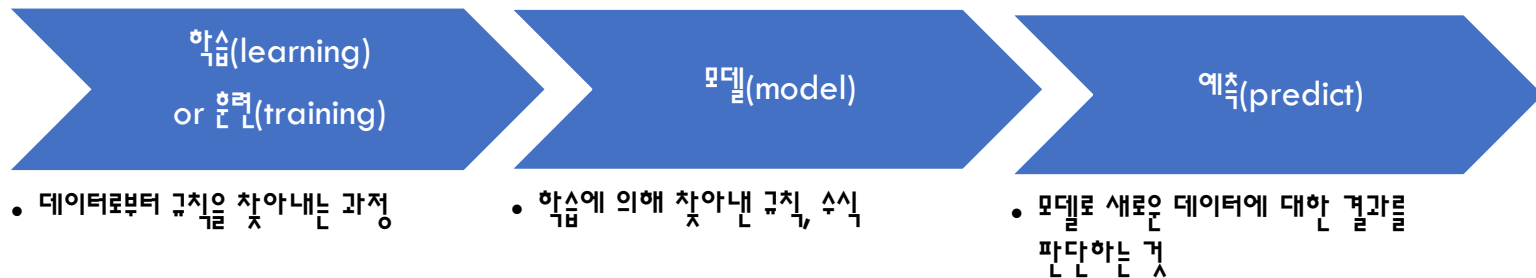
[ICT , DS&DE] Y-A , KIM

ML

- ML의 2가지 모드

- 학습모드 : 내부함수를 규칙에 맞게 재정리하는 모드

- 예측모드 : 존재하는 내부 함수로 결과를 판단하는 모드



Training set: 10개의 사진을 분류하는 data set

- labeled data 가 없는 데이터 사용
- Google news 그룹핑
- Word clustering

- 지도 학습의 종류

예측 알고리즘

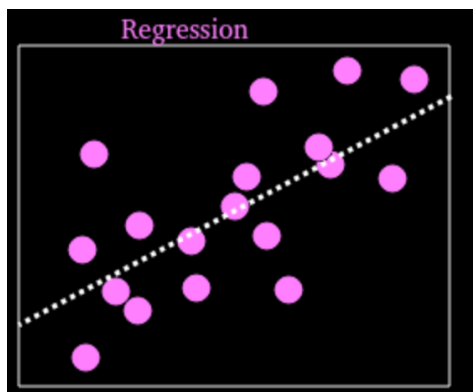
- 회귀 알고리즘 (regression)

이진 분류 알고리즘

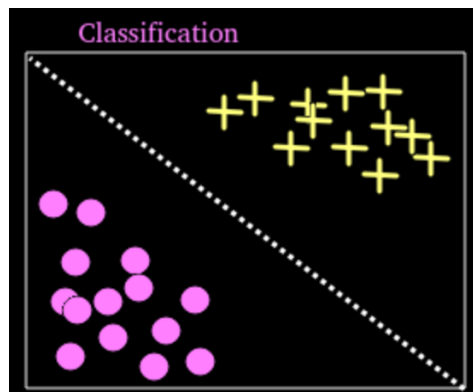
- 이진 분류 알고리즘 (binary classification)

다중 분류 알고리즘













- 다중 분류 알고리즘 (multi-label classification)

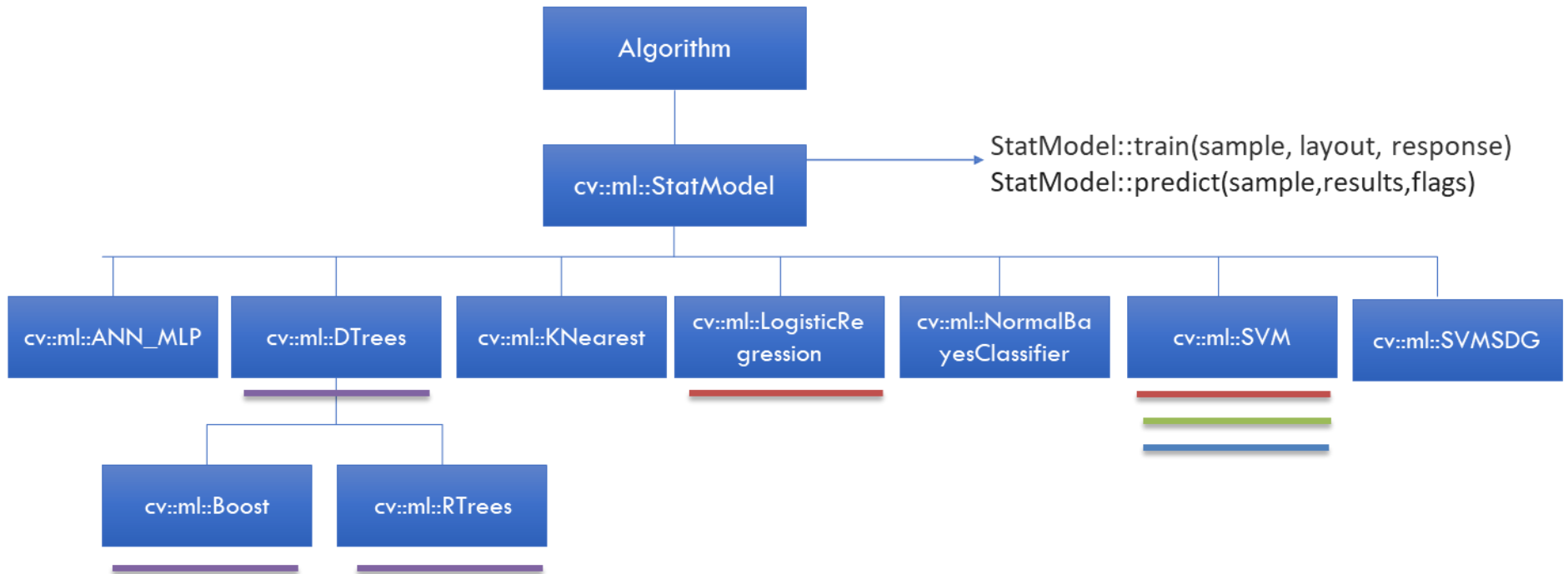


연산량 / MSE, RMSE



레이블/정확도

	Multi-Class			Multi-Label		
	Samples			Samples		
C = 3						
						
						
	Labels			Labels		
	[100]	[010]	[001]	[110]	[011]	[111]



- LB:Linear Binary
- LM:Linear Multiclass
- NB:Non-linear Binary
- NB:Non-linear Multiclass

OpenCV 머신러닝 클래스 종류

분류	설명	함수/메서드
ANN_MLP	인공 신경망(Artificial Neural Network), 다층 퍼셉트론(Multi-layer perceptron)을 의미	cv2.ml.ANN_MLP_create()
DTrees	결정 트리(decision trees)를 의미	cv2.ml.DTrees_create()
Boost	부스팅(boosting) 방법론으로, 약한 분류기(weak classifier)를 조합하여 강한 분류기를 생성	cv2.ml.Boost_create()
RTrees	랜덤 트리(random tree)나 랜덤 포레스트(random forest)를 의미	cv2.ml.RTrees_create()
EM	기대값 최대화(expectation Maximization)로, 가우시안 혼합 모델(Gaussian mixture model)을 의미	cv2.ml.EM_create()
KNearest	k 최근접 이웃(k-Nearest Neighbor) 방법론을 의미	cv2.ml.KNearest_create()
LogisticRegression	로지스틱 회귀. 이진 분류 알고리즘을 의미	cv2.ml.LogisticRegression_create()
NormalBayesClassifier	베이즈 분류기를 통한 분류 방법론을 의미	cv2.ml.NormalBayesClassifier_create()
SVM	서포트 벡터 머신(Support Vector Machine) 방법론을 의미	cv2.ml.SVM_create()
SVMSGD	확률적 경사하강법(Stochastic Gradient Descent) 방법론을 사용한 SVM을 의미	cv2.ml.SVMSGD_create()

선형 이진 분류 (Linear Binary Classification)

- 사례: 고양이와 개를 분류하는 문제로 주어진 이미지 특성을 기반으로 고양이 또는 개로 분류
- 분류기: Logistic Regression은 이러한 문제에 사용될 수 있는 선형 이진 분류기 예

선형 다중 분류 (Linear Multi-Class Classification):

- 사례: 여러 종류의 동물 (고양이, 개, 말, 코끼리 등)을 분류하는 문제로 주어진 이미지 특성을 기반으로 여러 동물 종류 중 하나로 분류
- 분류기: Softmax Regression (또는 Multinomial Logistic Regression)은 이러한 문제에 사용될 수 있는 선형 다중 분류기의 예

비선형 이진 분류 (Non-linear Binary Classification):

- 사례: 복잡한 텍스처나 패턴을 가진 두 종류의 이미지 (예: 자연 스냅샷과 도시 스냅샷)를 분류하는 문제
- 분류기: Support Vector Machine (SVM) with a non-linear kernel (예: RBF kernel)은 비선형 이진 분류의 예

비선형 다중 분류 (Non-linear Multi-Class Classification):

- 사례: 다양한 종류의 복잡한 텍스처와 패턴을 가진 여러 종류의 이미지 (예: 여러 종류의 과일 이미지)를 분류하는 문제.
- 분류기: Neural Networks (Deep Learning) 또는 SVM with a non-linear kernel과 One-vs-Rest 또는 One-vs-One 전략은 이러한 복잡한 문제에 사용될 수 있는 비선형 다중 분류기의 예

모형최적화 및 조정 방법

하이퍼파라미터 조정

격자 탐색(grid search), 임의 탐색(random search) 등의 방법으로 최적의 하이퍼파라미터 탐색

특징 선택 및 추출

중요한 특징만 선택하거나 새로운 특징 생성하여 모형 성능 향상

앙상블 방법

여러 모형의 예측을 결합하여 성능 향상

최근접 이웃 분류

- k -최근접 이웃 분류기의 이해
- k -NN을 위한 $KNearest$ 클래스의 이해
- k -NN 응용

k-최근접 이웃 분류기의 이해

- 최근접 이웃 알고리즘

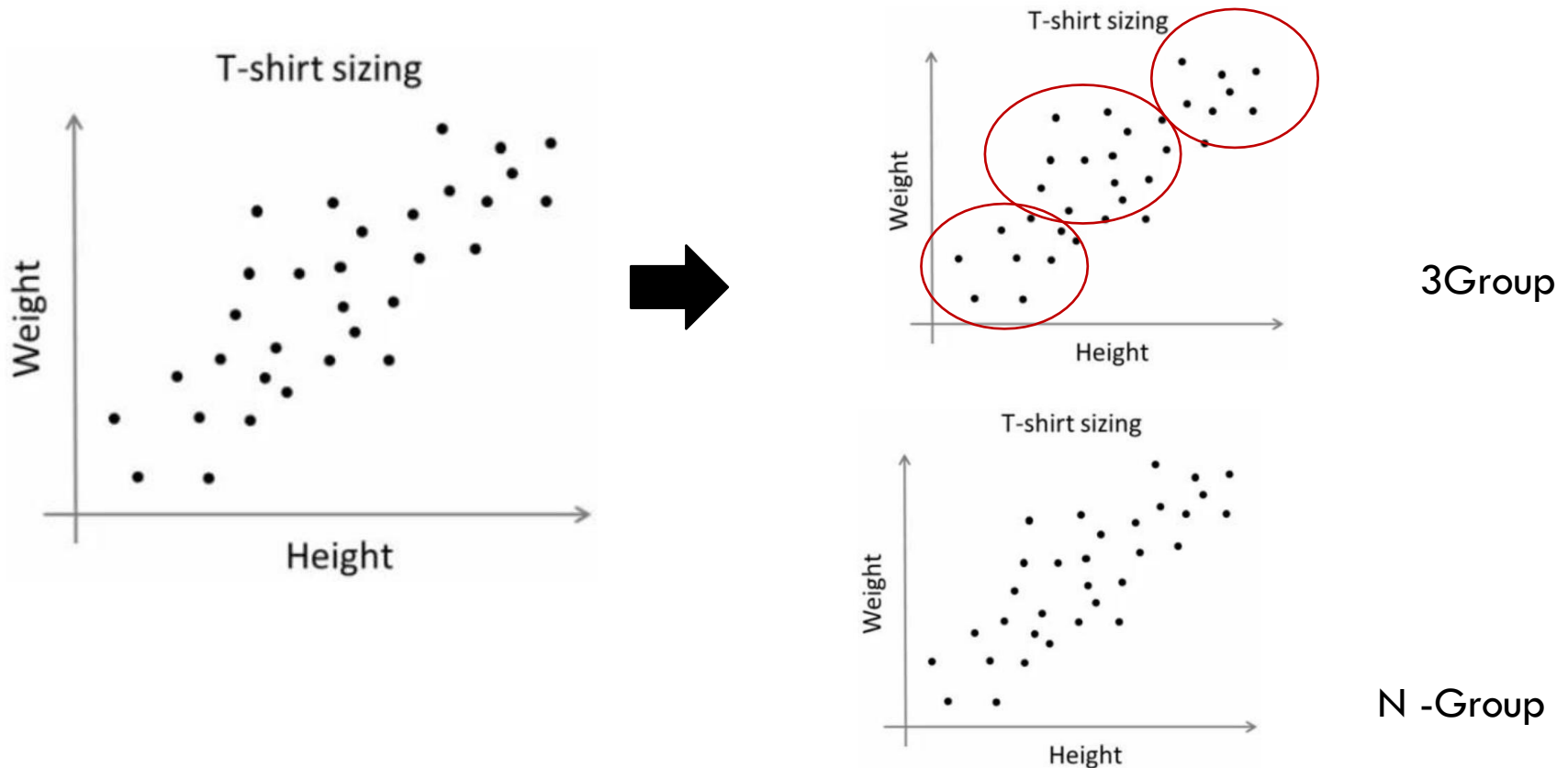
- 기존에 가지고 있던 데이터들을 일정한 규칙에 의해 분류된 상태에서 새로운 입력 데이터의 종류를 예측하는 분류 알고리즘

- 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

h-최적점 이온 분류기의 이해

- 티셔츠 모델을 생각해 보자.

- 제품을 판매하는 회사는 모든 크기의 사람들을 만족시키는 다양한 크기의 모델을 생산할 것이다.
- 사람들의 신장과 체중 데이터를 만들고 플로팅 후 그룹화한다.



k-최근접 이웃 분류기의 이해

- 알고리즘 작동 ? - 반복적인 그룹핑에 대한 반복 작업으로 진행

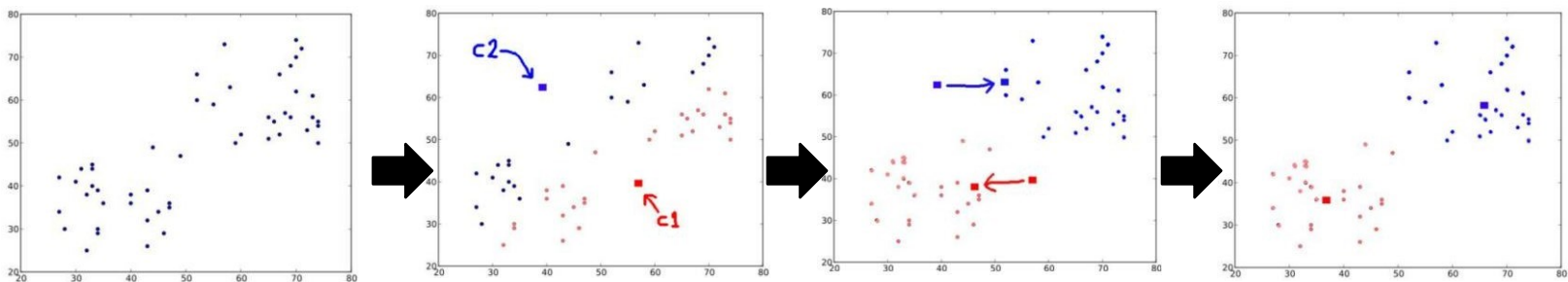
Step01: 무작위로 2개의 점 C1을 C2 생성하여 무게 중심 랜덤선택

Step02: 데이터 라벨링 c1으로 가까우면 0과 빨강, c2로 가까우면 1과 파랑

Step03: 전체 빨강과 파랑점의 좌표의 평균치를 계산해, 새로운 무게 중심지정, c1, c2가 무게 중심으로 이동

Step04: 양쪽의 중심이 특정한 점을 포함 해서 분류 될 때 까지 새로 계산 된 무게 중심을 사용하여 [Step 2]를 다시 계산하고 모든 데이터에 '0'또는 '1'레이블을 지정 한다.

*최대 반복 횟수나 특정의 정밀도를 종료 조건으로서 반복 처리를 정지할 수도 있다.



$$\text{minimize } \left[J = \sum_{\text{All Red_points}} \text{distance}(C1, \text{Red_Point}) + \sum_{\text{All Blue_Points}} \text{distance}(C2, \text{Blue_Point}) \right]$$

OpenCV의 K-Means ?

kmeans(data, K, bestLabels, criteria, attempts, flags[, centers])

-> retval, bestLabels, centers

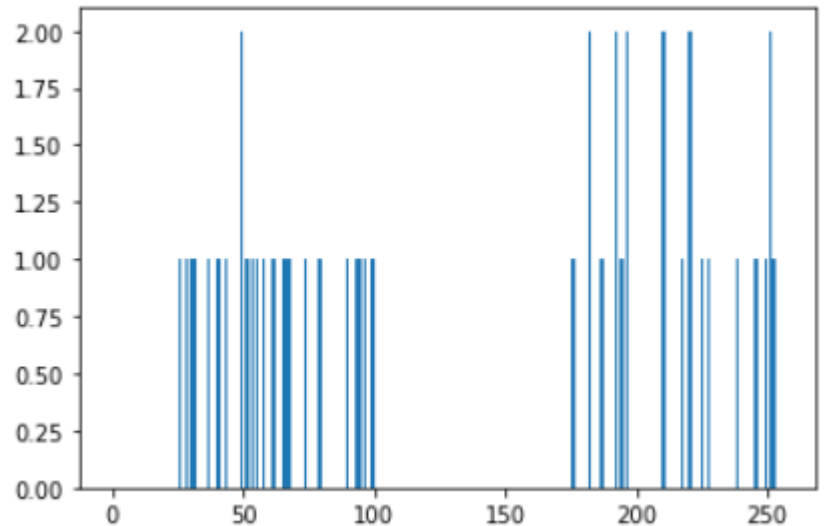
입력 파라미터	<ul style="list-style-type: none">data : np.float32 형의 데이터Nclusters (K) : 궁극적으로 필요한 클러스터 수.criteria : 반복 처리의 종료 조건조건 cv2.TERM_CRITERIA_EPS - 지정된 정밀도 (epsilon)에 도달하면 반복 계산을 종료 cv2.TERM_CRITERIA_MAX_ITER - 지정된 반복 횟수 (max_iter)에 도달하면 반복 계산을 종료 cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER - 위 조건 중 하나가 충족되면 반복 계산을 종료attempts : 다른 초기 라벨링을 사용하여 알고리즘을 실행하는 시도 횟수를 나타내는 플래그로 레이블을 반환Flags : 무게 중심의 초기 값을 결정하는 방법을 지정 cv2.KMEANS_PP_CENTERS, cv2.KMEANS_RANDOM_CENTERS
출력 파라미터	<p>retval : 각 점과 해당 중심의 거리의 제곱 합</p> <p>bestLabels : 각 요소에 주어진 레이블 ('0', '1'...)의 배열 (이전 튜토리얼의 'code')</p> <p>centers : 클러스터의 무게 중심의 배열</p>

h-최근접 이웃 분류기의 이해

- Exam01) 단일 특징 데이터를 이용해서 코드를 생각해 보자.

[step_01]: 학생들 신장(Height)에 대한 티셔츠의 크기를 결정하는 레이어아웃을 구성해 보자.

```
1 #Q1) 특징이 하나인 데이터
2 import numpy as np
3 import cv2
4 from matplotlib import pyplot as plt
5
6 x = np.random.randint(25,100,25)
7 y = np.random.randint(175,255,25)
8 z = np.hstack((x,y))
9 z = z.reshape((50,1))
10 z = np.float32(z)
11 plt.hist(z,256,[0,256]),plt.show()
```



K-최근접 이웃 분류기의 이해

- Exam01) 단일 특징 데이터를 이용해서 코드를 생각해 보자.

[step_02]: 조건에 따라 K-Means 함수를 적용하기 전에 *criteria* 를 지정 한다.

조건 : 반복 회수의 상한을 10회로 하고, 정밀도가 도달했을 때 종료하도록 종료 조건을 설정

```
1 #1-2)조건 정의
2 # 정의 criteria = ( type, max_iter = 10 , epsilon = 1.0 )
3 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
4
5 # 플래그 지정 flags
6 flags = cv2.KMEANS_RANDOM_CENTERS
7
8 # 적용 KMeans
9 compactness, labels, centers = cv2.kmeans(z, 2, None, criteria, 10, flags)
```

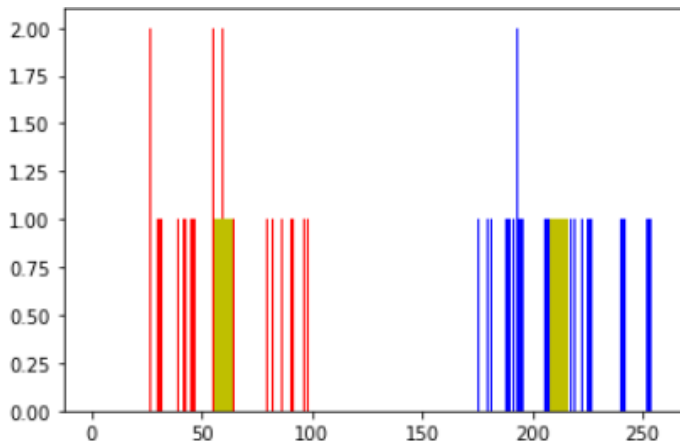
컴팩트, 라벨 및 무게 중심을 각각 출력 해봅니다 ~

k-최근접 이웃 분류기의 이해

- Exam01) 단일 특징 데이터를 이용해서 코드를 생각해 보자.

[step_03]: 레이블에 따라 다른 클러스터로 나누고 출력

```
1 #1-3)
2 #레이블에 따라 다른 클러스터로 나눈다.
3 A = z[labels==0]
4 B = z[labels==1]
5 #A를 빨간색, B를 파란색, 각 무게 중심을 노란색으로 그린다.
6 plt.hist(A,256,[0,256],color = 'r')
7 plt.hist(B,256,[0,256],color = 'b')
8 plt.hist(centers,32,[0,256],color = 'y')
9 plt.show()
```

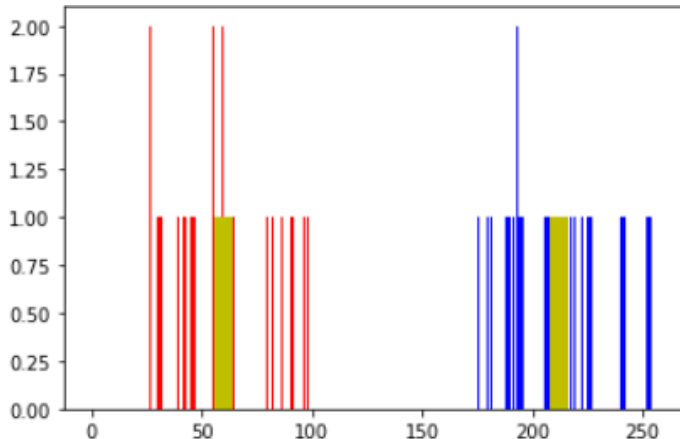


k-최근접 이웃 분류기의 이해

- Exam01) 단일 특징 데이터를 이용해서 코드를 생각해 보자.

[step_03]: 레이블에 따라 다른 클러스터로 나누고 출력

```
1 #1-3)
2 #레이블에 따라 다른 클러스터로 나눈다.
3 A = z[labels==0]
4 B = z[labels==1]
5 #A를 빨간색, B를 파란색, 각 무게 중심을 노란색으로 그린다.
6 plt.hist(A,256,[0,256],color = 'r')
7 plt.hist(B,256,[0,256],color = 'b')
8 plt.hist(centers,32,[0,256],color = 'y')
9 plt.show()
```



단일 특징은 데이터를 하나의 열 벡터로 사용!

Exam02) 다중 특징에 대한 특징(Features)으로 분류

- 신장과 체중이라는 특성으로 생각 = 50명의 신장 데이터를 50×2 로 생각
- 첫 번째 열은 Height, 두 번째 열은 Weight

```
1 #Q2) 다중특징
2 import numpy as np
3 import cv2
4 from matplotlib import pyplot as plt
5
6 X = np.random.randint(25,50,(25,2))
7 Y = np.random.randint(60,85,(25,2))
8 Z = np.vstack((X,Y))
9
10
11 Z = np.float32(Z)
12
13 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
14 ret,label,center=cv2.kmeans(Z,2,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
15
16 A = Z[label.ravel()==0]
17 B = Z[label.ravel()==1]
18
19 plt.scatter(A[:,0],A[:,1])
20 plt.scatter(B[:,0],B[:,1],c = 'r')
21 plt.scatter(center[:,0],center[:,1],s = 80,c = 'y', marker = 's')
22 plt.xlabel('Height'),plt.ylabel('Weight')
23 plt.show()
```

아오평을 생각해 봅니다!

k-최근접 이웃 분류기의 이해

- 색상의 양자와란? 이미지에서 사용되는 색상 수를 줄이는 과정

k-mean 으로 ? 3개의 특징(색의 RGB 성분)의 이미지를 $M \times 3$ 크기의 배열로 변형한다. (M = 이미지의 픽셀 수)

클러스터링 후 무게 중심의 값 (RGB 값)을 모든 픽셀에 적용하여 적용 후 이미지의 색상 수가 지정된 수가 되도록 한다.

```
1 #Q3) 색의 양자화
2 import numpy as np
3 import cv2
4
5 img = cv2.imread('c:\\myImg\\house_01.jpg')
6 Z = img.reshape((-1,3))
7
8
9 Z = np.float32(Z)
10
11
12 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
13 K = 8 #값을 2,4,8로 변경해서 확인
14 ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
15
16 # Now convert back into uint8, and make original image
17 center = np.uint8(center)
18 res = center[label.flatten()]
19 res2 = res.reshape((img.shape))
20
21 cv2.imshow('res_k_min',res2)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()
```

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

