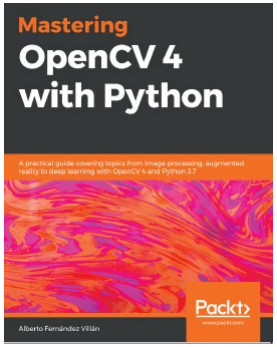
OpenCV Python

디지털영상처리/컴퓨터비전

K means Clustering

교재 10장의 일부..



Mastering OpenCV 4 with Python, Alberto, Packt, Pub. 2019

2021년 2학기

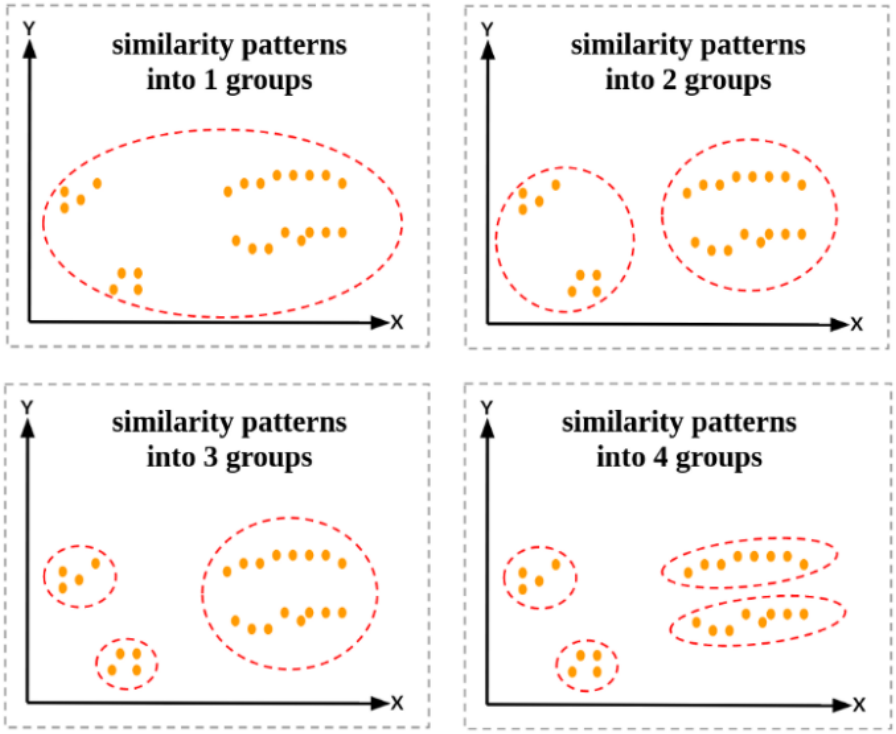
서경대학교 김진헌

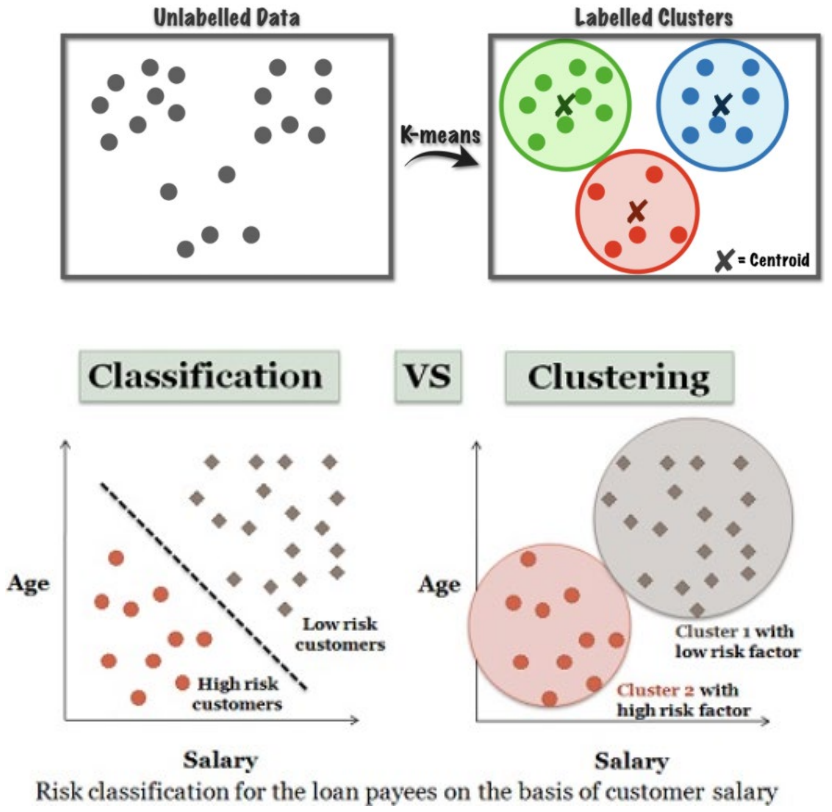
개요

1

- 본 topic의 목표
  - 기계학습의 입문 알고리즘이라 할 수 있는 k-means clustering의 특징과 원리를 이해한다.
- 내용
  - K 군집화알고리즘을 24비트 컬러를 64, 256 컬러로 줄이는데 활용해보자.
    - Indexed color를 사용하는 영상 파일(PNG, BMP 등)에서 사용하고 있음
- 코딩상의 촛점
  - OpenCV의 kmeans() 함수의 사용법
  - 2채널, 3채널 자료에 군집화 알고리즘 적용
  - scatter(), 각종 파이썬 함수-flatten(), ravel()

K-means clustering

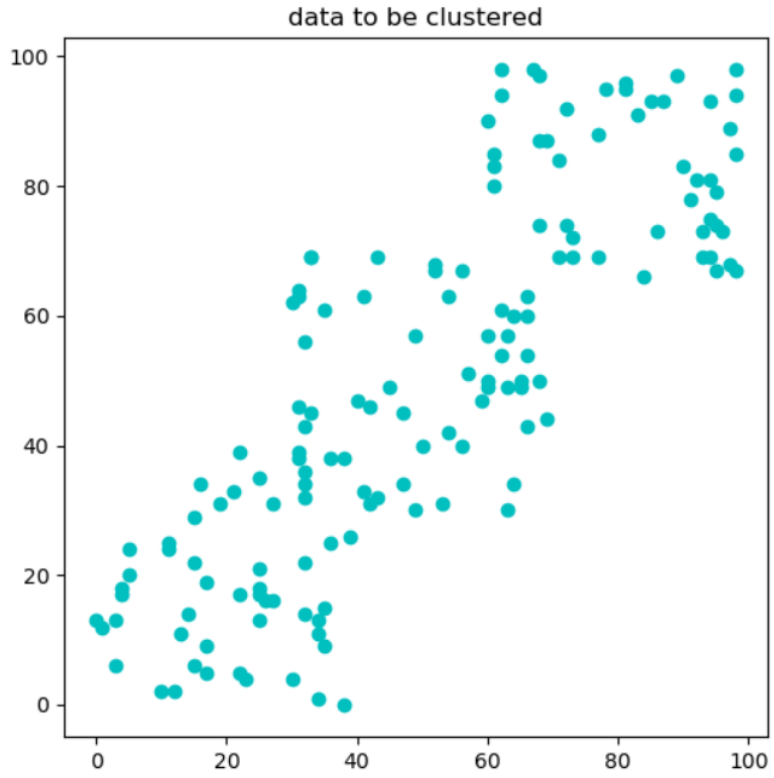




Risk classification for the loan payees on the basis of customer salary

K means clustering
- 랜덤 좌표에 위치를 나타내는 점 그리기

k_means_clustering_data_visualization.py



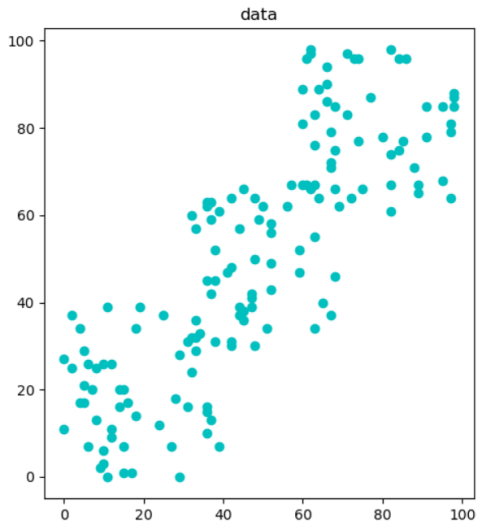

```
data = np.float32(np.vstack((a, b, c)))
scatter(data[:, 0], data[:, 1], c='m')
```

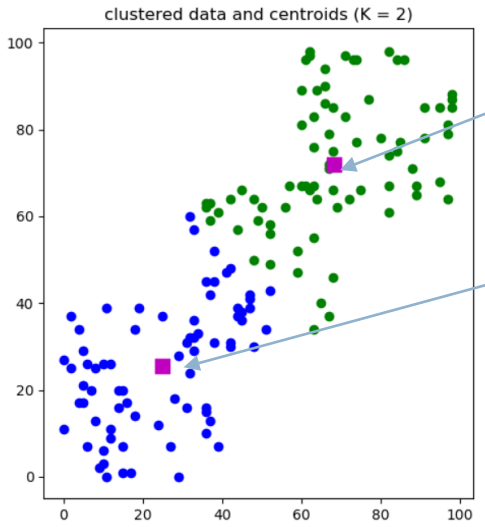
30개의 랜덤 좌표 x 3세트,총 150개의 랜덤 좌표에 magenta 색상으로 표시한다.
각 세트는 지정된 범위의 랜덤 좌표(x, y)로 구성되어 있다.
랜덤 좌표 정보는 randint() 함수로, 점을 찍는 함수는 scatter()로 구현하였다.

- 특정 범위의 정수로 이루어진 특정 차원의 어레이 생성하기
- numpy.random.randint(low, high=None, size=None, dtype='l')
- a = np.random.randint(0, 40, (50, 2))
- 위치값이 [0, 40)까지의 좌표 값이 존재하는 50개의 2차원 좌표 어레이를 생성한다.
- ⇨ 3개의 어레이를 vstack() 함수로 묶어 50x3개의 랜덤 좌표 데이터를 생성하여 그 지점에 점을 그린다.

K means clustering
- 2개로 클러스터링 한다.

k_means_clustering_k_2.py





클러스터 A의 중점

```
center:
 [[71.25      68.54762 ]
 [23.48485   25.621212]]
```

클러스터 B의 중점

kmeans() 함수의 반환값, ret

$$\sum_i \|\text{samples}_i - \text{centers}_{\text{labels}_i}\|^2$$

2개로 그룹핑

```
data: <class 'numpy.ndarray'>, shape=(150, 2), len=150
ret, label, center = cv2.kmeans(data, 2, None, criteria,
                                10, cv2.KMEANS_RANDOM_CENTERS)
ret: 71623.28
label: type=<class 'numpy.ndarray'>, shape=(150, 1)
center: type=<class 'numpy.ndarray'>, shape=(2, 2)
label.ravel(): shape=(150,)
```

10회의 초기화 라벨링을 통한 탐색 시도

data의 각 원소가 배정 받은 레이블을 저장

```
A = data[label.ravel() == 0]
B = data[label.ravel() == 1]
```

⇨

```
label.ravel(): shape=(150,)
A: shape=(73, 2)
B: shape=(77, 2)
```

kmeans() 함수

- *retval, bestLabels, centers*=**cv.kmeans**(*data*, K, None, criteria, attempts, flags)

data	•Data for clustering. An array of N-Dimensional points with float coordinates is needed. N 차원을 가진 Z개의 데이터이다. •이 경우 data.shape=(Z, N)이다. •(x, y) 위치 정보를 가진 Z개의 데이터: data.shape=(Z, 2) •후술할 색상 양자화 예제에서 3채널 영상 전체(면적이 A라 하자)의 화소를 데이터로 지정하면 data.shape=(A, 3)이다.
K	Number of clusters to split the set by.
bestLabels	Input/output integer array that stores the cluster indices for every sample. 0-based cluster index for the *data*

kmeans() 함수

- *retval, bestLabels, centers*=**cv.kmeans**(*data*, K, None, criteria, attempts, flags)

criteria The algorithm termination criteria, that is, the maximum number of iterations and/or the desired accuracy.

criteria= `TermCriteria (int type, int maxCount, double epsilon)`

Constructor

생성자 함수 없이 튜플형으로 (type, maxCount, epsilon)으로 써도 됨

type The type of termination criteria, one of TermCriteria::Type

maxCount The maximum number of iterations or elements to compute.

epsilon The desired accuracy or change in parameters at which the iterative algorithm stops.

kmeans() 함수

- *retval, bestLabels, centers*=**cv.kmeans**(*data*, K, None, criteria, attempts, flags)

attempts Flag to specify the number of times the algorithm is executed using dif ferent initial labellings. The algorithm returns the labels that yield the b est compactness (see the last function parameter).

flags Flag that can take values of **cv::KmeansFlags**

KMEANS_RANDOM_CENTERS Python: cv.KMEANS_RANDOM_CENTERS	Select random initial centers in each attempt.
KMEANS_PP_CENTERS Python: cv.KMEANS_PP_CENTERS	Use kmeans++ center initialization by Arthur and Vassilvitskii [Arthur2007].
KMEANS_USE_INITIAL_LABELS Python: cv.KMEANS_USE_INITIAL_LABELS	During the first (and possibly the only) attempt, use the user-supplied labels instead of computing them from the initial centers. For the second and further attempts, use the random or semi-random centers. Use one of KMEANS_*_CENTERS flag to specify the exact method.

centers Output matrix of the cluster centers, one row per each cluster center.

kmeans() 함수

- *retval, bestLabels, centers*=**cv.kmeans**(*data*, K, None, criteria, attempts, flags)

Returns

The function returns the compactness measure that is computed as

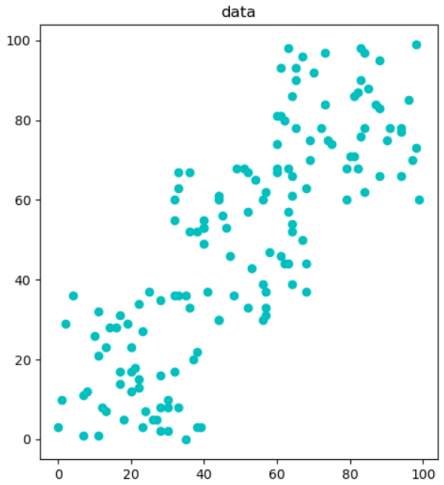
$$\sum_i \|\texttt{samples}_i - \texttt{centers}_{\texttt{labels}_i}\|^2$$

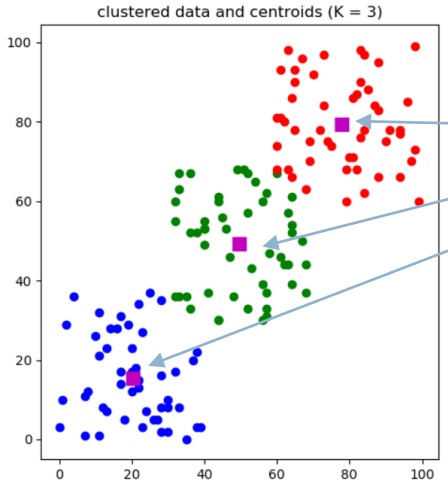
attempts 에서 지정한 회수 만큼 초기 중심값을 바꾸어가면서 가장 적은 compactness를 갖는 클러스터링 결과를 물색한다.

flags = KMEANS_USE_INITIAL_LABELS) flag를 선택하고 attempts=1로 선정하면 프로그래머가 직접 자신의 알고리즘으로 그 중심값의 초기화 값을 지정하면서 중심점을 찾아나간다.

K means clustering
- 3개로 클러스터링 한다.

k_means_clustering_k_3.py





각 클러스터의 중점

kmeans() 함수의 반환값, compactness,
ret= $\sum_i \|samples_i - centers_{labels_i}\|^2$

10번의 초기값 설정: 이 만큼의 시도를 해본 후 가장 적은 compactness(편차)를 내는 클러스터링 결과를 반환한다.

3개로 그룹핑

```
data: <class 'numpy.ndarray'>, shape=(150, 2), len=150

ret, label, center = cv2.kmeans(data, 3, None, criteria,
                                10, cv2.KMEANS_RANDOM_CENTERS)

ret: 39487.91
label: type=<class 'numpy.ndarray'>, shape=(150, 1)
center: type=<class 'numpy.ndarray'>, shape=(3, 2)
```

data의 각 원소가 배정 받은 레이블을 저장

```
A = data[label.ravel() == 0]
B = data[label.ravel() == 1]
C = data[label.ravel() == 2]
```

⇨

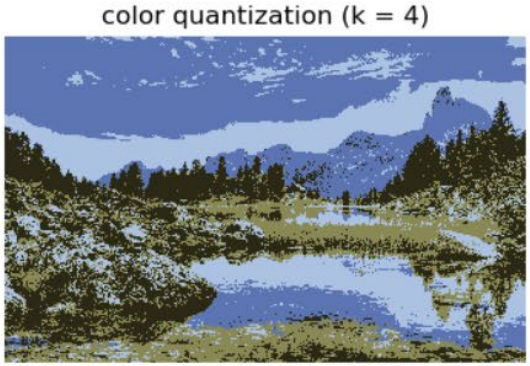
```
A: shape=(50, 2)
B: shape=(52, 2)
C: shape=(48, 2)
```

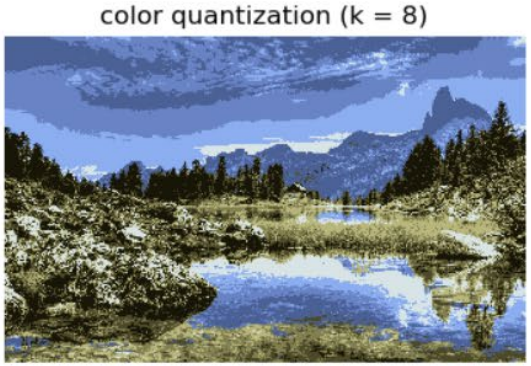
Color quantization using k-means clustering

영상존재하는 24비트 색상을 k개의 색상으로 압축: k개의 군집화

각 군집의 중심은 (r, g, b) 값을 갖고 있다.

k_means_color_quantization.py






```
k=2:  time=0.19,  PSNR=15.21
k=3:  time=0.33,  PSNR=17.03
k=4:  time=0.46,  PSNR=18.57
k=8:  time=0.90,  PSNR=21.95
k=16:  time=1.63,  PSNR=24.95
```

영상을 1비트로 묘사

영상을 2비트로 묘사

영상을 3비트로 묘사

영상을 4비트로 묘사

Color quantization using k-means clustering

영상존재하는 24비트 색상을 k개의 색상으로 압축: k개의 군집화

각 군집의 중심은 (r, g, b) 값을 갖고 있다.

k_means_color_quantization.py


```
k=32: time=2.80, PSNR=27.47
k=64: time=5.81, PSNR=29.88
k=128: time=12.02, PSNR=31.75
k=256: time=25.11, PSNR=32.76
```

영상을 5비트로 묘사
영상을 6비트로 묘사
영상을 7비트로 묘사
영상을 8비트로 묘사 ⇦ 영상 데이터를 1/3으로 축소

Color quantization using k-means clustering

영상존재하는 24비트 색상을 k개의 색상으로 압축: k개의 군집화

각 군집의 중심은 (r, g, b) 값을 갖고 있다.

k_means_color_quantization.py

```
data = np.float32(image).reshape((-1, 3))
# 맨 뒤자리 차원 3은 고정하고 나머지는 정리하여 1차원으로 바꾼다.
```

3채널 영상 전체(면적이 A라 하자)의 화소를 데이터로 지정하면 data.shape=(A, 3)이다.

```
ret, label, center = cv2.kmeans(data, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
```

k=2: data.shape=(400000, 3)
k=2: center.shape=(2, 3)
k=3: data.shape=(400000, 3)
k=3: center.shape=(3, 3)
k=6: data.shape=(400000, 3)
k=6: center.shape=(6, 3)

클러스터의 갯수와 같은 수의
중심값이 반환된다.
이는 (r, g, b)에 따라 갖고 있어서
두번째 차원이 3이다.

각 화소(r, g, b)의 값을 배정받은
레이블의 중앙값으로 매핑시킨다.

```
result = center[label.flatten()]
result = result.reshape(img.shape)
psnr = cv2.PSNR(image, result)
```

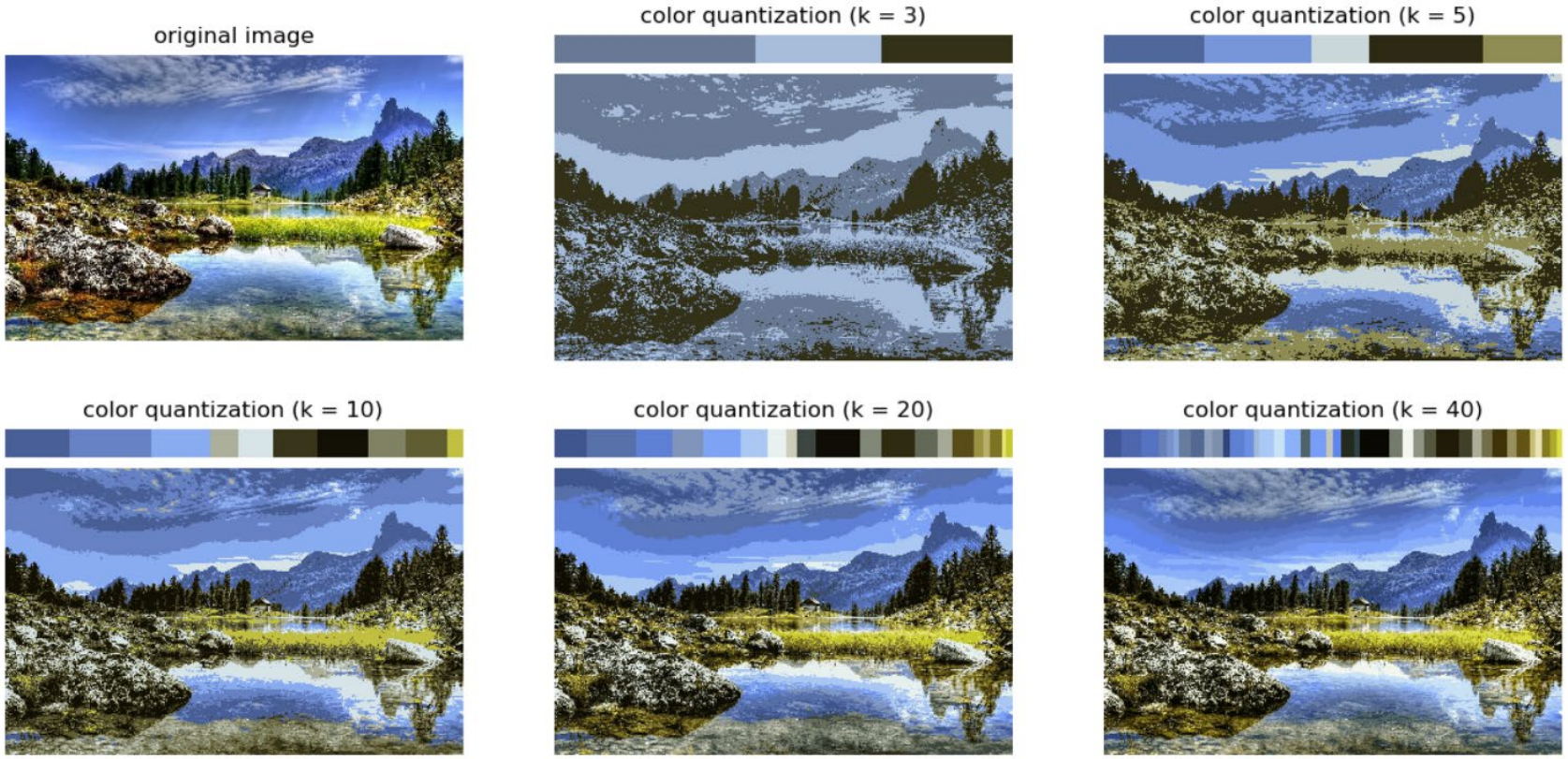
원래 영상과 같이 3차원의 영상으로 복원
면적x 3 => 행x열x3

두 영상의 PSNR을 연산한다.

Color quantization using k-means clustering

배정받은 레이블의 분포도를 그래프와 문자로 출력한다

k_means_color_quantization_distribution.py



각 k에 대하여 화소의 수량이 많은 순으로 레이블을 재배치 하여 출력한 결과

```
k=3: counter=Counter({0: 175300, 2: 114788, 1: 109912})
k=5: counter=Counter({0: 99656, 2: 92850, 3: 88215, 4: 68037, 1: 5
k=10: counter=Counter({5: 71295, 9: 56275, 1: 51161, 8: 44675, 0:
k=20: counter=Counter({16: 43130, 12: 38141, 9: 33107, 18: 32526,
k=40: counter=Counter({38: 25871, 23: 19068, 32: 17077, 9: 15989,
```

K-means Clustering 알고리즘

위키피디아 자료 링크

- 개념적 설명은 자료를 참조하기를 추천합니다.
- 알고리즘의 목표
- 알고리즘의 설명 및 애니메이션
- K-measn clustering 알고리즘의 한계점

OpenCV machine learning Tutorials -python, K-means Clustering

opencv machine learning tutorials

- docs.opencv.org => main page => opencv-python tutorials -> machine learning

•**K-Nearest Neighbour**
•Learn to use kNN for classification Plus learn about handwritten digit recognition using kNN

•**Support Vector Machines (SVM)**
•Understand concepts of SVM

✓•**K-Means Clustering**
•Learn to use K-Means Clustering to group data to a number of clusters. Plus learn to do color quantization using K-Means Clustering

PSNR

- 영상의 손괴가 일어난 정도를 나타내는 척도. 보통 40dB 이상 넘어가면 육안으로는 두 영상의 차이를 분간하기 어려워진다.
- SNR=10 log(신호전력/ 잡음전력)
  - 전력의 개념을 각 계조치의 제곱으로 대용
  - Peak란 말은 신호전력이 최대일 때를 일컫는 말. 따라서 신호 255의 제곱(=255^2)을 사용한다.

$$PSNR = 10 \cdot \log_{10}\left(\frac{255^2}{MSE}\right)[dB]$$

$$MSE = \frac{1}{NM}\sum_{x=1}^{N}\sum_{y=1}^{M}\left(f(x,y) - g(x,y) \right)^2$$

- f(x,y) : 좌표(x,y)의 원본영상 픽셀값. 영상의 크기는 가로*세로=N*M.
- g(x,y) : 좌표(x,y)의 비교영상 픽셀값. 영상의 크기는 가로*세로=N*M.
- MSE : Mean Square Error. 두 영상의 오차의 제곱을 영상의 면적으로 나눔.

참고: Indexed Color

- 칼라 팔레트를 사용하는 파일의 색상 정보 표현 기법
  - 8비트 픽셀 정보의 실제 색상 정보는 칼라 팔레트에 정의되어 있다.
  - 이 팔레트는 파일의 헤더 부분에 정의되어 있다.
  - 8비트는 Video Card에서 실제 색상으로 변환과정에 RGB용 3개의 DAC를 거친다.
  - 색상의 분포를 잘 활용하면 데이터를 줄이면서 제법 실사와 가까운 영상을 만들어 낼 수 있다.

Color Palette (Registers of RAM DAC)

참고: Indexed Color

https://en.wikipedia.org/wiki/Indexed_color

- 인덱스 색상 지원 그래픽 파일 형식
  - Photoshop(PSD), BMP, DICOM(Digital Imaging and Communications in Medicine), GIF, Photoshop EPS, 대용량 문서 형식(PSB), PCX, Photoshop PDF, Photoshop Raw, Photoshop 2.0, PICT, PNG, Targa® 또는 TIFF 형식
  - 각종 그래픽 파일 형식 소개
    - URL: https://helpx.adobe.com/kr/photoshop/using/file-formats.html
  - Web에서 PNG , GIF, JPEG , SVG 중 어떤 것을 사용하면 좋을까요?
- 인덱스 칼라 파일의 Python 코딩
  - https://stackoverflow.com/questions/33022983/read-in-an-indexed-color-image-in-python
  - https://pillow.readthedocs.io/en/latest/
  - How to reduce color palette with PIL
-