zero-base /

Life Changing Education

Chapter 31. 군집을 이용한 이미지 분할

이미지 분할 image segmentation

zero-base /

이미지 분할

- 이미지 분할 image segmentation은 이미지를 여러 개로 분할하는 것
- 시맨틱 분할 semantic segmentation은 동일 종류의 물체에 속한 픽셀을 같은 세그먼트로 할당
- 시맨틱 분할에서 최고의 성능을 내려면 역시 CNN 기반으로 …
- 지금은 단순히 색상 분할로 시도

zero-base /

이미지 한 장을 읽자

```
from matplotlib.image import imread
image = imread('./ladybug.png')
image.shape
```

```
(533, 800, 3)
```

zero-base /

이렇게 생김

```
plt.imshow(image);
```

zero-base /

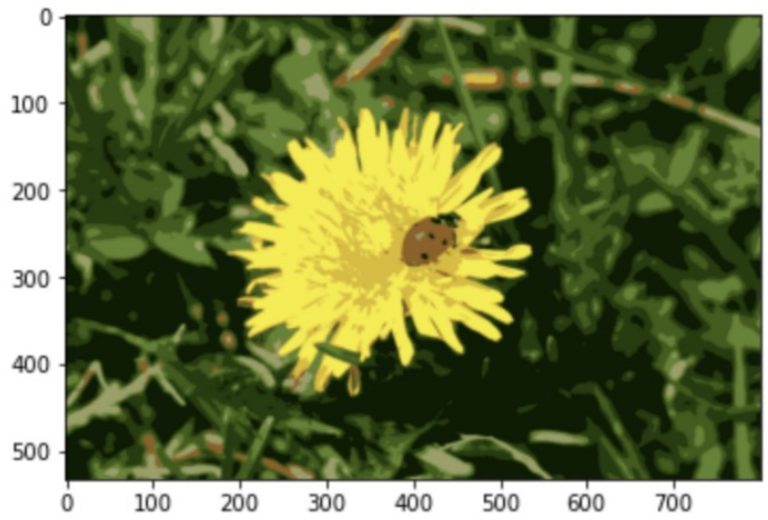
색상별로 클러스터링 하라고 함

```
from sklearn.cluster import KMeans

X = image.reshape(-1, 3)
kmeans = KMeans(n_clusters=8, random_state=13).fit(X)
segmented_img = kmeans.cluster_centers_[kmeans.labels_]
segmented_img = segmented_img.reshape(image.shape)
```

zero-base /

결과 - 색상의 종류가 단순하게 됨

```
plt.imshow(segmented_img);
```



zero-base /

이번에는 여러개의 군집을 비교

```
segmented_imgs = []
n_colors = (10, 8, 6, 4, 2)
for n_clusters in n_colors:
    kmeans = KMeans(n_clusters=n_clusters, random_state=13).fit(X)
    segmented_img = kmeans.cluster_centers_[kmeans.labels_]
    segmented_imgs.append(segmented_img.reshape(image.shape))
```

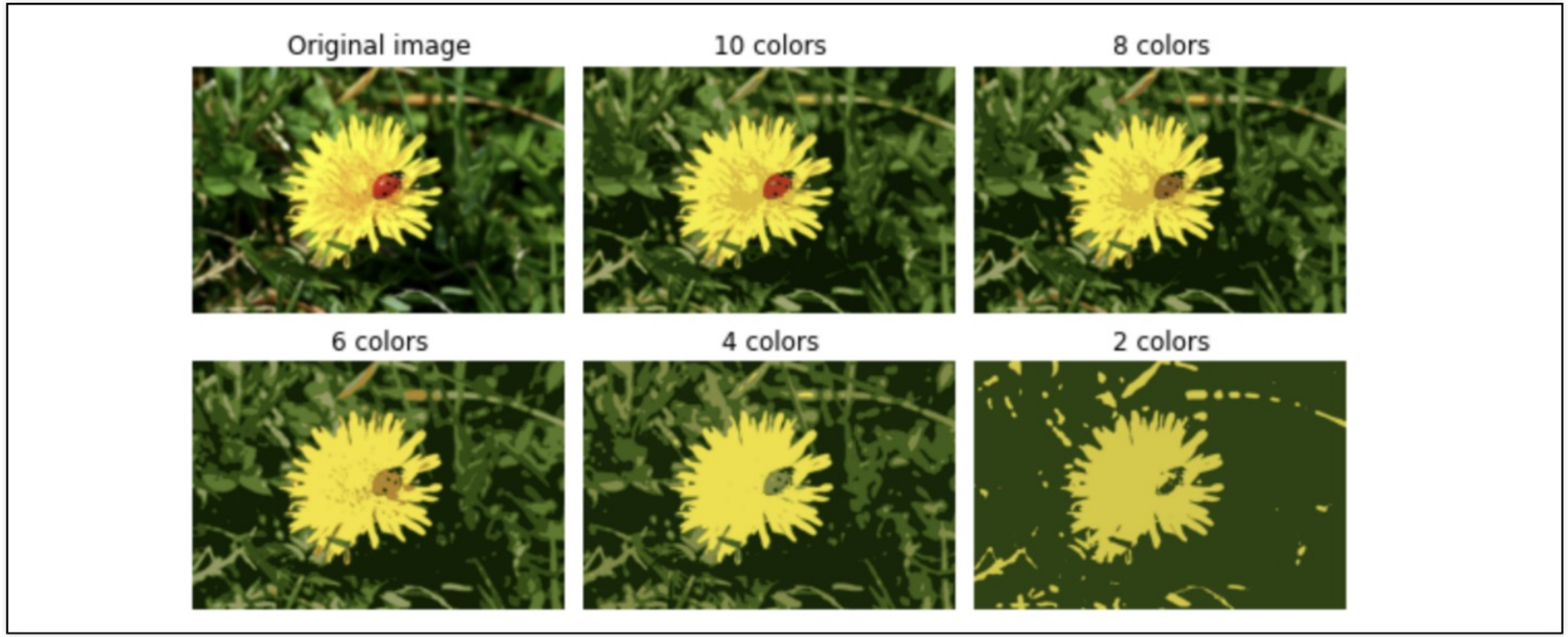
zero-base /

이번에는 좀 복잡하게 결과를 시각화

```
plt.figure(figsize=(10,5))
plt.subplots_adjust(wspace=0.05, hspace=0.1)

plt.subplot(231)
plt.imshow(image)
plt.title("Original image")
plt.axis('off')

for idx, n_clusters in enumerate(n_colors):
    plt.subplot(232 + idx)
    plt.imshow(segmented_imgs[idx])
    plt.title("{} colors".format(n_clusters))
    plt.axis('off')

plt.show()
```

zero-base /

이 정도만 되어도 간단하게 쓸 만하지 않을까



zero-base /

그럼 MNIST 데이터로 다시 가보자

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

X_digits, y_digits = load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X_digits,
                                                    y_digits, random_state=13)
```

zero-base /

로지스틱 회귀

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(multi_class="ovr",
                             solver="lbfgs", max_iter=5000, random_state=13)
log_reg.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=5000, multi_class='ovr', random_state=13)
```

- 다중 분류라서

zero-base /

결과도 나쁘지는 않다

```
log_reg.score(X_test, y_test)
```

```
0.9622222222222222
```

zero-base /

이번에는 전처리 느낌으로 kmeans를 통과시켜볼까?

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ("kmeans", KMeans(n_clusters=50, random_state=13)),
    ("log_reg", LogisticRegression(multi_class="ovr",
                                   solver="lbfgs", max_iter=5000,
                                   random_state=13)),
])
pipeline.fit(X_train, y_train)
```

zero-base /

약간 상승한다~

```
pipeline.score(X_test, y_test)
```

```
0.9711111111111111
```

zero-base /

Gridsearch로 가보자

```
from sklearn.model_selection import GridSearchCV

param_grid = dict(kmeans__n_clusters=range(2, 100))
grid_clf = GridSearchCV(pipeline, param_grid, cv=3, verbose=2)
grid_clf.fit(X_train, y_train)
```

zero-base /

좀 더 상승한다

```
grid_clf.best_params_
```

```
{'kmeans__n_clusters': 77}
```

```
grid_clf.score(X_test, y_test)
```

```
0.9733333333333334
```