completed_notebook

June 5, 2020

Image Compression with K-means Clustering

0.1 ### Task 1: Importing Libraries

[2]:
```
from __future__ import print_function
%matplotlib inline
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as image
plt.style.use("ggplot")

from skimage import io
from sklearn.cluster import KMeans

from ipywidgets import interact, interactive, fixed, interact_manual, IntSlider
import ipywidgets as widgets
```

[3]:
```
plt.rcParams['figure.figsize'] = (20, 12)
```

0.2 ### Task 2: Data Preprocessing

[4]:
```
img = io.imread('images/1-Saint-Basils-Cathedral.jpg')
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(img);
```

[5]:
```
img.shape
```

[5]:
```
(600, 394, 3)
```

[6]:
```
img
```

[6]:
```
Array([[[ 15,  28,  73],
        [ 15,  28,  73],
        [ 15,  28,  73],
        …,
        [ 13,  46, 115],
        [ 14,  47, 116],
        [ 14,  47, 116]],

       [[ 15,  28,  73],
        [ 15,  28,  73],
        [ 15,  28,  73],
        …,
        [ 13,  46, 115],
        [ 13,  46, 115],
        [ 13,  46, 115]],

       [[ 15,  28,  73],
        [ 15,  28,  73],
        [ 15,  28,  73],
        …,
        [ 13,  46, 115],
        [ 12,  45, 114],
        [ 12,  45, 114]],

       …,

       [[ 92,  70,  93],
        [ 91,  72, 102],
        [ 84,  68, 105],
        …,
        [205,  83,  96],
        [230, 102,  93],
        [230, 102,  93]],

       [[ 98,  82,  93],
        [ 96,  77,  99],
        [105,  80, 112],
        …,
        [147,  63,  76],
```

```
        [160,  63,  83],
        [161,  64,  84]],

       [[ 90,  71,  73],
        [104,  77,  92],
        [116,  80, 106],
        …,
        [179,  76,  67],
        [ 98,  57,  75],
        [ 95,  54,  72]]], dtype=uint8)
```

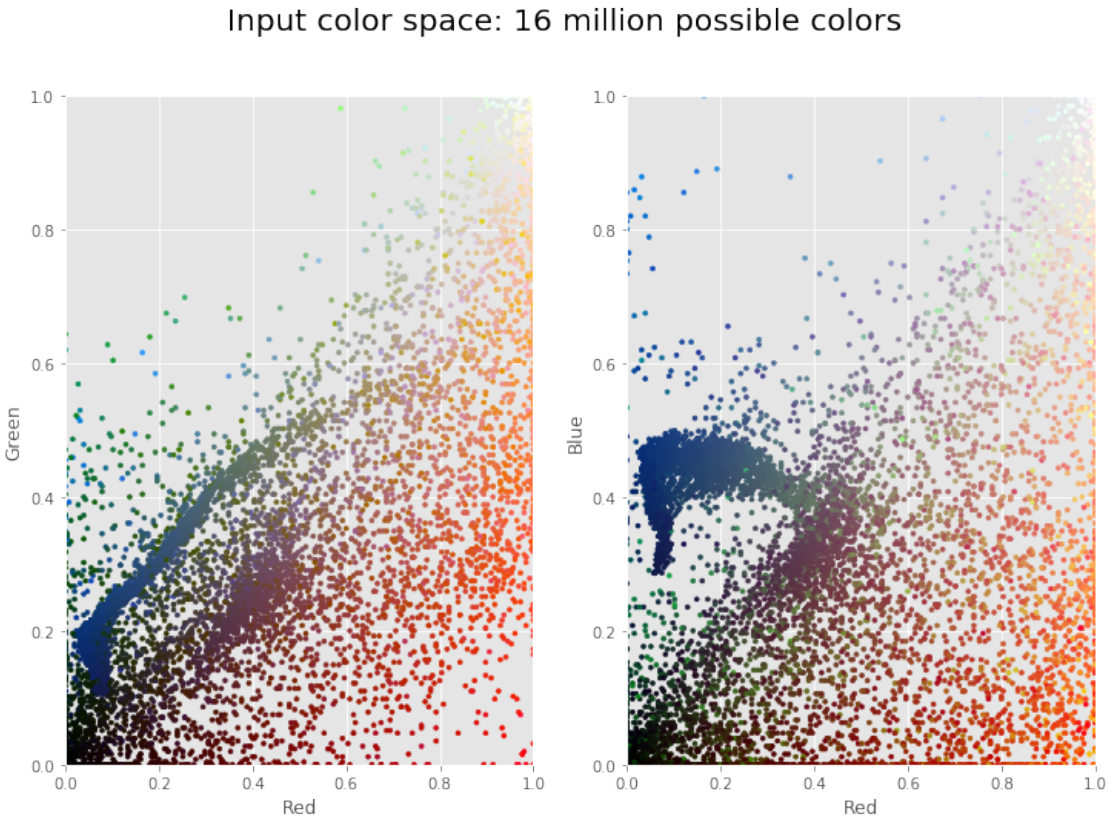
[8]:
```
img_data = (img / 255.0).reshape(600 * 394, 3) # .reshape(-1, 3)
img_data.shape
```

[8]:
```
(236400, 3)
```

0.3 ### Task 3: Visualizing the Color Space using Point Clouds

[15]:
```
from plot_utils import plot_utils
```

[16]:
```
x = plot_utils(img_data, title='Input color space: 16 million possible colors')
x.colorSpace()
```



0.4 ### Task 4: Visualizing the K-means Reduced Color Space

Incheol [CC BY-SA 4.0], via Wikimedia Commons

K-means Algorithm:

1. Initialization: Randomly sample k colors from the input image. These are the initial k means $\mu_1, \mu_2, ..., \mu_k$.
2. For each pixel in the image, assign it to its nearest mean given by

$$c^{(i)} := \text{argmin}_j \left\| x^{(i)} - \mu_j \right\|^2$$

3. Update the means using the pixel assignments from Step 2.

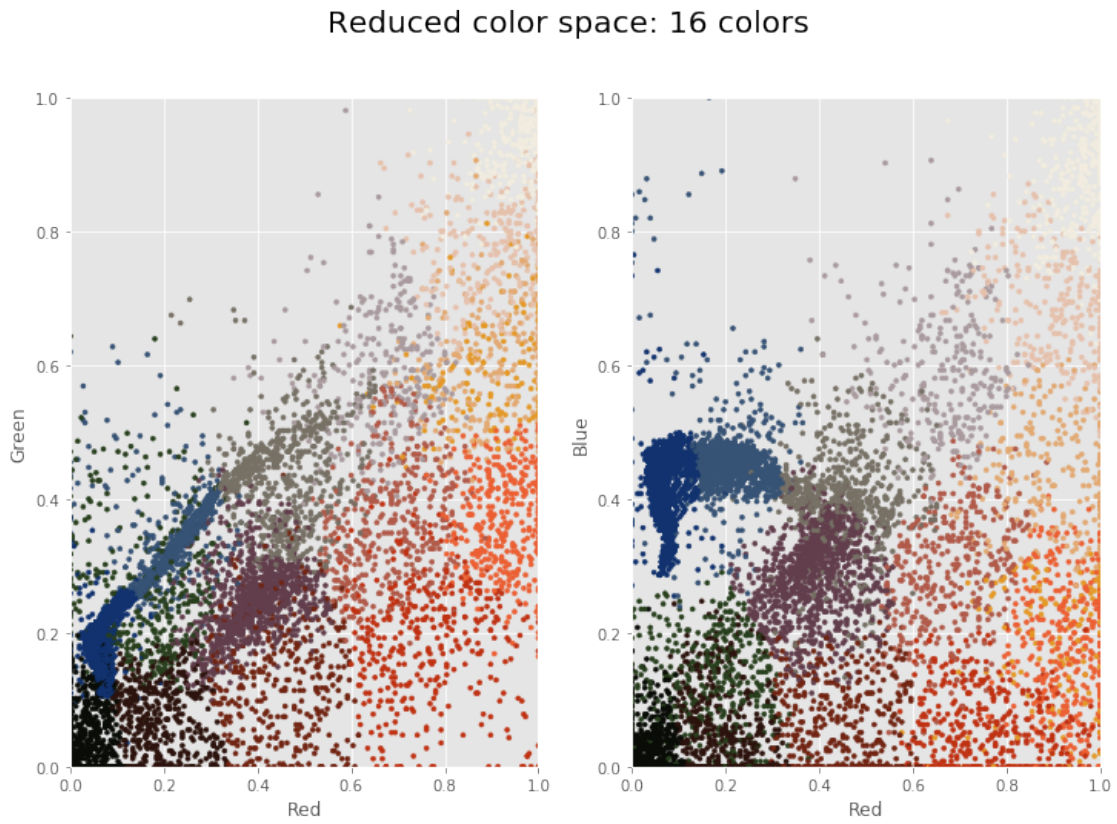
$$\mu_j := \frac{\sum_i^n 1\left\{c^{(i)} = j\right\} x^{(i)}}{\sum_i^n 1\left\{c^{(i)} = j\right\}}$$

4. Repeat Steps 2 and 3 until convergence.

[17]:
```
from sklearn.cluster import MiniBatchKMeans
```

[18]:
```
kmeans = MiniBatchKMeans(16).fit(img_data)
k_colors = kmeans.cluster_centers_[kmeans.predict(img_data)]

y = plot_utils(img_data, colors=k_colors, title="Reduced color space: 16␣
 ↪colors")
y.colorSpace()
```



0.5 ### Task 5: K-means Image Compression with Interactive Controls

[19]:
```
img_dir = 'images/'
```

[20]:
```
@interact
def color_compression(image=os.listdir(img_dir),␣
 ↪k=IntSlider(min=1,max=256,step=1,value=16,

                                                  ␣
 ↪continuous_update=False,

                                                  ␣
 ↪layout=dict(width='100%'))):

    input_img = io.imread(img_dir + image)
    img_data = (input_img / 255.0).reshape(-1, 3)

    kmeans = MiniBatchKMeans(k).fit(img_data)
    k_colors = kmeans.cluster_centers_[kmeans.predict(img_data)]
    #After K-means has converged, load the large image into your program and
```

```
    #replace each of its pixels with the nearest of the centroid colors you␣
↪found
    #from the small image.
    k_img = np.reshape(k_colors, (input_img.shape))

    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle('K-means Image Compression', fontsize=20)

    ax1.set_title('Compressed')
    ax1.set_xticks([])
    ax1.set_yticks([])
    ax1.imshow(k_img)

    ax2.set_title('Original (16,777,216 colors)')
    ax2.set_xticks([])
    ax2.set_yticks([])
    ax2.imshow(input_img)

    plt.subplots_adjust(top=0.85)
    plt.show()
```

```
interactive(children=(Dropdown(description='image', options=('3-peacock.jpg', '1-Saint-Basils-C
```

[ ]:

[ ]: