HW14-作业讲评

王瑞环

1.1 Hough变换

```
image_blurred = cv2.blur(image_gray, (5, 5))
edges = cv2.Canny(image_blurred, 100, 120, apertureSize=3)
lines = cv2.HoughLines(edges, 0.5, np.pi / 180, 40)
```

image_blurred

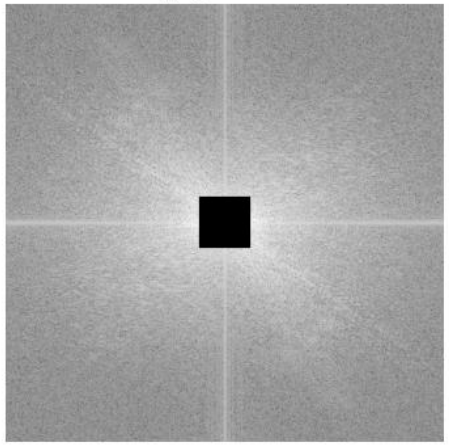
edges

1.2.1 频域滤波

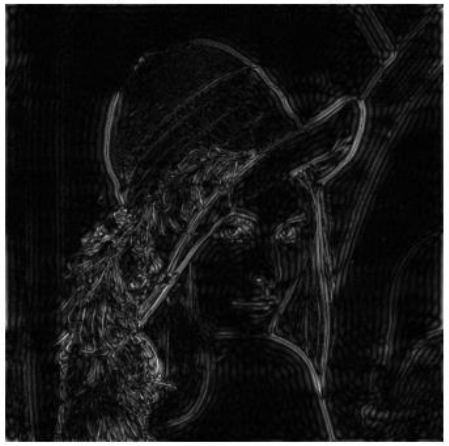
```
c1, c2 = img.shape[0] // 2, img.shape[1] // 2
mask_high = np.ones((img.shape[0], img.shape[1]))
mask_high[c1 - 30: c1 + 30, c2 - 30: c2 + 30] = 0
mask_low = 1 - mask_high
```

original

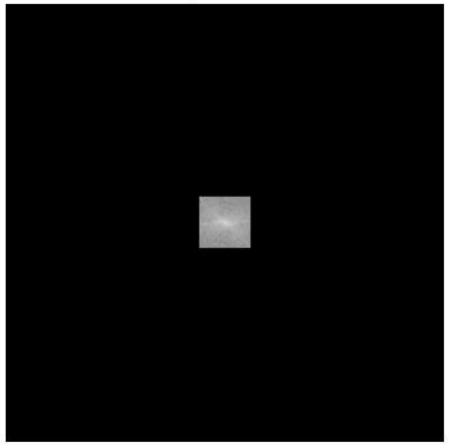
high pass filter



high pass filtered image



low pass filter



low pass filtered image

1.2.2 离散余弦变换

```
img_encode = np.zeros((150, 150, 3))
img_encode[:, :, 0] = cv2.idct(img_dct[-150:, -150:, 0])
img_encode[:, :, 1] = cv2.idct(img_dct[-150:, -150:, 1])
img_encode[:, :, 2] = cv2.idct(img_dct[-150:, -150:, 2])
plt.imshow(img_encode / 32)
plt.show()
```

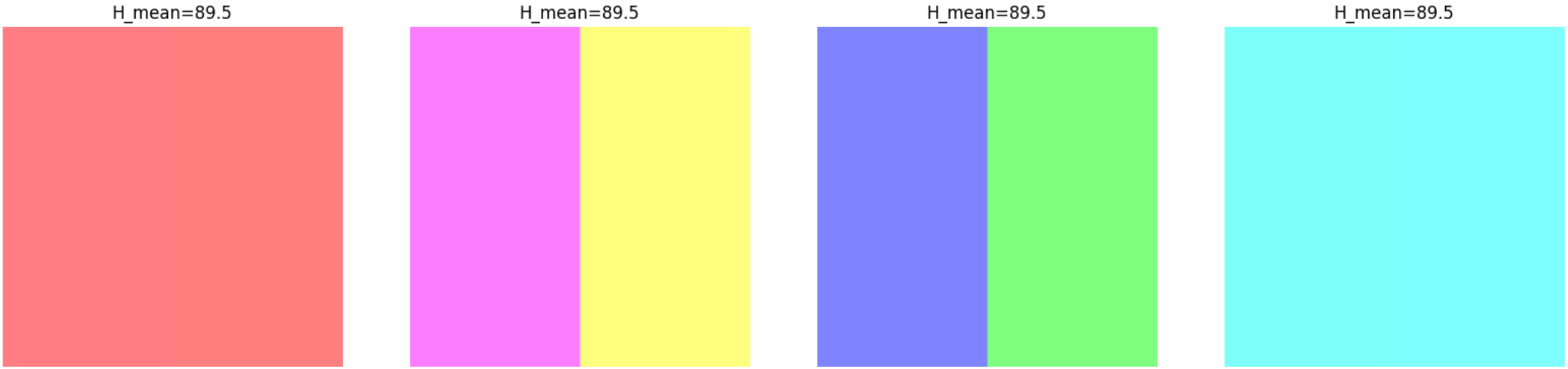
1.3.1 色彩特征提取

```python
def build_feature(img, method):
    assert method in ['h', 's', 'v', 'colorfulness']

    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    H, S, V = cv2.split(hsv)
    average_h = np.mean(H)
    average_s = np.mean(S)
    average_v = np.mean(V)

    R, G, B = cv2.split(img)
    rg = np.absolute(R - G)
    yb = np.absolute(0.5 * (R + G) - B)
    (rbMean, rbStd) = (np.mean(rg), np.std(rg))
    (ybMean, ybStd) = (np.mean(yb), np.std(yb))

    stdRoot = np.sqrt((rbStd ** 2) + (ybStd ** 2))
    meanRoot = np.sqrt((rbMean ** 2) + (ybMean ** 2))
    colorfulness = stdRoot + (0.3 * meanRoot)
```

```python
    if method == 'h':
        return average_h
    elif method == 's':
        return average_s
    elif method == 'v':
        return average_v
    elif method == 'colorfulness':
        return colorfulness
```

1.3.2 聚类改进

- "Trash in, trash out"
- 均值的缺陷：仅反映全局和整体，而忽略局部和分布情况
- （关于H通道，OpenCV中为了能用8 bit表示，将0~360线性映射到了0~180，不少同学忽略了这一点）



1.3.2 聚类改进 – 其他统计量

- 添加标准差/方差等高阶矩特征

```
# 吴昀泽
def build_feature_optimized(img, method):
    if method == 'h':
        feature = np.array([np.mean(img[:,:,0]),np.std(img[:,:,0])])
```

Cluster 3, Method h, Cluster center: [215.11533815 53.12739824]

Cluster 4, Method h, Cluster center: [40.65663009 49.99028455]

1.3.2 聚类改进 – 直方图

- 使用颜色直方图作为特征，将图像中的颜色分布情况考虑进来

```python
# 汤齐宇
def build_feature_new(img, method, bin_num):
    if method == 'h':
        hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
        h_hist, _ = np.histogram(hsv[:,:,0], bins=bin_num, range=(0, 180))
        return h_hist / np.sum(h_hist)
```

```
Cluster 1, Method h, Cluster center: [0.5592302  0.1155274  0.04580614 0.08449077 0.06511684 0.12982865]
```


```
Cluster 2, Method h, Cluster center: [0.12043229 0.03079647 0.03942406 0.6341624  0.10976329 0.0654215 ]
```

1.3.2 聚类改进 – 色彩空间划分

- 参考文献：https://www.cnki.com.cn/Article/CJFDTotal-JSGG200212030.htm

```python
# 程一哲
#----------辅助分类函数----------------
#返回符合bins[i] <= H < bins[i + 1]的组号i
def get_code(H, bins):
    for i in range(len(bins)):
        if H >= bins[i]:
            if i + 1 == len(bins):
                return 0
            if H < bins[i + 1]:
                return i

#----------数据标签预处理---------------
def get_labels(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    label = np.zeros(32)

    for i in range(hsv_img.shape[0]):
        for j in range(hsv_img.shape[1]):
            H = hsv_img[i, j, 0] * 2
            S = hsv_img[i, j, 1] / 255
            V = hsv_img[i, j, 2] / 255
            if V < 0.2:
                code = 0
            elif S < 0.1:
                code = get_code(V, [0.2, 0.5, 0.8, 1]) + 1
            else:
                code = 4 + 4 * get_code(H, [0, 20, 45, 75, 165, 200, 270, 330])\
                         + 2 * get_code(S, [0.1, 0.45, 1])\
                         + get_code(V, [0, 0.2, 0.5])
            label[code] += 1
    return label
```

```python
n_cluster = 7
labels = [get_labels(poster) for poster in posters]

kmeans = KMeans(n_clusters=n_cluster, init='k-means++').fit(labels)

cluster_center = kmeans.cluster_centers_
color = ['Black', 'DeepGray', 'LightGray', 'White',\
         'Red', 'Orange', 'Yellow', 'Green',\
         'Cyan', 'Blue', 'Purple']

for i in range(n_cluster):
    color_id = np.argmax(cluster_center[i])
    print(f'Cluster {i}, Method hsv, Color {color[color_id if color_id < 4 else
(int(color_id / 4) + 3)]}')

    idx_to_show = np.where(kmeans.labels_ == i)[0][:10]
    posters_to_show = [posters[j] for j in idx_to_show]

    show_imgs(posters_to_show)
```

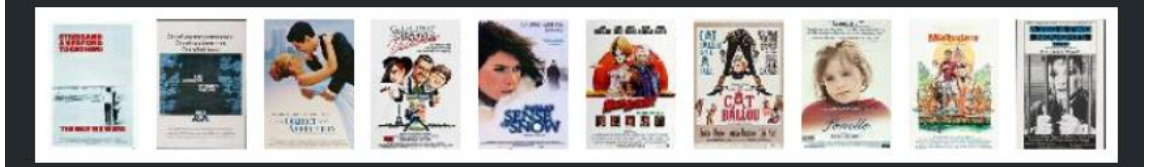
1.3.2 聚类改进 – 色彩空间划分

Cluster 0, Method hsv, Color Red

Cluster 1, Method hsv, Color Black

Cluster 2, Method hsv, Color Black

Cluster 3, Method hsv, Color White



Cluster 4, Method hsv, Color DeepGray

Cluster 5, Method hsv, Color Orange

Cluster 6, Method hsv, Color Blue

1.3.2 聚类改进 – 众数/多数均值

```python
# 曾为帅，代码有改动
def my_build_feature(img):
    img_hsv = cv2.cvtColor(img,cv2.COLOR_RGB2HSV)
    h = cv2.split(img_hsv)[0]
    kernel = cv2.getGaussianKernel(3,sigma=1)
    h_filtered = cv2.filter2D(h,-1,kernel)
    h_segment = h_filtered.astype(np.uint8) // 30
    h_count_max = np.bincount(h_segment.flatten()).argmax()
    h_eq_max = np.where(h_segment == h_count_max)
    feature = h_filtered[h_eq_max].mean()
    return feature
```

Cluster 1, Cluster center: [109.69147208]

Cluster 2, Cluster center: [168.75520424]

Cluster 3, Cluster center: [19.9026114]

Cluster 4, Cluster center: [129.95525567]

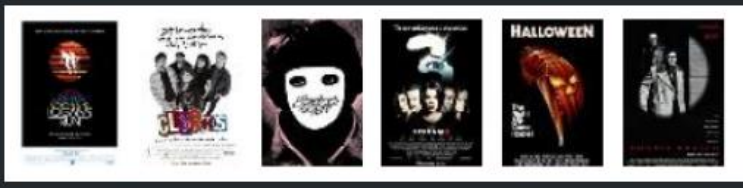
1.3.2 聚类改进 – HoG

```python
# 周晨昊
def build_feature_hog(img, method):
    assert method == 'h'
    hsv_img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    h_mean = np.mean(hsv_img[:, :, 0])
    h_hist = cv2.calcHist(
        [hsv_img[:, :, 0]], [0], None, [180], [0, 180]).flatten()
    gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    if gray_img.shape != (80, 55):  # resize to 80*55
        gray_img = cv2.resize(gray_img, (80, 55))
    # 参数设定部分略去
    hog = cv2.HOGDescriptor(winSize, blockSize, blockStride,
                            cellSize, nbins, derivAperture,
                            winSigma, histogramNormType,
                            L2HysThreshold, gammaCorrection,
                            nlevels, signedGradient)
    hog_feature = hog.compute(gray_img).flatten()
    feature = np.concatenate([hog_feature, h_hist])
    return feature
```

Cluster 0, Method h

Cluster 1, Method h

Cluster 2, Method h



Cluster 3, Method h

Cluster 4, Method h

1.3.2 聚类改进 – 局部滑窗直方图

```python
# 占佳豪
def color_histogram(image, num_regions, bins=(8, 8, 8)):
    # 将图像划分为 num_regions x num_regions 个局部区域
    h, w, _ = image.shape
    region_h = h // num_regions
    region_w = w // num_regions
    # 初始化局部颜色直方图
    local_histograms = []
    # 遍历每个局部区域
    for i in range(num_regions):
        for j in range(num_regions):
            # 提取当前局部区域的图像
            region = image[i * region_h : (i + 1) * region_h, j * region_w : (j + 1) * region_w]
            # 计算当前局部区域的颜色直方图
            hist = cv2.calcHist([region], [0, 1, 2], None, bins, [0, 180, 0, 256, 0, 256])
            hist = cv2.normalize(hist, hist).flatten()
            # 将局部颜色直方图添加到列表中
            local_histograms.append(hist)
    # 将所有局部颜色直方图拼接为一个特征向量
    feature_vector = np.concatenate(local_histograms)
    return feature_vector
```

1.3.2 聚类改进 – H的环结构

- 使用三角函数上的均值代替角度上的均值

```python
# 刘和金
def circular_mean(img):
    hsv_image = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hue_channel = hsv_image[:,:,0]

    radians = np.deg2rad(hue_channel)
    sin_mean = np.mean(np.sin(radians))
    cos_mean = np.mean(np.cos(radians))
    mean_angle = np.arctan2(sin_mean, cos_mean)
    mean_hue = np.rad2deg(mean_angle)
    return mean_hue
```

Cluster 0, Method circular_mean_h, Cluster center:

Cluster 1, Method circular_mean_h, Cluster center:

Cluster 2, Method circular_mean_h, Cluster center:

Cluster 3, Method circular_mean_h, Cluster center:

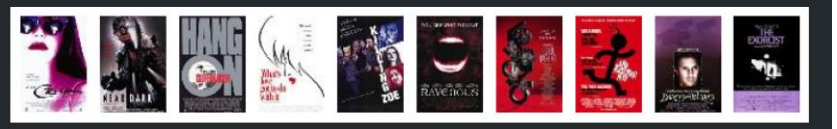
Cluster 4, Method circular_mean_h, Cluster center:

1.3.2 聚类改进 – V/S的影响

```python
# 赵凌哲
def build_feature(img, method):
    if method == 'h_plus':
        mask0 = np.ones_like(h)
        mask1 = np.ones_like(h)
        mask0[v<0.5 * 255] = 0
        mask1[s<0.3 * 255] = 0
        mask = mask0*mask1
        # 用遮罩去掉低饱和度区域和低亮度区域的影响
        #（进一步改进可以进行加权，而不是简单设为0）
        h0 = h
        h1 = (h+90)%180
        # h1的目的是将环整体旋转180°
        # 减少红色分布在330-360，0-30两区域导致的问题
        h_plus0 = h0*mask
        h_plus1 = h1*mask
        length = mask.sum()
        if length == 0:
            return (0, 0)
        return (h_plus0.sum()/length, h_plus1.sum()/length)
```

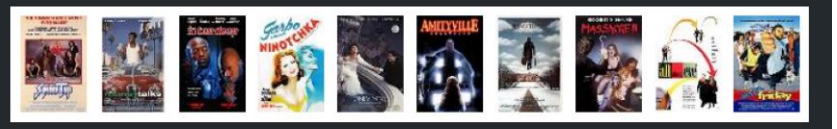
Cluster 0, Method h_plus, Cluster center: [46.69903754 80.02993764]

Cluster 1, Method h_plus, Cluster center: [153.76222244 22.86609717]

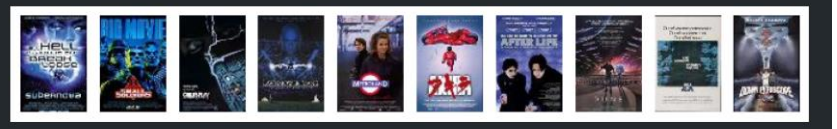


Cluster 2, Method h_plus, Cluster center: [20.75474465 101.50050693]

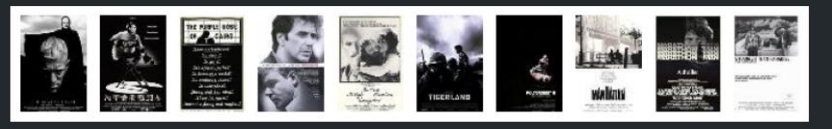
Cluster 3, Method h_plus, Cluster center: [76.51878872 57.47740881]



Cluster 4, Method h_plus, Cluster center: [107.43244695 29.41095218]



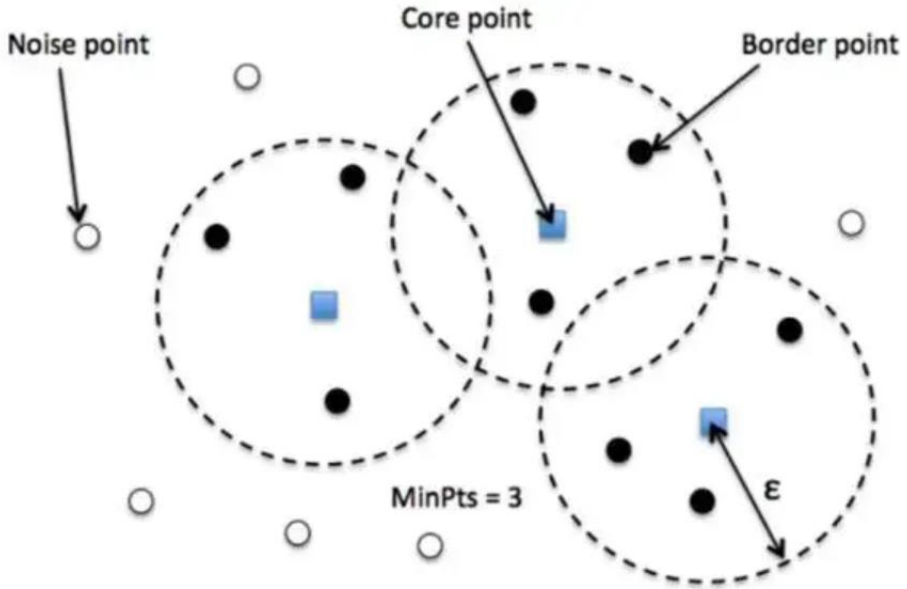
Cluster 5, Method h_plus, Cluster center: [0. 0.]

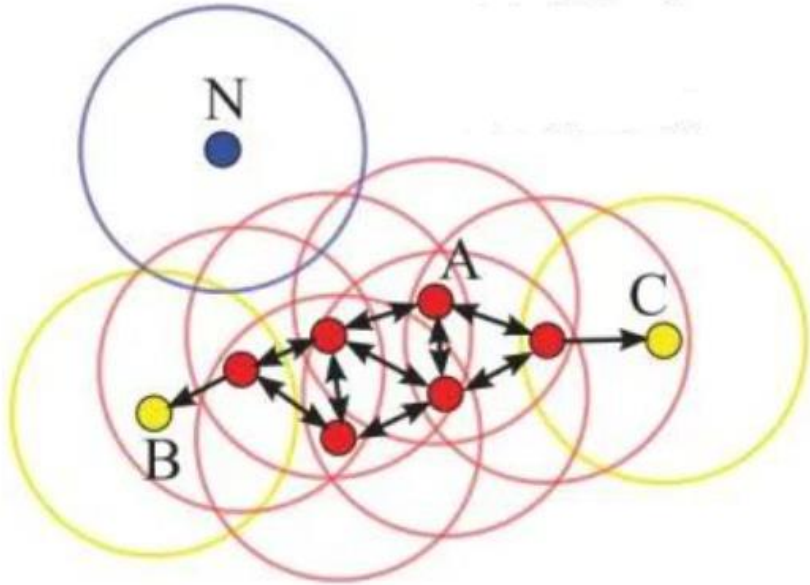


1.3.2 聚类改进 – DBSCAN

```
# 王开
from sklearn.cluster import dbscan
def build_feature_new(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    h = hsv[:, :, 0]
    return (np.mean(h), 1.1 *  np.std(h))
def run_kmeans_and_plot_new(n_clusters):
    ary = [build_feature_new(poster_i) for poster_i in posters]
    features = np.array(ary)
    core_samples,cluster_ids = dbscan(features, eps = 4,
min_samples=9)
    for i in range(-1, max(cluster_ids) + 1):
        if (i == -1) : print("[NOISE]", end = "")
        print(f'Cluster {i + 1}, DBSCAN of h, Core Sample:
{core_samples[i]}')
        idx_to_show = np.where(cluster_ids == i)[0][:10]
        posters_to_show = [posters[j] for j in idx_to_show]
        show_imgs(posters_to_show)
n_cluster_h = 7
run_kmeans_and_plot_new(n_cluster_h)
```

1.3.2 聚类改进 – DBSCAN



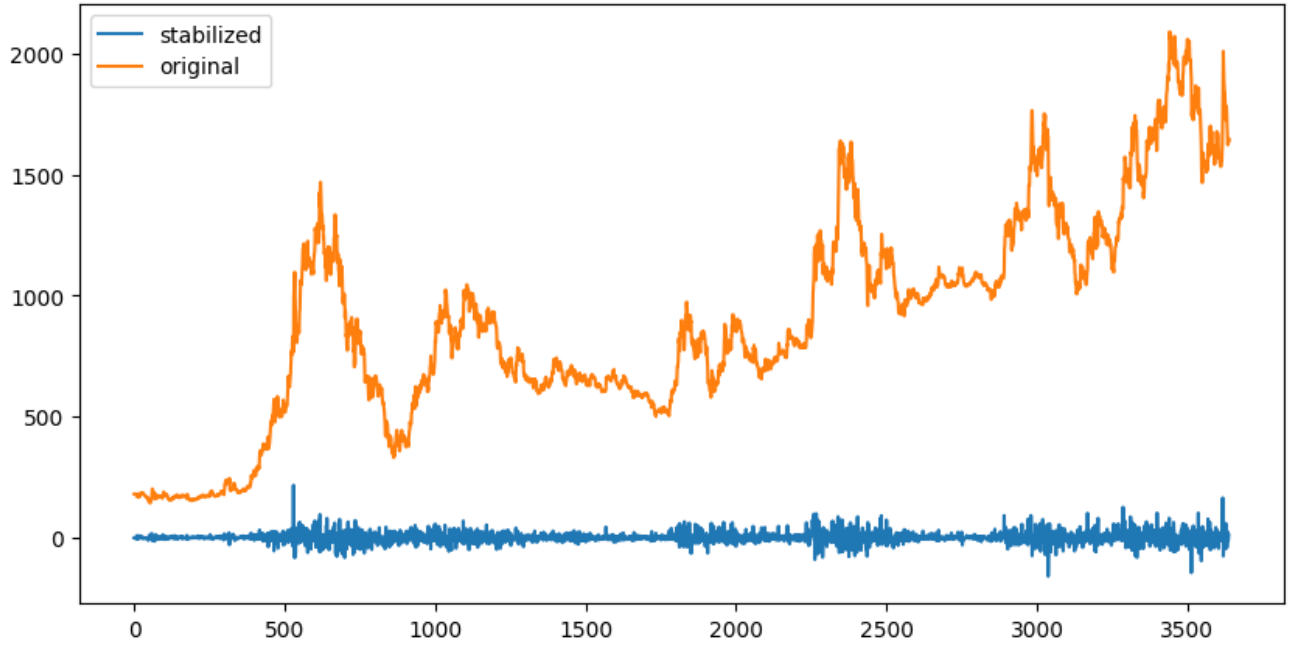


2.1 成分分解

```
choice = 'additive'
decomposition = seasonal_decompose(x=stock, model=choice, period=30)
plt.figure(figsize=(10, 10))
plt.subplot(4, 1, 1)
plt.title('PA Stock High')
plt.ylabel('Origin')
plt.plot(stock)
plt.subplot(4, 1, 2)
plt.plot(decomposition.trend)
plt.ylabel('Trend')
plt.subplot(4, 1, 3)
plt.plot(decomposition.seasonal)
plt.ylabel('Seasonal')
plt.subplot(4, 1, 4)
plt.plot(decomposition.resid)
plt.ylabel('Resid')
plt.xlabel('Date')
plt.show()
```

2.2 平稳化

```
d = np.array(stock)
while adfuller(d)[1] >= 0.05:
    d = np.diff(d)
plt.figure(figsize=(10, 5))
plt.plot(d, label='stabilized')
plt.plot(stock, label='original')
plt.legend()
plt.show()
```



2.3.1 均值与标准差

- 滑动窗口操作：pd.Series.rolling
- https://pandas.pydata.org/docs/reference/api/pandas.Series.rolling.html#pandas.Series.rolling

```
window = 30
rolling_mean = stock.rolling(window=window).mean()
rolling_std = stock.rolling(window=window).var() ** .5
plt.figure(figsize=(10, 5))
plt.plot(stock,label='PA')
plt.plot(rolling_mean, label='Rolling Mean')
plt.plot(rolling_std, label='Rolling Std')
plt.legend()
plt.show()
```

2.3.2 指数平滑

```
alpha = 0.03
beta = 0.06
n = 30
ls = [0]
bs = [0]
preds = []
for i in range(len(stock) - n):
    ls.append(alpha * stock[i] + (1 - alpha) * (ls[-1] + bs[-1]))
    bs.append(beta * (ls[-1] - ls[-2]) + (1 - beta) * bs[-1])
    preds.append(ls[-1] + n * bs[-1])
plt.figure(figsize=(10, 5))
plt.plot(stock, label='PA')
plt.plot(range(n, len(stock)), preds, label='smoothed')
plt.legend()
plt.show()
```