HW06

碩一資工 吳承翰 0856105

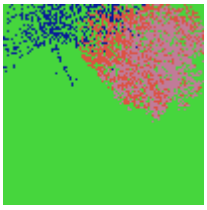
code:https://github.com/chiha8888/NCTU-ML-class/tree/master/HW06

1. You need to make videos or GIF images to show the clustering procedure (visualize the cluster assignments of data points in each iteration, colorize each cluster with different colors) of your kernel k-means, spectral clustering (both normalize cut and ratio cut) program.
2. In addition to cluster data into 2 clusters, try more clusters (e.g. 3 or 4) and show your results.

Kernel k-means:





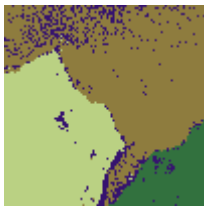


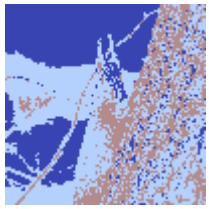
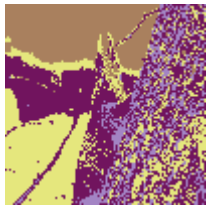

from left to right: k=2,k=3,k=4 (k=#cluster)
All using gamma_s=gamma_c=0.001. Initial centers was picked by k-means++.

Unnormalized spectral clustering:







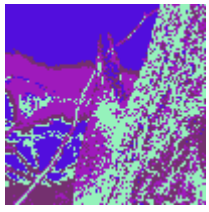



Normalized spectral clustering:







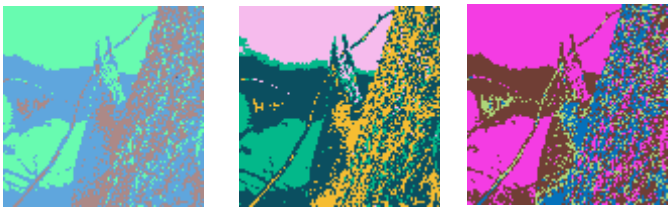
3.For the initialization of k-means clustering used in kernel k-means and spectral clustering (both normalize cut and ratio cut), try different ways and show corresponding results, e.g. k-means++.

I implement 3 ways for initialization:
“random”: initial center was choose from gaussian distribution with the mean and variance from the data points.
“pick”: randomly pick k data points as cluster centers.
“k_means_plusplus”: using k-means++ initial method.

Kernel k-means:

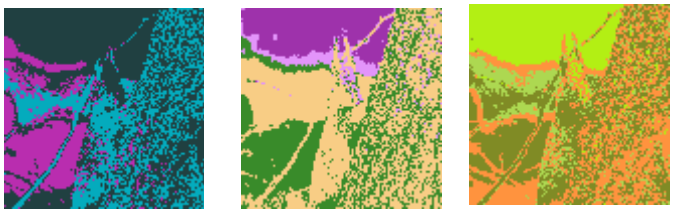
from left to right: “random”,”pick”,”k_means_plusplus”
All with k=3, gamma_s=gamma_c=0.001

Unnormalized spectral clustering:



from left to right: “random”,”pick”,”k_means_plusplus”
All with k=4, gamma_s=gamma_c=0.001

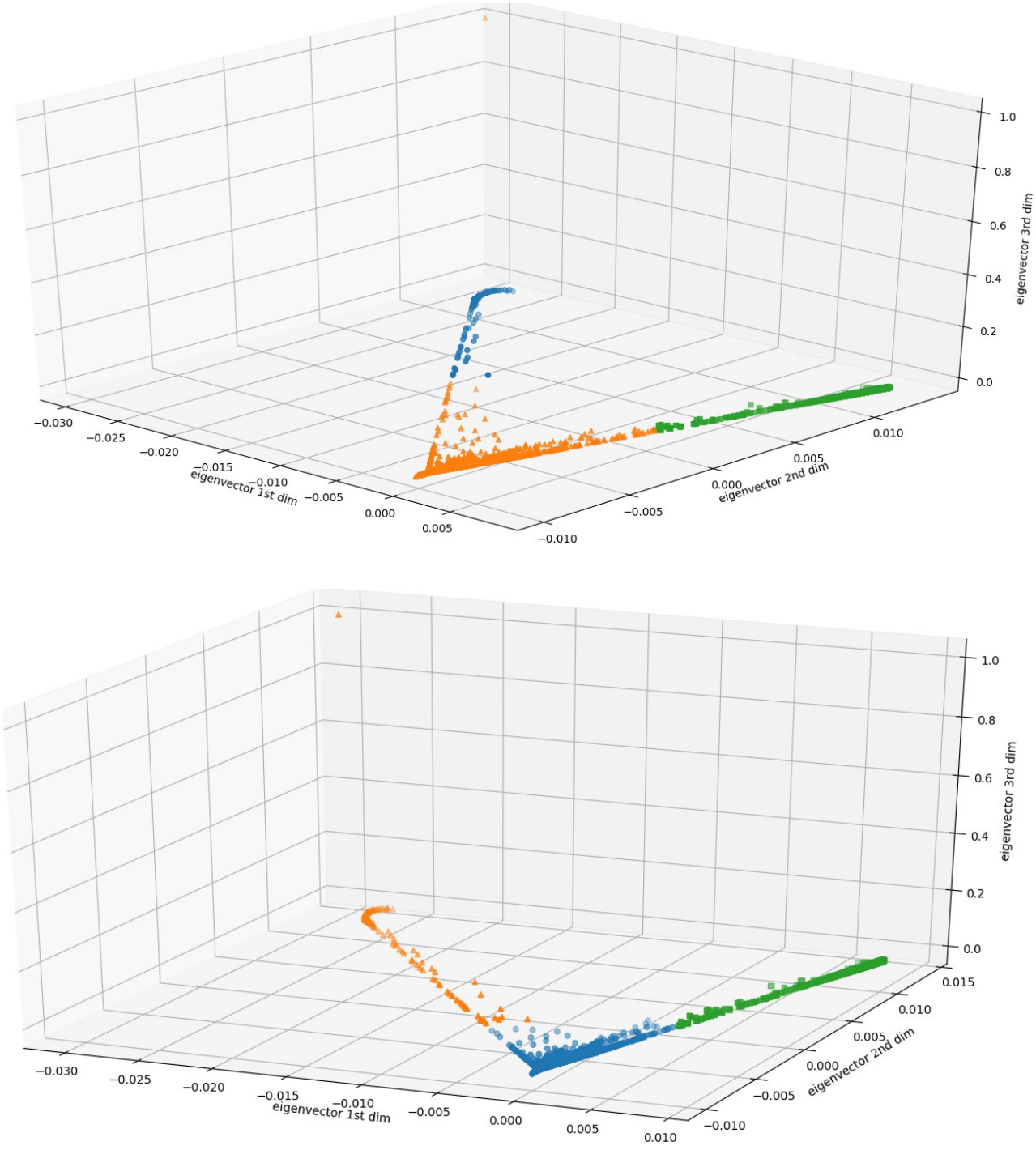
Normalized spectral clustering:



from left to right: “random”,”pick”,”k_means_plusplus”
All with k=4, gamma_s=gamma_c=0.001

4.For spectral clustering (both normalize cut and ratio cut), you can see if data points within the same cluster do have the same coordinates in the eigenspace of graph Laplacian. You should plot the result and discuss it in the report.

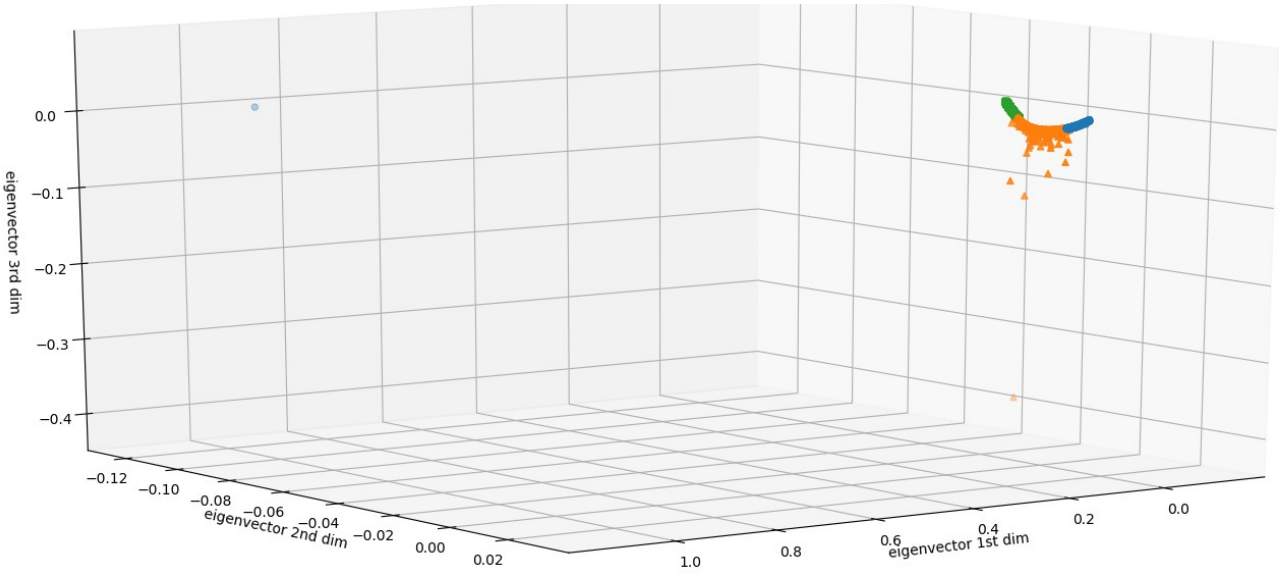
Let me discuss under the condition of k=3 of image 1 (with gamma_s=gamma_c=0.001):

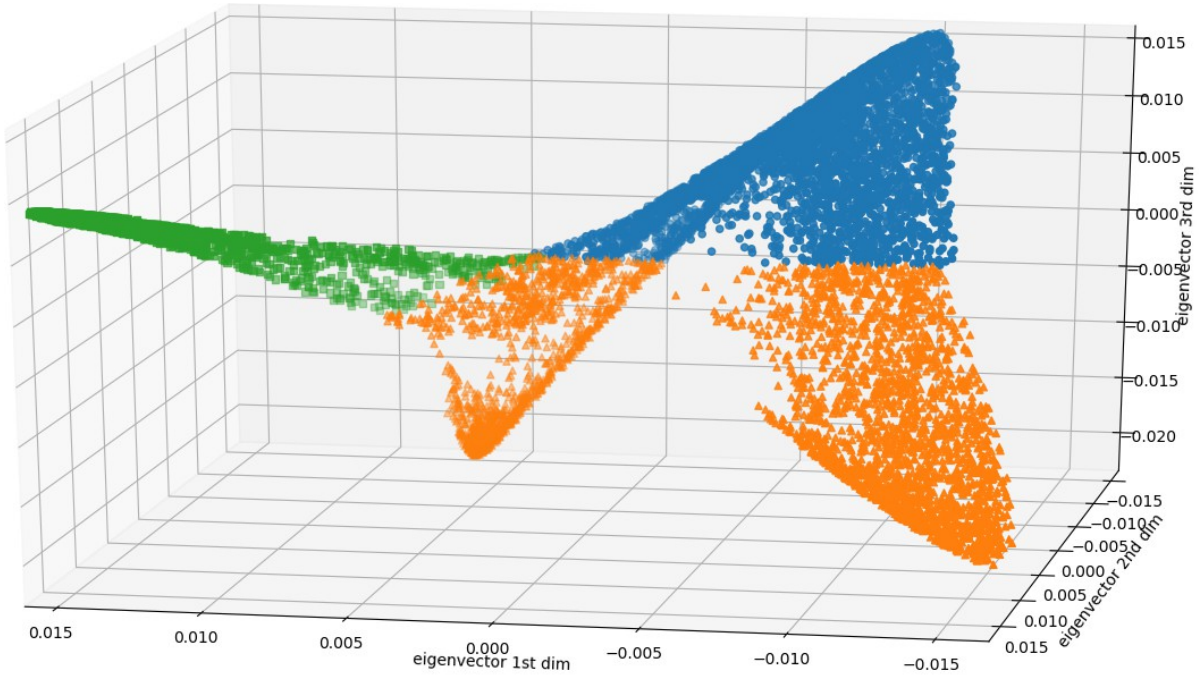


upper picture: Unnormalized spectral clsutering
lower picture: Normalized spectral clustering

I found that:
1. 1st dim eigenvector range(-0.030~0.008) > 2nd dim eigenvector range(-0.010~0.012) >3rd dim eigenvector range(excluding the extremum point)
2. the summation of each dim of eigenvector summation = 0
3. both unnormalized and normalized picture looks almost the same

Let me discuss under the condition of k=3 of image 2 (with gamma_s=gamma_c=0.001):





upper picture: Unnormalized spectral clsutering
lower picture: Normalized spectral clustering
I found that:
1. 1st dim eigenvector range & dim eigenvector range & 3rd dim eigenvector range doesn't have clearly relation
2. the summation of each dim of eigenvector summation = 0
3. both unnormalized and normalized picture is very dislike

5.code discussion:

kernel k-means:

```
# set parameters
img_path='image1.png'
image_flat,HEIGHT,WIDTH=imread(img_path)
gamma_s=0.001
gamma_c=0.001
k=3  # k clusters
k_means_initType='k_means_plusplus'
gif_path=os.path.join('GIF','{}_{}Clusters_{}'.format(img_path.split('.')[0],k,'kernel k-means.gif'))

Gram=precomputed_kernel(image_flat,gamma_s,gamma_c)
belongings,segments=kmeans(Gram,k,HEIGHT,WIDTH,initType=k_means_initType)
save_gif(segments,gif_path)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
def precomputed_kernel(X,gamma_s=1,gamma_c=1):
    '''
    kernel function: k(x,x')= exp(-r_s*||S(x)-S(x')||**2)* exp(-r_c*||C(x)-C(x')||**2)
    :@param X: (H*W=10000,rgb=3) ndarray
    :@param gamma_s: gamma of spacial
    :@param gamma_c: gamma of color
    :@return : (10000,10000) ndarray
    '''
    n=len(X)
    # S(x) spacial ingormation
    S=np.zeros((n,2))
    for i in range(n):
        S[i]=[i//100,i%100]
    K=squareform(np.exp(-gamma_s*pdist(S,'sqeuclidean')))*squareform(np.exp(-gamma_c*pdist(X,'sqeuclidean')))

    return K
```

Use preocomputed_kernel() to compute a 10000*10000 gram matrix, then using kmeans() to do cluster.

Unormalized spectral clustering:

```
# set parameters
img_path='image2.png'
image_flat,HEIGHT,WIDTH=imread(img_path)
gamma_s=0.001
gamma_c=0.001
k_means_initType='k_means_plusplus'
k=3  # k clusters
gif_path=os.path.join('GIF','{}_{}Clusters_{}'.format(img_path.split('.')[0],k,'unnormalized.gif'))

# similarity matrix
W=precomputed_kernel(image_flat,gamma_s,gamma_c)
# degree matrix
D=np.diag(np.sum(W,axis=1))
L=D-W

'''
eigenvalue,eigenvector=np.linalg.eig(L)
np.save('{}_eigenvalue_{:.3f}_{:.3f}_unnormalized'.format(img_path.split('.')[0],gamma_s,gamma_c),eigenvalue)
np.save('{}_eigenvector_{:.3f}_{:.3f}_unnormalized'.format(img_path.split('.')[0],gamma_s,gamma_c),eigenvector)
'''

eigenvalue=np.load('{}_eigenvalue_{:.3f}_{:.3f}_unnormalized.npy'.format(img_path.split('.')[0],gamma_s,gamma_c))
eigenvector=np.load('{}_eigenvector_{:.3f}_{:.3f}_unnormalized.npy'.format(img_path.split('.')[0],gamma_s,gamma_c))
sort_index=np.argsort(eigenvalue)
# U
U=eigenvector[:,sort_index[1:1+k]]

# k-means
belonging,segments=kmeans(U,k,HEIGHT,WIDTH,initType=k_means_initType)

save_gif(segments,gif_path)
if k==3:
    plot_eigenvector(U[:,0],U[:,1],U[:,2],belonging)
```

Get the eigenvalue & eigenvector of L, then using the 2~(k+1)th column eigenvector with respect to the 2nd ~(k+1)th smallest eigenvalue

Normalized spectral clustering:

```
# set parameters
img_path='image2.png'
image_flat,HEIGHT,WIDTH=imread(img_path)
gamma_s=0.001
gamma_c=0.001
k=3  # k clusters
k_means_initType='k_means_plusplus'
gif_path=os.path.join('GIF','{}_{}Clusters_{}'.format(img_path.split('.')[0],k,'normalized.gif'))

# similarity matrix
W=precomputed_kernel(image_flat,gamma_s,gamma_c)
# degree matrix
D=np.diag(np.sum(W,axis=1))
L=D-W
D_inverse_square_root=np.diag(1/np.diag(np.sqrt(D)))
L_sym=D_inverse_square_root@L@D_inverse_square_root

'''
eigenvalue,eigenvector=np.linalg.eig(L_sym)
np.save('{}_eigenvalue_{:.3f}_{:.3f}_normalized'.format(img_path.split('.')[0],gamma_s,gamma_c),eigenvalue)
np.save('{}_eigenvector_{:.3f}_{:.3f}_normalized'.format(img_path.split('.')[0],gamma_s,gamma_c),eigenvector)
'''

eigenvalue=np.load('{}_eigenvalue_{:.3f}_{:.3f}_normalized.npy'.format(img_path.split('.')[0],gamma_s,gamma_c))
eigenvector=np.load('{}_eigenvector_{:.3f}_{:.3f}_normalized.npy'.format(img_path.split('.')[0],gamma_s,gamma_c))
sort_index=np.argsort(eigenvalue)
# U:(n,k)
U=eigenvector[:,sort_index[1:1+k]]
# T:(n,k) each row with norm 1
sums=np.sqrt(np.sum(np.square(U),axis=1)).reshape(-1,1)
T=U/sums

# k-means
belonging,segments=kmeans(T,k,HEIGHT,WIDTH,initType=k_means_initType)

save_gif(segments,gif_path)
if k==3:
    plot_eigenvector(U[:,0],U[:,1],U[:,2],belonging)
```

Get the eigenvalue & eigenvector of L_sym, then using the 2~(k+1)th column eigenvector with respect to the 2nd ~(k+1)th smallest eigenvalue