

```

1  #
2  # svd_image.py
3  #
4
5  import matplotlib.pyplot as plt
6
7  import numpy as np
8  import numpy.linalg as LA
9  import argparse
10 import scipy.misc as misc
11 from sklearn import decomposition
12 import time
13
14
15
16 def reconstr_matrix(U,D,V,k):
17     rec_mat=np.dot(U[:, :k], np.dot(D[:k, :k], V[:k, :]))
18     return rec_mat
19
20
21 def rgb2gray(rgb):
22
23     if(len(rgb.shape)>2):
24         r, g, b = rgb[:, :, 0], rgb[:, :, 1], rgb[:, :, 2]
25         L = 0.2989 * r + 0.5870 * g + 0.1140 * b
26     else:
27         L = rgb
28     return L
29
30 def ParseArguments():
31     parser = argparse.ArgumentParser(description="Project ")
32     parser.add_argument('--image', default="", required=True, help='Color image (default: %(default)s)')
33     parser.add_argument('--alg', default="pca", required=False, help='pca, nmf (default: %(default)s)')
34     args = parser.parse_args()
35     return args.image,args.alg
36
37
38 image_file, alg = ParseArguments()
39
40 #image = misc.imread(image_file ,mode="RGB")
41
42 image = plt.imread(image_file)
43
44 M=rgb2gray(image)
45
46
47
48 #glowna czesc programu: wykonaj svd
49
50
51 print("Calculating SVD ...", end="", flush=True)
52 start_time = time.time()
53 U,d,V=LA.svd(M)
54 print("\t\t took %s seconds " % round((time.time() - start_time),5))
55
56
57
58
59 h=M.shape[0]
60 w=M.shape[1]
61
62 #d jest wektorem, zamieniamy w macierz diagonalna
63 D=np.diag(d)
64
65 ile=4;
66 nr=1;
67
68 how_much_rec=1;
69
70 rr=reconstr_matrix(U,D,V,how_much_rec)
71
72 error_x = []
73 error_y = []
74 eigen_sum = []
75
76
77
78
79
80

```

```

81 for i in range(0,ile):
82     for j in range(0, ile):
83         fig1=plt.figure(1);
84         plt.subplot(ile,ile,nr);
85
86         #zrekonstruuuj macierz uzywajac how_much_rec wektorow wlasnych
87         rr=reconstr_matrix(U,D,V,how_much_rec)
88         #liczymy bledy
89         error_x.append(how_much_rec)
90         err = ((M-rr)**2).sum();
91         error_y.append(err)
92         eigen_sum.append((d[how_much_rec:len(d)]**2).sum())
93         #rysuj rr
94         plt.imshow(rr, cmap=plt.get_cmap('gray'))
95         plt.title("ile = " + str(how_much_rec))
96         plt.axis('off')
97         how_much_rec+=2;
98
99         nr=nr+1
100     fig1.suptitle("PCA reconstructed")
101
102
103     nr=1;
104     ile=4
105     how_much_rec=0
106     for i in range(0,ile):
107         for j in range(0, ile):
108             how_much_rec+=1;
109             fig2=plt.figure(2);
110             plt.subplot(ile,ile,nr);
111             k=how_much_rec
112             rr_eigenimage = np.dot(U[:,k:(k+1)],V[k:(k+1),:])
113             plt.imshow(rr_eigenimage, cmap=plt.get_cmap('gray'))
114             plt.title("i = " + str(how_much_rec))
115             plt.axis('off')
116             nr=nr+1
117
118     fig2.suptitle("PCA eigenimages")
119
120
121
122
123     #osobny rys. z bledami
124     plt.figure(3)
125     plt.plot(error_x,np.zeros(len(error_x)), color='red');
126     plt.plot(error_x,error_y, color='blue')
127
128     plt.plot(error_x,np.zeros(len(error_x)), color='red');
129     plt.plot(error_x,eigen_sum, color='green')
130
131
132     plt.figure(4)
133     plt.imshow(M, cmap=plt.get_cmap('gray'))
134     plt.title("Original (grayscale)")
135
136
137
138
139     #NMF:
140
141     # ~ plt.figure(5)
142     # ~ plt.title("NMF")
143     # ~ M_recons=np.dot(W,H)
144     # ~ plt.imshow(M_recons, cmap=plt.get_cmap('gray'))
145
146
147     if(alg=="nmf" or alg=="NMF"):
148
149         #p_red=min(int(h/2),int(w/2))
150         p_red=min(int(h),int(w))
151
152         print("Calculating NMF...", end="", flush=True)
153         start_time = time.time()
154         model = decomposition.NMF(n_components=p_red, init='random', random_state=0)
155         W=model.fit_transform(M)
156         H=model.components_
157         print("\t\t took %s seconds " % round((time.time() - start_time),5))
158
159
160     nr=1;

```

```

161 ile=4
162 how_much_rec=0
163 for i in range(0,ile):
164     for j in range(0, ile):
165         how_much_rec+=8;
166         fig5=plt.figure(5);
167         plt.subplot(ile,ile,nr);
168         k=how_much_rec
169         rr_eigenimage = np.dot(U[:,k:(k+1)],V[k:(k+1),:])
170         M_recons=np.dot(W[:,0:k],H[0:k,:])
171         plt.imshow(M_recons, cmap=plt.get_cmap('gray'))
172         plt.title("i = " + str(how_much_rec))
173         plt.axis('off')
174         nr=nr+1
175
176 fig5.suptitle("NMF reconstructed")
177
178 nr=1;
179 ile=4
180 how_much_rec=0
181 for i in range(0,ile):
182     for j in range(0, ile):
183         how_much_rec+=1;
184         fig6=plt.figure(6);
185         plt.subplot(ile,ile,nr);
186         k=how_much_rec
187         rr_eigenimage = np.dot(U[:,k:(k+1)],V[k:(k+1),:])
188         M_recons=np.dot(W[:,(k-1):k],H[(k-1):k,:])
189         plt.imshow(M_recons, cmap=plt.get_cmap('gray'))
190         plt.title("i = " + str(how_much_rec))
191         plt.axis('off')
192         nr=nr+1
193
194 fig6.suptitle("NMF eigenimages")
195
196
197
198
199
200
201
202 plt.show()
203

```