プログラム

実行環境と用いた言語・ライブラリを以下の表 1 に示す。

表 1: プログラムの実行環境

OS	:	Microsoft Windows 10 Pro (64bit)
CPU	:	Intel(R) Core(TM) i5-4300U
RAM	:	4.00 GB
使用言語	:	Python3.6
可視化	:	matplotlib ライブラリ

Listings 1: assignment2.py

```
# -*- coding: utf-8 -*-


import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import sqrtm


def generate_data(n=1000):
    x = np.concatenate([
        np.random.rand(n, 1),
        np.random.randn(n, 1)
        ], axis=1)
    x[0, 1] = 6   # outlier

    # Standardization
    x = (x - np.mean(x, axis=0)) / np.std(x, axis=0)

    M = np.array([[1, 3], [5, 3]])
    x = x.dot(M.T)
    x = np.linalg.inv(sqrtm(np.cov(x, rowvar=False))).dot(x.T).T
    return x


def metric_s4(s, derivative=0):
    if derivative == 0:
        met = s**4
    elif derivative == 1:
        met = 4*s**3
    elif derivative == 2:
        met = 12*s**2
    else:
        raise ValueError('Derivatives more than second are not defined. But
```

```
        the input was: {}'.format(derivative))
34      return met
35
36
37  def metric_logcosh(s, derivative=0):
38      if derivative == 0:
39          met = np.log(np.cosh(s))
40      elif derivative == 1:
41          met = np.tanh(s)
42      elif derivative == 2:
43          met = 1 - np.tanh(s)**2
44      else:
45          raise ValueError('Derivatives more than second are not defined. But
        the input was: {}'.format(derivative))
46      return met
47
48
49  def metric_exp(s, derivative=0):
50      if derivative == 0:
51          met = - np.exp(-(s**2)/2)
52      elif derivative == 1:
53          met = s*np.exp(-(s**2)/2)
54      elif derivative == 2:
55          met = (1 - s**2) * np.exp(-(s**2)/2)
56      else:
57          raise ValueError('Derivatives more than second are not defined. But
        the input was: {}'.format(derivative))
58      return met
59
60
61  def normalize(b):
62      b = b / np.linalg.norm(b)
63      if b[0] < 0:
64          b *= -1
65      return b
66
67
68  def update(b, x_whitened, metric):
69      n = len(x_whitened)
70      s = x_whitened.dot(b)
71      b_new = (
72          (np.mean(metric(s, 2))) * b
73          - (1/n) * np.sum(x_whitened * metric(s, 1)[:, np.newaxis], axis=0)
74          )
75      return b_new
76
77
```

```
78 def train(x_whitened, metric, max_iter=100, eps=1e-4, n=5):
79     d = x_whitened.shape[1]
80 
81     # initialize b
82     b = np.random.randn(d)
83     b = normalize(b)
84 
85     b_olds = []
86     for i in range(max_iter):
87         b_old = b.copy()
88         b = update(b, x_whitened, metric)
89         b = normalize(b)
90 
91         # if converge, break
92         if len(b_olds) < n:
93             b_olds.append(b_old)
94         else:
95             b_olds[:-1] = b_olds[1:]
96             b_olds[-1] = b_old
97             b_olds = np.array(b_olds)
98             diffs = np.sqrt(np.sum((b_olds - b)**2, axis=1))
99             if diffs.max() < eps:
100                 break
101     n_iter = i + 1
102     return b, n_iter
103 
104 
105 def main():
106     # settnigs
107     n_sample = 1000
108     #metric, metric_name = metric_s4, 's4'
109     #metric, metric_name = metric_logcosh, 'logcosh'
110     metric, metric_name = metric_exp, 'exp'
111 
112     offset = 1.0
113     np.random.seed(0)
114     scatter_path =
        f'../figures/assignment2_result_{metric_name}_n{n_sample}_scatter.png'
115     hist_path =
        f'../figures/assignment2_result_{metric_name}_n{n_sample}_hist.png'
116 
117 
118     # load data
119     x = generate_data(n_sample)
120     #x_whitened = (x - np.mean(x, axis=0)) / np.std(x, axis=0)
121 
122 
```

```
123     # train
124     b, n_iter = train(x, metric, max_iter=1000, eps=1e-4, n=5)
125
126
127     # result
128     print(f'Metric: {metric_name}')
129     print(f'#Data: {n_sample} \t #Iter: {n_iter}')
130     print('b: {} (norm = {})'.format(b, np.linalg.norm(b)))
131
132
133     # plot scatter
134     scale = 1e3
135     x0_max, x0_min = x[:, 0].max(), x[:, 0].min()
136     x1_max, x1_min = x[:, 1].max(), x[:, 1].min()
137
138
139     plt.scatter(x[:, 0], x[:, 1], color='royalblue', s=8)
140     plt.quiver(
141         0, 0, b[0]*scale, b[1]*scale,
142         color='darkcyan', angles='xy', scale_units='xy', scale=6.5,
143         )
144     plt.quiver(
145         0, 0, -b[0]*scale, -b[1]*scale,
146         color='darkcyan', angles='xy', scale_units='xy', scale=6.5,
147         )
148     plt.xlim([x0_min-offset, x0_max+offset])
149     plt.ylim([x1_min-offset, x1_max+offset])
150     plt.savefig(scatter_path)
151     plt.show()
152
153
154     # plot hist
155     x_casted = x.dot(b)
156     plt.hist(x_casted, bins=50, rwidth=0.9)
157     plt.savefig(hist_path)
158     plt.show()
159
160
161 if __name__ == '__main__':
162     main()
```