# 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


def generate_sample(n):
    sample = (np.random.randn(n) + np.where(np.random.rand(n) > 0.3, 2., -2.))
    return sample


def gauss_dist(x, mu, sigma):
    d = mu.shape[0]
    phi = (1/(2 * np.pi * sigma**2)**(d/2)) * np.exp(-(1/(2*sigma**2) * (x - mu)**2))
    return phi


def calc_w_phi(x, w, mu, sigma):
    phi_list = []
    m = mu.shape[0]
    for j_dash in range(m):
        phi_list.append(gauss_dist(x, mu[j_dash], sigma[j_dash]))
    phi_list = np.array(phi_list).T
    w_phi = w * phi_list
    return w_phi


def gaussian_mixture_model(x, w, mu, sigma):
    w_phi = calc_w_phi(x, w, mu, sigma)
    q = np.sum(w_phi, axis=1)
    return q


def calc_eta(x, w, mu, sigma):
    w_phi = calc_w_phi(x, w, mu, sigma)
    w_phi_sum = w_phi.sum(axis=1)
    eta = w_phi
    for j in range(eta.shape[1]):
        eta[:, j] /= w_phi_sum
    return eta


def update_w(j, eta):
    w_new = np.mean(eta[:, j])
    return w_new


def update_mu(j, eta, x):
    mu_new = (np.sum(eta[:, j] * x)) / np.sum(eta[:, j])
    return mu_new


def update_sigma(j, eta, x, mu):
    d = mu.shape[1]
    sigma_new_2 = (1/d) * (np.sum(eta[:, j] * (x - mu[j])**2))/
    np.sum(eta[:, j])
    sigma_new = np.sqrt(sigma_new_2)
    return sigma_new


def update(x, w, mu, sigma):
    m = w.shape[0]
    eta = calc_eta(x, w, mu, sigma)
    w_new = np.empty_like(w)
    mu_new = np.empty_like(mu)
    sigma_new = np.empty_like(sigma)
    for j in range(m):
        w_new[j] = update_w(j, eta)
        mu_new[j] = update_mu(j, eta, x)
        sigma_new[j] = update_sigma(j, eta, x, mu)
    return w_new, mu_new, sigma_new


def calc_Q(x, w, mu, sigma):
    w_phi = calc_w_phi(x, w, mu, sigma)
    eta = calc_eta(x, w, mu, sigma)
    Q = np.sum(np.sum(eta * np.log(w_phi), axis=1))
    return Q


def train(x, w, mu, sigma, eps=1e-4, n_converge=5, max_iter=100, show_log=True,):
    n = x.shape[0]
    QbyN_list = []
    for idx_iter in range(max_iter):
        w, mu, sigma = update(x, w, mu, sigma)
        Q = calc_Q(x, w, mu, sigma)
        QbyN = Q/n
        if show_log:
            print(’Iter: {} \t Q: {:.1f} \t Q/n: {:.4f}’.format(idx_iter+1, Q, QbyN))
        if len(QbyN_list) < n_converge:
            QbyN_list.append(QbyN)
        else:
            QbyN_list = QbyN_list[1:] + [QbyN]
            if (max(QbyN_list) - min(QbyN_list)) < eps:
                break
    n_iter = idx_iter + 1
    return w, mu, sigma, n_iter


def main():
    # settings
    n = 10000
    m = 2
    d = 1
    result_path = ’../figures/assignment2_result_n{}.png’.format(n)
    offset = 1.0
    np.random.seed(0)

    # data
    sample = generate_sample(n)

    # init params
    w = np.random.rand(m)
    w = w / w.sum()
    mu = (np.random.rand(m, d) - 0.5) * 6
    sigma = np.random.rand(m)

    print(’Init’)
    print(’w: {}’.format(w))
    print(’sigma: {}’.format(sigma))
    print(’mu: \n{}’.format(mu))
    print()

    # train
    w, mu, sigma, n_iter = train(
        sample, w, mu, sigma,
        eps=1e-4, n_converge=5, max_iter=100,
        show_log=True,
        )
    Q = calc_Q(sample, w, mu, sigma)

    # result
    print()
    print(’Result’)
    print(’n: {} \t Iter: {}’.format(n, n_iter))
    print(’Q: {} \t Q/n: {}’.format(Q, Q/n))
    print(’w: {} (sum = {})’.format(w, w.sum()))
    print(’sigma: {}’.format(sigma))
    print(’mu: \n{}’.format(mu))
    print()

    # plot
    x_axis = np.linspace(sample.min()-offset, sample.max()+offset, 100)
    q = gaussian_mixture_model(x_axis, w, mu, sigma)
    plt.plot(x_axis, q, color=’darkcyan’)
    plt.hist(sample, bins=50, normed=True, color=’lightblue’)
    plt.xlabel(’$x$’)
    plt.savefig(result_path)
    plt.show()


if __name__ == ’__main__’:
    main()
```