Assignment 8

EM アルゴリズムによって，2 つのガウス混合分布を推定した結果，表 1 のパラメータが得られた．また，この分布を利用して，データのクラス分けを行った結果，図 1 の結果が得られた．

	Class 1	Class 2
π	0.355	0.645
$\boldsymbol{\mu}$	(-1.273, -0.666)	(1.273, 0.666)
$\boldsymbol{\Sigma}$	0.0522 0.0272 0.0272 0.477	0.459 0.0624 0.0624 0.197

表 1 EM アルゴリズムで得られた推定値

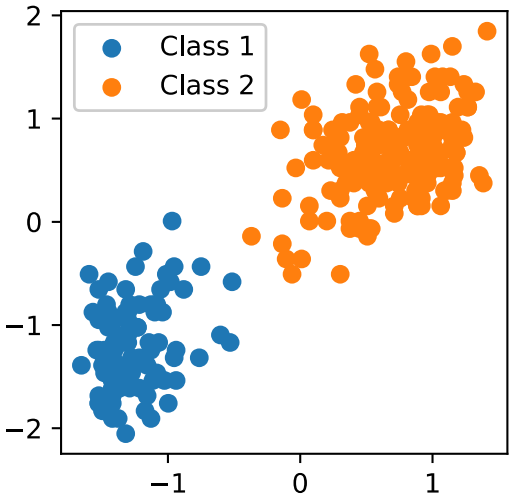

図 1 得られた推定値によるクラス分類

パラメータ推定に使用したソースコードの一部メソッド(fit, calc_probability, E_step, M_step)を図 2 に示す．なお，ソースコード全体は，添付する”Assignment8_code.py”に記載する．パラメータ推定は，fit メソッドに間欠泉のデータを与えて行った．

fit メソッドでは，平均値の更新量が小さくなるまで，E-step と M-step を繰り返している．calc_probability メソッドでは，入力 x を与えたときの各クラスに属する確率を戻り値としている．E_step メソッドは，calc_probability メソッドの同じ働きをしている．M_step メソッドでは，入力 x と x が各クラスに属する確率 pr を入力として受け取り，各種の推定値の更新を行っている．

```
1   def fit(self, x):
2     for t in range(self.max_itr):
3       old_mu = self.mu.copy()
4       pr = self.E_step(x)
5       if np.abs(old_mu - self.mu).mean() < self.eps:
6         return
7
8     def calc_probability(self, x):
9       prob = self.pi[:, None] * two_dim_gaussian(x, self.mu, self.sigma)
10      return prob / prob.sum(axis = 0)
11
12    def E_step(self, x):
13      return self.calc_probability(x)
14
15    def M_step(self, x, pr):
16      nk = pr.sum(axis = 1)
17      self.mu = (x[None, :, :]*pr[:, :, None]).sum(axis = 1)/ nk
18      for i in range(self.n_class):
19        square = (x - self.mu[i])[:, :, None] * ¥
                   (x - self.mu[i])[:, None, :]
20        prod = pr[i][:, None, None]
21        self.sigma[i] = (square * prod).sum(axis = 0) / nk[i]
22      self.pi = pr.mean(axis = 1)
```

図 1 パラメータ推定に使用したソースコード