9.2.2 混合分布のEMアルゴリズム

平成28年9月11日

概 要

PRMLの「9.2.2 混合分布のEMアルゴリズム」についての実装と考察

目 次

1 問題設定 **2**

2 アルゴリズム **2**

3 コード **3**

4 結果 **4**

5 まとめ **5**

1　問題設定

混合分布の EM アルゴリズムで分類を行う.

2　アルゴリズム

ここでは混合ガウス分布の EM アルゴリズムについて説明する.
まずデータ点 $\mathbf{x}_n$ $(n = 1, \ldots, N)$ が, 混合ガウス分布

$$p(\mathbf{x}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \quad (9.7)$$

に従うと仮定する.
EM アルゴリズムは, パラメータ $\pi_k, \boldsymbol{\mu}_k, \Sigma_k$ をデータ点から推測するものである.
まず (9.7) より対数尤度関数は

$$\ln p(X | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \right\} \quad (9.14)$$

となるが, これを $\boldsymbol{\mu}_k$ で微分すると

$$\sum_{n=1}^{N} \frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)} \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) = \mathbf{0} \quad (9.16)$$

となり

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n \quad (9.17)$$

を得る. ただし,

$$\gamma_{nk} = \frac{\pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j N(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)} \quad (9.13), \quad N_k = \sum_{n=1}^{N} \gamma_{nk} \quad (9.18)$$

で, γ_{nk} は, 混合要素 k がデータ点 $\mathbf{x}_n$ の観測を説明する度合いと見れ, 負担率と呼ばれる.
同様に, Σ_k も定まり,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \quad (9.19)$$

となる. そして, 混合係数 π_k であるが, 制約条件 $\sum_k \pi_k = 1$ があるので, ラグランジュの未定乗数法で

$$\ln p(X | \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \quad (9.20)$$

を π_k について微分することによって,

$$\pi_k = \frac{N_k}{N} \quad (9.22)$$

を得る.
EM アルゴリズムでは, 負担率とパラメータの更新を交互に行うことで, パラメータの推定を行う

のである.

ただ, EM アルゴリズムは計算量が多いので, まず K-means クラスタリングで大まかにクラスタリングをしてから, EM-アルゴリズムを走らせるという流れ.

EM アルゴリズム

1. まず, K 個の μ_k, Σ_k, π_k を導入する.(K-means クラスタリングの結果を使う)
2. (E ステップ) 負担率 γ_{nk} を計算する.
3. (M ステップ) 計算しなおした負担率を用いて, μ_k, Σ_k, π_k を計算する.
4. 2,3 を繰り返し, $\ln P(X|\mu, \Sigma, \pi)$ の値が収束したとき終了する.

3 コード

K-means クラスタリングによる分類のコード (K-means.py).

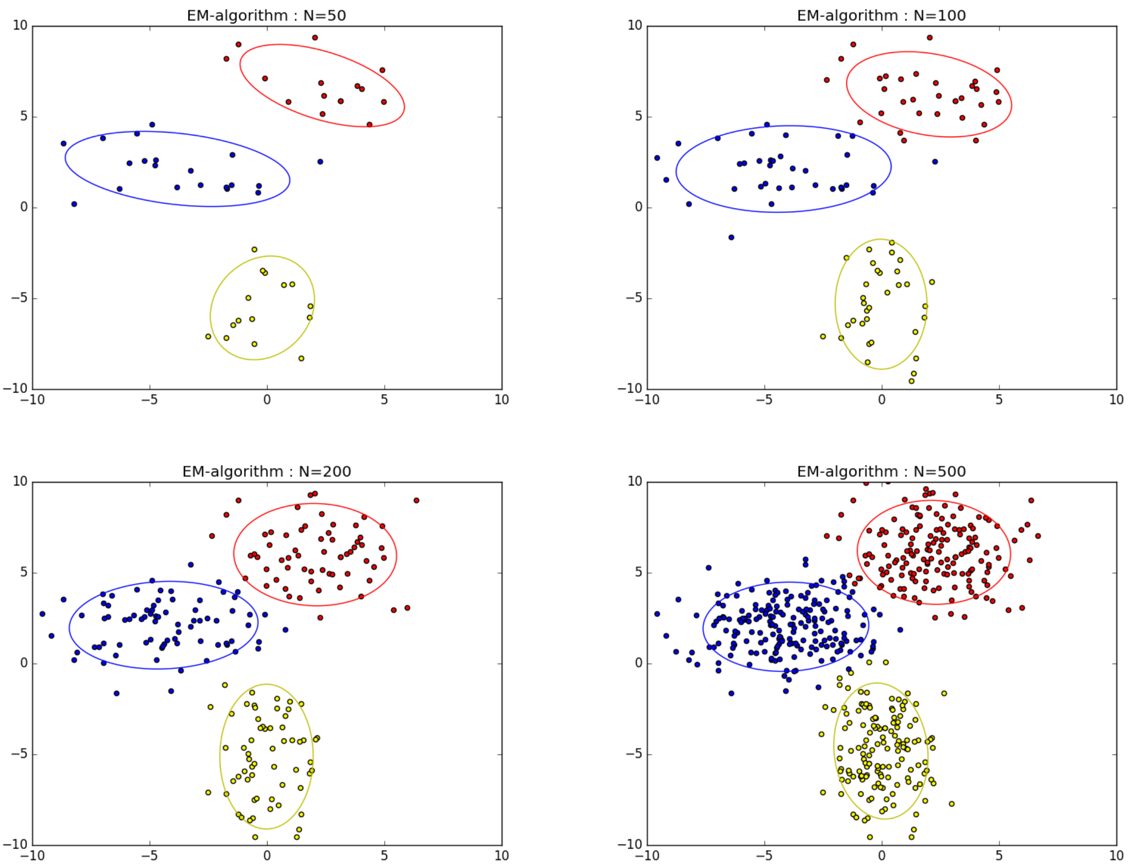
```
Pi=num[:]/N
sig=np.zeros((K,2,2))
for k in range(K):
  for n in C[k]:
    sig[k,:]+=outer(x[n,:]-mu[k,:],x[n,:]-mu[k,:])
  sig[k]/=num[k]
gamma=np.zeros((N,K))
ln_p=1
diff=1

while abs(diff)>10**-3:
  diff=ln_p

  for n in range(N):
    for k in range(K):
      dist[n,k]=gauss(x[n,:],mu[k,:],sig[k,:])
  #Eステップ
  for n in range(N):
    for k in range(K):
      gamma[n,k]=Pi[k]*dist[n,k]
    for k in range(K):
      tmp=0
      for j in range(K):
        tmp+=Pi[j]*dist[n,j]
      gamma[n,k]/=tmp
  #Mステップ
  num=np.zeros(K)
  for k in range(K):
    for n in range(N):
      num[k]+=gamma[n,k]

  mu=np.zeros((K,2))
  for k in range(K):
    for n in range(N):
      mu[k,:]+=gamma[n,k]*x[n,:]
    mu[k,:]/=num[k]

  sig=np.zeros((K,2,2))
  for k in range(K):
    for n in range(N):
```

```
      sig[k,:]+=gamma[n,k]*outer(x[n,:]-mu[k,:],x[n,:]-mu[k,:])
    sig[k,:]/=num[k]
  Pi[:]=num[:]/N
  #対数尤度計算
  ln_p=0
  for n in range(N):
    tmp=0
    for k in range(K):
      tmp+=Pi[k]*dist[n,k]
    ln_p+=log(tmp)

  diff-=ln_p
  print(abs(diff))

C=[[],[],[]]
for n in range(N):
  for k in range(K):
    if k==np.argmax(dist[n,:]):
      C[k].append(n)
```

4 結果

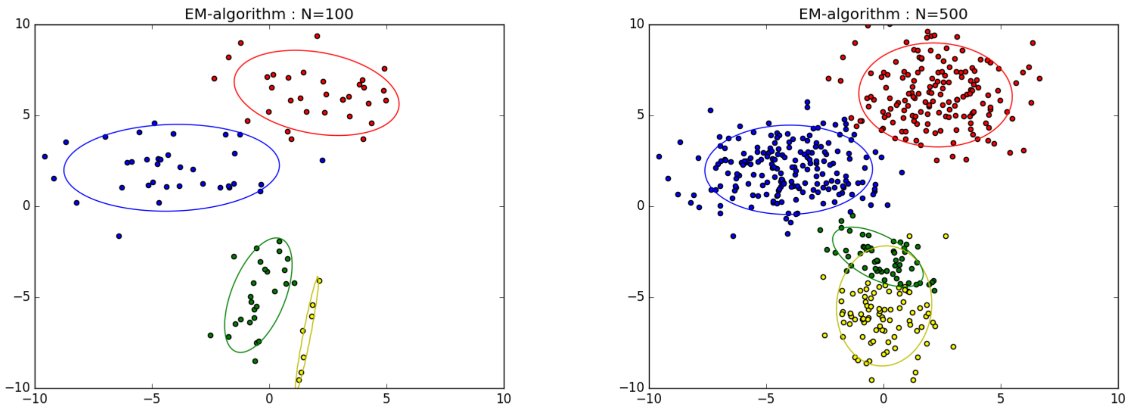
3 クラス分類を行った.

図 1: $N = 50, 100, 200, 500$



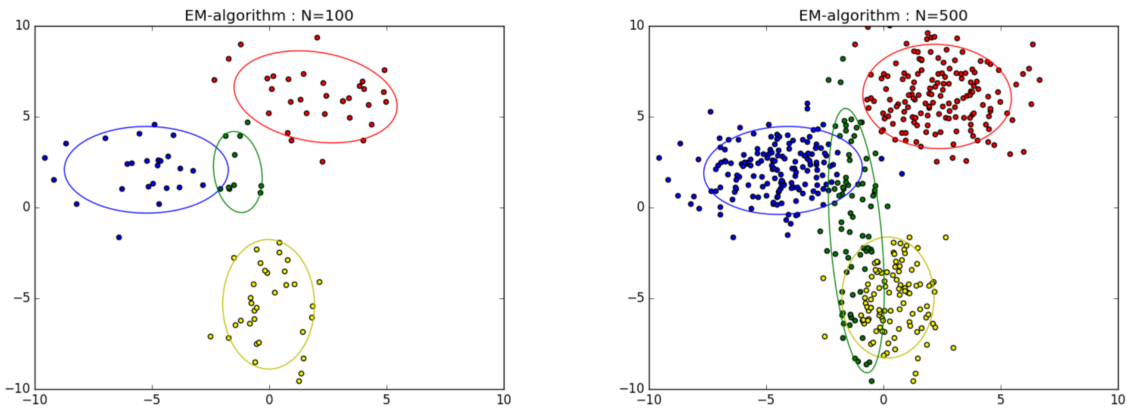
4 クラス分類を行った.(初期プロトタイプ $(3,3), (3,-3), (-3,3), (-3,-3)$)

図 2: $N = 100, 500$



初期プロトタイプを変えてみた.(初期プロトタイプ $(3,3),(3,-3),(0,-3),(0,0)$)

図 3: $N = 100, 500$



5　まとめ

K-means 法の後でも, EM-アルゴリズムが収束するのには結構時間がかかった.
また, クラスター数が正しくないとき初期プロトタイプにも大きく依存する.
つまり, クラスター数が分かっているときはうまくいったが, クラスター数が分からないときには, まずクラスター数を知ることが重要となる.