

レポート課題 2: 混合ガウスモデルの最尤推定

提出締切 1月25日(金) 18:00. 提出先: A-333

目的: ある未知の確率分布 $p(x)$ にしたがって生成されたデータ x_1, x_2, \dots, x_n が観測されている. m 個のガウス関数の重み付きの和で, できるだけ精度よく, $p(x)$ を近似してみよう. for 文を使わない octave のプログラミングも体験しよう.

基礎知識 (教科書第8章, pp.105-118)

$x_i \sim p(x)$ とする. x は画像など多次元ベクトルでもよいが, ここでは簡単のため1次元の場合を考える. $p(x)$ を複数のガウス関数 $\phi(x; \mu, \sigma^2)$ の重み付きの和

$$q(x; \theta) = \sum_{l=1}^m w_l \phi(x; \mu_l, \sigma_l^2) \quad (1)$$

で近似して表現することを試みる. これを混合ガウスモデルという. もちろん

$$\phi(x; \mu_l, \sigma_l^2) = \frac{1}{\sqrt{2\pi}\sigma_l} e^{-\frac{(x-\mu_l)^2}{2\sigma_l^2}}$$

である. データ x_1, \dots, x_n からもとの確率分布を推定するにはパラメータ

$$\theta = (\mu_1, \dots, \mu_m, \sigma_1, \dots, \sigma_m, w_1, \dots, w_m), \quad \sum_{l=1}^m w_l = 1$$

について, 尤度を最大にする最尤推定値

$$\theta^* = \operatorname{argmax}_{\theta} \prod_{i=1}^n q(x_i; \theta) = \operatorname{argmax}_{\theta} \sum_{i=1}^n \log q(x_i; \theta) \quad (2)$$

を求めればよい. この θ^* を求める次の手順が知られている (詳細は教科書を参照).

EM アルゴリズム

1. w_l, μ_l, σ_l の値として, 適当な初期値を設定 ($l = 1, \dots, m, \sum_{l=1}^m w_l = 1, w_l \geq 0$).

2. $\eta_{i,l} := \frac{w_l \phi(x_i; \mu_l, \sigma_l^2)}{\sum_{l'=1}^m w_{l'} \phi(x_i; \mu_{l'}, \sigma_{l'}^2)}$ を計算 ($i = 1, \dots, n, l = 1, \dots, m$).

3. $w_l := \frac{1}{n} \sum_{i=1}^n \eta_{i,l}$, $\mu_l := \frac{\sum_{i=1}^n \eta_{i,l} x_i}{\sum_{i=1}^n \eta_{i,l}}$, $\sigma_l := \sqrt{\frac{\sum_{i=1}^n \eta_{i,l} (x_i - \mu_l)^2}{\sum_{i=1}^n \eta_{i,l}}}$ を計算.

4. 2~3 を尤度が収束するまで繰り返す.

Python でも、for 文は使えるが計算スピードは遅くなる。できる限り for 文を使わないほうがよい (octave 版は教科書 p.34)。では、どう書けばよいか。

$$\eta_{i,l} := \frac{w_l \phi(x_i; \mu_l, \sigma_l^2)}{\sum_{l'=1}^m w_{l'} \phi(x_i; \mu_{l'}, \sigma_{l'}^2)}$$

を計算する例を示そう。ここで、 $i = 1, \dots, n$, $l = 1, \dots, m$ であるので、C 言語なら 2 次元配列 $\eta[i][l]$ を用意し、2 重 for ループで値を順に代入するだろう。一方、Python では、変数のコピーをたくさん用意することで for 文を使わず同じことが実現できる (教科書 p.118)。

```
tmp1 = np.square( np.tile(x, (m,1)) - np.tile(mu, (1, n)) )
tmp2 = 2*np.tile( sigma2, (1, n) )
tmp3 = np.tile(w, (1, n) ) * np.exp(-tmp1/tmp2) / np.sqrt(np.pi*tmp2)
eta = tmp3 / np.tile(np.sum(tmp3, axis=0), (m, 1) )
```

はじめてこれを見た場合、なんのことも分からない。そういう場合は、小さい具体例で考えればよい。 $i = 1, \dots, n$, $l = 1, \dots, m$ の 2 重ループを実現するため、 n 個のデータをまとめた x (n 次元横ベクトル) を縦方向に m 個、各ガウスモデルの平均値をまとめた μ (m 次元縦ベクトル) を横方向に n 個複製する。このようにしておくと

$$\phi(x_i; \mu_l, \sigma_l^2) = \frac{1}{\sqrt{2\pi\sigma_l}} e^{-\frac{(x_i - \mu_l)^2}{2\sigma_l^2}}$$

の指数関数の肩にある分数の分子 $(x_i - \mu_l)^2$ が、 $m \times n$ の各要素ごとに計算できる ($i = 1, \dots, n$, $l = 1, \dots, m$)。これが tmp1 で、計算結果は $m \times n$ 行列である。次に、tmp2 は指数関数の肩にある分数の分母 $2\sigma_l^2$ の計算である ($m \times n$ 行列)。あとで tmp1/tmp2 をしたいので、tmp1 と tmp2 の型は同一でなければいけない。ここで * とか / は、行列の要素ごとの掛け算、割り算を意味している。あらかじめ指定しておく必要がある sigma2 は σ^2 の値、つまり標準偏差ではなく分散の値であることを確認しておこう。

tmp3 中に、 $\text{np.sqrt}(\text{np.pi} * \text{tmp2})$ がある。この型は tmp2 と同じで $m \times n$ である。したがって $\text{np.exp}(-\text{tmp1}/\text{tmp2}) / \text{np.sqrt}(\text{pi} * \text{tmp2})$ は $e^{-\frac{(x_i - \mu_l)^2}{2\sigma_l^2}}$ である ($m \times n$ 行列)。tmp3 の最初の項 $\text{np.tile}(w, (1, n))$ は、 w という m 次元縦ベクトルを横に n 個複製して作った $m \times n$ 行列である。これで tmp3 が $w_l \phi(x_i; \mu_l, \sigma_l^2)$ を意味していることが分かった。

$\text{eta} = \text{tmp3} / \text{np.tile}(\text{np.sum}(\text{tmp3}, \text{axis}=0), (m, 1))$ を考えよう。 $\text{np.sum}(\text{tmp3}, \text{axis}=0)$ は tmp3 の l について全部足している。 $\text{axis}=0$ というのは縦方向に要素を足すという意味である。足し算した結果は 1 つであるが、計算した後、結果を縦に m 個複製している。

次に,

$$w_l := \frac{1}{n} \sum_{i=1}^n \eta_{i,l}, \quad \mu_l := \frac{\sum_{i=1}^n \eta_{i,l} x_i}{\sum_{i'=1}^n \eta_{i',l}}, \quad \sigma_l^2 := \frac{\sum_{i=1}^n \eta_{i,l} (x_i - \mu_l)^2}{\sum_{i'=1}^n \eta_{i',l}}$$

をどのように計算しているか確認しておこう。

```
tmp4 = np.sum(eta, axis=1)
w = tmp4/n
w = w.reshape(m,1)
mu = (eta.dot(x))/tmp4
mu = mu.reshape(m,1)
sigma2 = np.sum(tmp1*eta, axis=1)/tmp4
sigma2 = sigma2.reshape(m,1)
```

`np.sum(eta, axis=1)` の 1 は、`eta` の型が $m \times n$ であるので、横方向に (n 個を) 足し込むという意味があり、 $\sum_{i=1}^n$ に対応する。結果、`w` の型は $m \times 1$ である。`tmp1`, `tmp4` は効率的に再利用でき、`mu`, `sigma2` が計算できる。計算結果の型はすべて $m \times 1$ (m 次元縦ベクトル) である。

ここまですべてをまとめておこう。くどいが $i = 1, \dots, n$, $l = 1, \dots, m$ である。

tmp1	$(x_i - \mu_l)^2$	$m \times n$
tmp2	$2\sigma_l^2$	$m \times n$
tmp3	$w_l \phi(x_i; \mu_l, \sigma_l^2)$	$m \times n$
eta	$\frac{w_l \phi(x_i; \mu_l, \sigma_l^2)}{\sum_{l'=1}^m w_{l'} \phi(x_i; \mu_{l'}, \sigma_{l'}^2)}$	$m \times n$
tmp4	$\sum_{i=1}^n \frac{w_l \phi(x_i; \mu_l, \sigma_l^2)}{\sum_{l'=1}^m w_{l'} \phi(x_i; \mu_{l'}, \sigma_{l'}^2)}$	$m \times 1$

最後に

$$L(\theta) = \sum_{i=1}^n \log \sum_{l=1}^m w_l \phi(x_i; \mu_l, \sigma_l^2) \quad (3)$$

の計算を考えよう。この計算には `tmp3` が再利用できる。まず、 $\sum_{l=1}^m w_l \phi(x_i; \mu_l, \sigma_l^2)$ は `np.sum(tmp3, axis=0)` で計算できる ($1 \times n$ の横ベクトル)。この各要素を `log` をとり、 n 項を足し算すればよいのだから `np.sum(np.log(np.sum(tmp3, axis=0)))` で計算できる。横方向に足しこむ意味の `axis=1` は省略できる。

```
Lnew = np.sum(np.log(np.sum(tmp3, axis=0)))
```

コンピュータ演習： $m = 3$ の混合ガウスモデルから、データを n 個生成する関数を作成せよ。モデルのパラメータの値を適当に指定し、具体的にデータを 1000 個生成し、ヒストグラムを描け。また、4つの関数、 $q(x; \theta)$, $\phi(x; \mu_l, \sigma_l^2)$ を同一グラフに描いてみよ ($l = 1, \dots, 3$)。

$w_1 = w_2 = w_3 = 1/3, \mu_1 = 1, \mu_2 = 2, \mu_3 = 3, \sigma_1 = 0.1, \sigma_2 = 0.3, \sigma_3 = 0.5$ とする。

$$q(x; \theta) = \sum_{l=1}^m w_l \phi(x; \mu_l, \sigma_l^2)$$

からデータを 1000 個生成する例を以下に示す。

ガウス混合モデルからのデータ生成： test_gmm.py

```
# -*- coding: utf-8 -*-
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

def myrand_gmm(n, fill=0.0): # n は生成するデータの個数
    x=np.zeros(n)
    g=np.random.randn(n)
    u=np.random.rand(n)
    mu = np.array([1.0, 2.0, 3.0]) # 各ガウス分布の平均値. 値を変えていろいろ試す. これは混合
    数 m=3 の場合
    sigma = np.array([0.1, 0.3, 0.5]) # 各分布の標準偏差. 値を変えていろいろ試す.
    flag=(0<=u) & (u<1/3) # この例は, 各分布から 1/3 の確率でデータが出現する場合.
    x = (mu[0] + sigma[0]*g)*flag
    flag=(1/3<=u) & (u<2/3)
    x += (mu[1] + sigma[1]*g)*flag
    flag=(2/3<=u) & (u<=1)
    x += (mu[2] + sigma[2]*g)*flag
    return x

n = 1000 # 標本数 (サンプル数).
x = myrand_gmm(n) # 乱数の生成. 実験に使うデータを生成する関数名に書き換える.
m = 3 # 混合数. この値を変えて実験する.

mu = np.array([1.0, 2.0, 3.0])
sigma = np.array([0.1, 0.3, 0.5])
sigma2 = sigma*sigma
w = np.ones(m)/m

xx = np.arange(0,5,0.01) #0 から 5 まで, 0.01 間隔.
y0 = norm.pdf(xx, mu[0], np.sqrt(sigma2[0] ) )
y1 = norm.pdf(xx, mu[1], np.sqrt(sigma2[1] ) )
y2 = norm.pdf(xx, mu[2], np.sqrt(sigma2[2] ) )
y = w[0]*y0 + w[1]*y1 + w[2]* y2

plt.plot(xx,y,color='r')
plt.hist(x, bins='auto', normed=True)
plt.show()
```

% python test_gmm.py

で実行できる。もし理解できないところがあれば、ipython を立ち上げた後、1行1行実行し、whos で各変数の形を確認しながら進むとよい。

p118a.py

```
# -*- coding: utf-8 -*-
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

# def myrand_gmm(n, fill=0.0): # n は生成するデータの個数
# 関数を定義しておく. 前ページ参照.

n = 5000 # 標本数 (サンプル数).
x = myrand_gmm(n) # 乱数の生成. 実験に使うデータを生成する関数名に書き換える.
m = 3 # 混合数. この値を変えて実験する.

# 初期値の設定. w, mu, sigma2 は m 次元縦ベクトル.
L = -np.inf
w = np.ones(m)/m # m 個の正規分布の重みの初期値
w = w.reshape(m,1)
mu = np.linspace( min(x), max(x), m) # 平均値の初期値
mu = mu.reshape(m,1) # m を明示的に縦ベクトルにする
sigma2 = np.ones(m)/10 # 分散の初期値
sigma2 = sigma2.reshape(m,1)

while 1:
    tmp1 = np.square( np.tile(x, (m,1)) - np.tile(mu, (1, n)) )
    tmp2 = 2*np.tile( sigma2, (1, n) )
    tmp3 = np.tile(w, (1, n) ) * np.exp(-tmp1/tmp2) / np.sqrt(np.pi*tmp2)
    eta = tmp3 / np.tile(np.sum(tmp3, axis=0), (m, 1) ) # ここまでが  $\eta$  の計算
    tmp4 = np.sum(eta, axis=1)
    w = tmp4/n
    w = w.reshape(m,1)
    mu = (eta.dot(x))/tmp4
    mu = mu.reshape(m,1)
    sigma2 = np.sum(tmp1*eta, axis=1)/tmp4
    sigma2 = sigma2.reshape(m,1)
    Lnew = np.sum(np.log(np.sum(tmp3,axis=0))) # 更新後の対数尤度

    if Lnew - L < 0.0001:
        break
    L = Lnew

xx = np.arange(0,5,0.01) # 0 から 5 まで、0.01 間隔.
y0 = norm.pdf(xx, mu[0], np.sqrt(sigma2[0] ) )
y1 = norm.pdf(xx, mu[1], np.sqrt(sigma2[1] ) )
y2 = norm.pdf(xx, mu[2], np.sqrt(sigma2[2] ) )
y = w[0]*y0 + w[1]*y1 + w[2]* y2

plt.plot(xx,y,color='r')
plt.hist(x, bins='auto', normed=True)
plt.show()
```

レポート課題：

1. $m = 2$, $w_1 = w_2 = 1/2$, $\mu_1 = 1$, $\mu_2 = 2$, $\sigma_1 = 0.1$, $\sigma_2 = 0.3$ とする.

$$q(x; \theta) = \sum_{l=1}^m w_l \phi(x; \mu_l, \sigma_l^2)$$

からデータを1000個生成し、ヒストグラムを描け。3つの確率密度関数, $q(x; \theta)$, $\phi(x; \mu_l, \sigma_l^2)$ も同一のグラフに描きなさい ($l = 1, 2$) .

2. $m = 3$ で μ, σ を適当な値に設定し、確率密度関数 $p(x)$ からデータを生成する関数を作れ. $m = 3$ のモデル $q(x)$ を使い、最尤推定せよ (p118a.py 参照). EM アルゴリズムの繰り返し回数が増えるにしたがい、各パラメータの値が、もとの確率分布のパラメータに近づいていく様子を示せ.

以下の問を 2. と同様に実験し、分析・考察せよ.

3. 真の分布 $p(x)$ を $m = 5$ で作成し、 $m = 3$ のモデル $q(x)$ を使い最尤推定せよ.
4. 真の分布 $p(x)$ を $m = 3$ で作成し、 $m = 5$ のモデル $q(x)$ を使い最尤推定せよ.
5. 真の分布が、教科書 p.172 の `myrand(n)` で定義される分布とする. $m = 2, 3, 5$ のモデル $q(x)$ を使い最尤推定せよ.
6. (オプション) 適当な問題を作り、結果を分析せよ.

レポートの最後には、感想を記述してほしい. 理解できた点, 理解できない点・疑問点などを、具体的に箇条書きしてほしい. このプリント中に理解しにくい点があった場合は、何ページ何行目の、どこの部分が分かりにくかったか、具体的に、指摘してほしい.

注意事項：

1. この文章中には記述ミスがあるかもしれない. その場合は以下のページで更新情報を伝えます. 気づいた点があれば、気軽にメールで連絡してほしい.

http://www.cs.miyazaki-u.ac.jp/~date/lectures/pattern/octave4pattern_Ch8.html

2. レポートの L^AT_EX を使った簡単な書き方は

<http://www.cs.miyazaki-u.ac.jp/~date/lectures/latex/latexreport.html> を参照.

3. レポートは、1年前の自分が読んでも、何を調べようとしているのか (目的), 得られた結果 (図) が分かるように書いていけばよい (コレは簡単ではない).
4. 独力で課題が遂行できそうにない場合は、早めに相談すること.