6.4.5〜6.46 ガウス過程による分類

平成 28 年 9 月 11 日

概 要

PRML の「6.4.5 ガウス過程による分類」「6.4.6 ラプラス近似」についての実装と考察.

目 次

1　問題設定

t_n がガウス分布に従っているものをガウス過程という. このガウス過程について考察する.
ここでは, ガウス過程による分類を行う.

2　アルゴリズム

目標変数が $t \in 0, 1$ であるような 2 クラス分類問題を考える.
目標変数 t の確率分布は, 関数 $a(x)$ 上でのガウス過程を定義し, これをロジスティックシグモイド関数を用いて

$$p(t|a) = \sigma(a)^t(1 - \sigma(a))^{1-t} \quad (6.73)$$

とあらわせる.
ここでの目標は $p(t_{N+1}|\boldsymbol{t})$ を求めることである. そのため, まずベクトル $\boldsymbol{a}_{N+1}$ に対するガウス過程による事前分布を考える.

$$p(\mathbf{a}_{N+1}) = N(\mathbf{a}_{N+1}|\mathbf{0}, C_{N+1}) \quad (6.74)$$

これを用い $p(\mathbf{t})$ が与えられる. また,　共分散行列は

$$C(x_n, x_m) = k(x_n, x_m) + \nu\delta_{nm} \quad (6.75)$$

が用いられることが多い (ν は数値的安定性の観点からノイズ項として入れられる). そして, 求めたい予測分布は

$$p(t_{N+1} = 1|\mathbf{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)da_{N+1} \quad (6.76)$$

となるが, この積分は解析的に求めることができず, サンプリングや解析的な近似を用いる.

3　ラプラス近似

(6.76) の予測分布を求める解析的な近似法の一つとして, ラプラス近似がある. a_{N+1} の事後分布のガウス分布による近似を考えるのである.
ベイズの定理と $p(\mathbf{t}_N|a_{N+1}, \mathbf{a}_N) = p(\mathbf{t}_N|\mathbf{a}_N)$ から

$$p(\mathbf{a}_N|\mathbf{t}_N) = \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)d\mathbf{a}_N \quad (6.77)$$

となり, またガウス過程による回帰で求めた (6.66),(6.67) から

$$p(a_{N+1}|\mathbf{a}_N) = N(a_{N+1}|\mathbf{k}^T C_N^{-1}\mathbf{a}_N, c - \mathbf{k}^T C_N^{-1}\mathbf{k}) \quad (6.78)$$

となるため, (6.77) の積分はこれと事後分布 $p(\mathbf{a}_N|\mathbf{t}_N)$ のラプラス近似との (2 つのガウス分布) たたみ込みとなる.
(6.73) より

$$p(\mathbf{t}_N|\mathbf{a}_N) = \prod_{n=1}^{N} \sigma(a_n)^{t_n}(\sigma(a_n))^{1-t_n} = \prod_{n=1}^{N} e^{a_n t_n}\sigma(-a_n) \quad (6.79)$$

となる.
またベイズの定理

$$p(\mathbf{a}_N|\mathbf{t}_N) = \frac{p(\mathbf{t}_N|\mathbf{a}_N)p(\mathbf{a}_N)}{p(\mathbf{t}_N)}$$

の対数は定数部分を除き

$$\begin{aligned}\Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N|\mathbf{a}_N) \\ &= -\frac{1}{2}\mathbf{a}_N^T C_N^{-1}\mathbf{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|C_N| + \mathbf{t}_N^T\mathbf{a}_N - \sum_{n-1}^{N}\ln(1+e^{a_n}) \quad (6.80)\end{aligned}$$

この勾配は

$$\nabla\Psi(\mathbf{a}_N) = \boldsymbol{t}_N - \boldsymbol{\sigma}_N - C_N^{-1}\mathbf{a}_N \quad (6.81)$$

となるため, ラプラス近似で用いる平均は

$$\mathbf{a}_N^* = C_N(\mathbf{t}_N - \boldsymbol{\sigma}_N) \quad (6.84)$$

ただし, $\boldsymbol{\sigma}_N$ は要素に $\sigma(a_n)$ を持つため, これはニュートン-ラフソン法など逐次更新式を用いる.

$$\begin{aligned}\mathbf{a}_N^{new} &= \mathbf{a}_N^{old} - H^{-1}\nabla\Psi(\mathbf{a}_N^{old}) \\ &= C_N(I + W_N C_N)^{-1}\{\mathbf{t}_N - \boldsymbol{\sigma}_N + W_N\mathbf{a}_N\}\end{aligned}$$

ここで, ヘッセ行列は

$$H = -\nabla\nabla\Psi(\mathbf{a}_N) = W_N + C_N^{-1} \quad (6.85)$$

であることを用いている. また W_N は $\sigma(a_n)(1-\sigma(a_n))$ 要素に持つ対角行列である.
そして, 肝心のラプラス近似は

$$q(\mathbf{a}_N) = N(\mathbf{a}_N|\mathbf{a}_N^*, H^{-1}) \quad (6.86)$$

である.
よって, (6.77) の積分は線形ガウスモデルに対応しており

$$E[a_{N+1}|\mathbf{t}_N] = \mathbf{k}^T(\mathbf{t}_N - \boldsymbol{\sigma}_N) \quad (6.87)$$

$$var[a_{N+1}|\mathbf{t}_N] = c - \mathbf{k}^T(W_N^{-1} + C_N)^{-1}\mathbf{k} \quad (6.88)$$

となる. この結果とプロビット関数の逆関数による近似 (4.153) を用いることにより, $p(t_{N+1}=1|\mathbf{t}_N)$ の積分を求めることができ,

$$p(t_{N+1}=1|\mathbf{t}) = \sigma(\kappa(\sigma^2)\mu)$$

ここで, μ には (6.87),σ^2 には (6.88) の結果を用い $\kappa(\cdot)$ は

$$\kappa(\sigma^2) = (1+\pi\sigma^2/8)^{-1/2} \quad (4.154)$$

である.

4 コード

プロットの部分は除いた (bunnrui.py).

```
"""カーネル関数の定義"""
theta=1
def gauss(x,z):
  res=0
  for i in range(2):
    res+=np.exp(-(x[i]-z[i])**2/2*theta)
  return res
```

```
def sigma(z):
  return 1/(1+np.exp(-z))

def kappa(z):
  return 1/sqrt(1+pi*z/8)

"""Wの最適化"""
for N in [10,30,50,100]:
  x=data[:N,0:2]
  t=data[:N,2]
  nu=0
  C=np.identity(N)*nu
  for n in range(N):
    for m in range(N):
      C[n,m]+=gauss(x[n,:],x[m,:])

  frag=0
  a=np.random.rand(N)
  sig=np.zeros(N)
  I=np.identity(N)
  W=np.zeros((N,N))
  while frag==0:
    b=a
    sig=sigma(a)
    for n in range(N):
      W[n,n]=sig[n]*(1-sig[n])
    a=dot(C,dot(inv(I+dot(W,C)),(t-sig+dot(W,a))))
    if norm(b-a)<N*10**-8:
      frag=1
  H=W+inv(C)

  #求まったパラメータからモデル関数を作り

  def E(z):
    k=np.zeros(N)
    for n in range(N):
      k[n]=gauss(x[n],z)
    return dot(k,t-sig)

  def var(z):
    c=gauss(z,z)+nu
    k=np.zeros(N)
    for n in range(N):
      k[n]=gauss(x[n,:],z)
    return c-dot(k,dot(inv(inv(W)+C),k))

  def model(z):
    return sigma(kappa(var(z))*E(z))
```

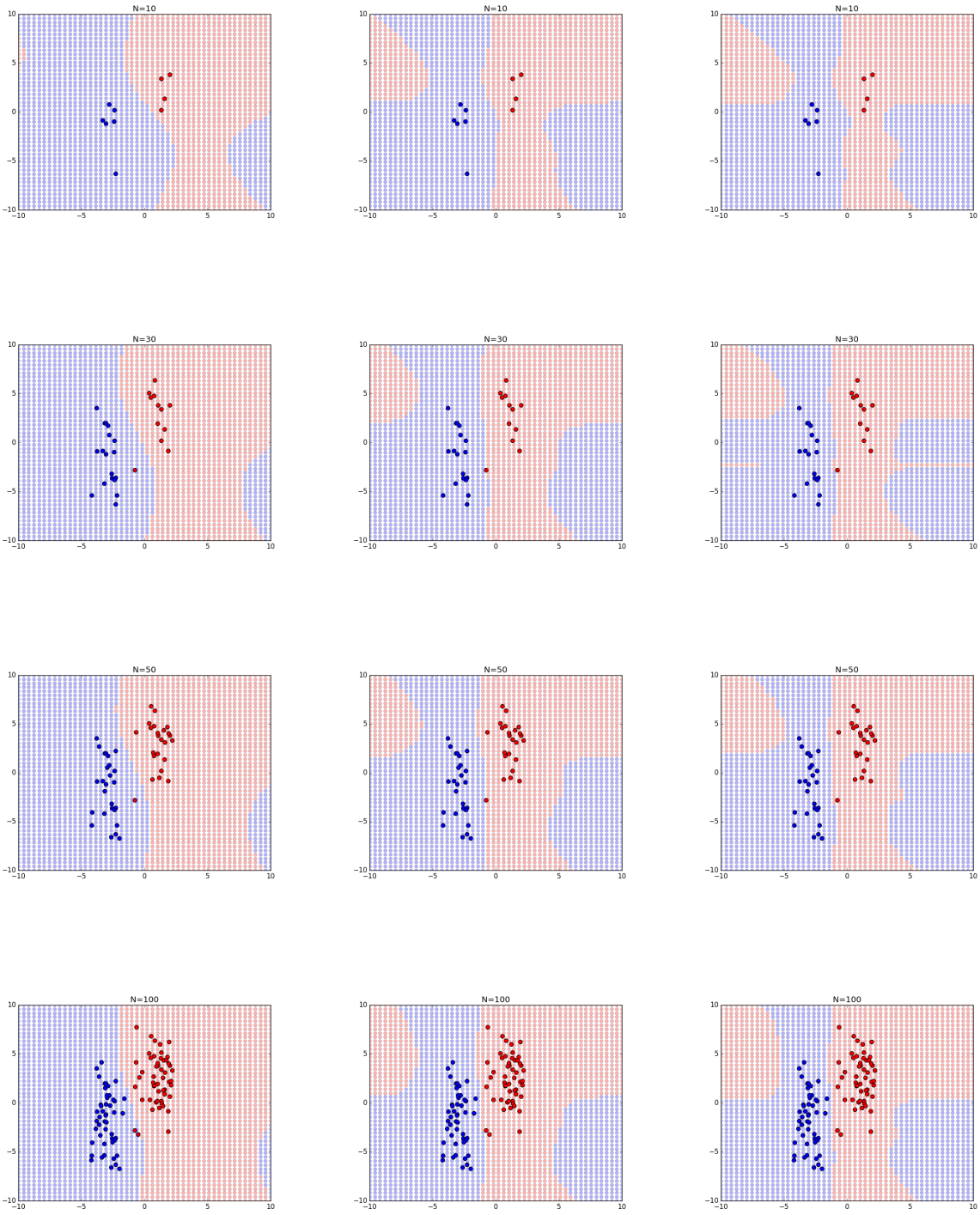
5 結果

5.1 ガウスカーネル

カーネル関数に

$$k(x_n, x_m) = exp\{-\frac{\theta}{2}(x_n - x_m)^2\}$$

を用いて, $\theta = 0.05, 0.3, 1$ としてデータ数 N$= 10, 30, 50, 100$ にそれぞれ試してみた



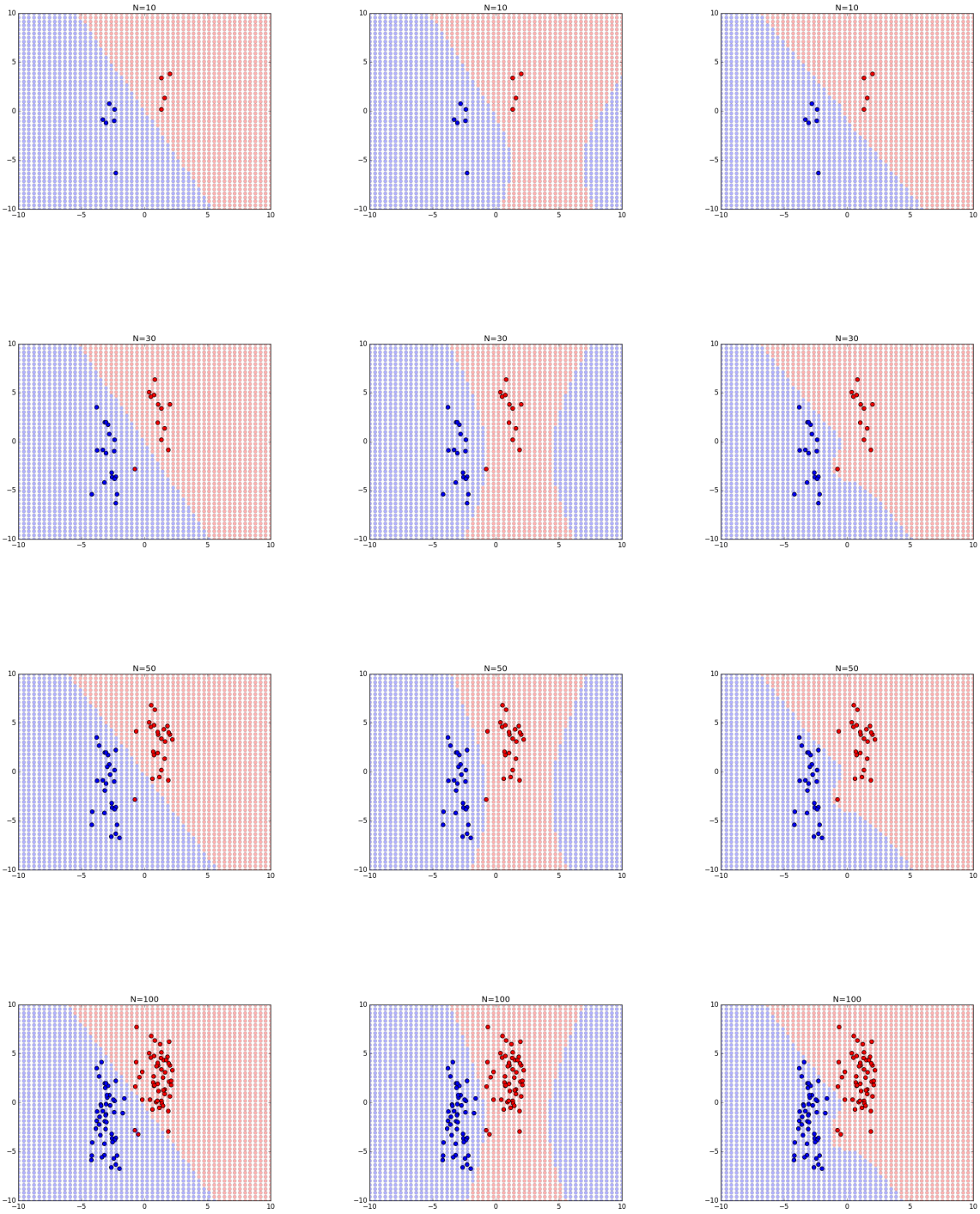
分類には成功しているが, 飛び地が発生している. 人ならこのような分類は行わないであろう. 原因としては, カーネル関数が問題に不適合, パラメータが最適化されていない, そもそもデータが近くにない点では予測が成立しないなどが考えられる.

5.2 多項式カーネル

カーネル関数に

$$k(x_n, x_m) = (\boldsymbol{x}^T \boldsymbol{z} + 1)^{\theta_0}/\theta_1$$

を用いて, $\theta_0 = 1, 2, 3, \theta_1 = N$ ($\theta_0 = 3$ のとき), 1 (その他) としてデータ数 N= $10, 30, 50, 100$ にそれぞれ試してみた. ($\theta_1 = N$ を用いたのは C の逆行列が計算できなくなるエラーを除くため, $\theta_0 = 2$ のとき $\theta_1 = N$ とすると悪影響を及ぼした)



この場合も, カーネル関数に大きく依存した形になっている.

6　超パラメータの学習

カーネル関数に含まれるパラメータ θ について最適化する. 尤度 $p(\boldsymbol{t}|\theta)$ を最大化する.

$$p(\mathbf{t}|\theta) = \int p(\mathbf{t}_N|\mathbf{a}_N)p(\mathbf{a}_N|\theta)d\mathbf{a}_N \quad (6.89)$$

ラプラス近似より

$$\ln p(\mathbf{t}_N|\theta) = \Psi(\mathbf{a}_N^*) - \frac{1}{2}\ln|W_N + C_N^{-1}| + \frac{N}{2}\ln(2\pi) \quad (6.90)$$

これを θ で微分したものを 0 とする.

7　まとめ

課題としてはカーネル関数の選択に尽きる. データに対して最適なカーネル関数を選択できる基準があればいいと思った. また, 分類自体はうまくいくものの, 決定境界はきっちり定まっていない印象が大きい.