3.3 ベイズ線形回帰

平成 28 年 9 月 14 日

概 要

PRML の「3.3.1 パラメータの分布」「3.3.2 予測分布」についての実装と考察

目 次

1　3.3.1 パラメータの分布

1.1　問題設定

線形回帰モデルのベイズ的取扱いについて考える.
ただし, 超パラメータ α, β については考えない.

1.2　アルゴリズム

尤度関数は (β を既知として),

$$p(\mathbf{t}|X, W, \beta) = \prod_{n=1}^{N} N(t_n|\mathbf{w}^T \boldsymbol{\phi}(x_n), \beta^{-1}) \quad (3.10)$$

となり, モデルパラメータ W の事前分布として,

$$p(\mathbf{w}) = N(\mathbf{w}|\mathbf{m}_0, S_0) \quad (3.48)$$

を用いる. これは, 尤度関数の共役事前分布からきている.
そして, ベイズの定理から 事後分布 $\propto$ 尤度 $\times$ 事前分布 となるので,

$$p(\mathbf{w}|\mathbf{t}) = N(\mathbf{w}|\mathbf{m}_N, S_N) \quad (3.49)$$

ただし,

$$S_N^{-1} = S_0^{-1} + \beta \Phi^T \Phi \quad (3.50), \quad \mathbf{m}_N = S_N(S_0^{-1}\mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \quad (3.51)$$

そして, 事前分布として,

$$p(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \alpha^{-1} I) \quad (3.52)$$

を選ぶとすると,

$$S_N^{-1} = \alpha I + \beta \Phi^T \Phi \quad (3.53), \quad \mathbf{m}_N = \beta S_N \Phi^T \mathbf{t} \quad (3.54)$$

となる.

1.3　コード

事後分布のパラメータを求める (test.py)

```
for N in [20,100,500]:
        x=data[:N,0]
        t=data[:N,1]
        for M in [4,10,20]:
                """Wの最適化"""
                A=np.zeros((M,M))
                W=np.zeros(M)
                P=np.zeros((N,M))
                I=np.identity(M)

                mu=[(m+0.5)*(2*pi/(M+1)) for m in range(M)]
                s=(2*pi)**2/12


                for n in range(N):
                        for m in range(M):
                                P[n,m]=tanh_basis(x[n],m,mu[m],s)
```

```
alpha=10**(-10)
beta=1.0/(0.3)**2


S_N=inv(alpha*I+beta*np.dot(P.T,P))
m_N=beta*np.dot(S_N,np.dot(P.T,t))

#求まったパラメータからモデル関数を作り
def model_f(x):
        sum=0
        for m in range(M):
                sum+=m_N[m]*tanh_basis(x,m,mu[m],s)
        return sum

"""表示"""

#パラメータ Wの出力
#print("m_N=",m_N)
#print("S_N=",S_N)

#誤差関数を定義し出力
def Error():
        sum=0
        for n in range(50):
                sum+=(model_f(new_x[n])-new_t[n])**2
        return np.sqrt(sum/50)

print(N,"\t",M,"\t%.2f" % Error())
```

1.4 結果

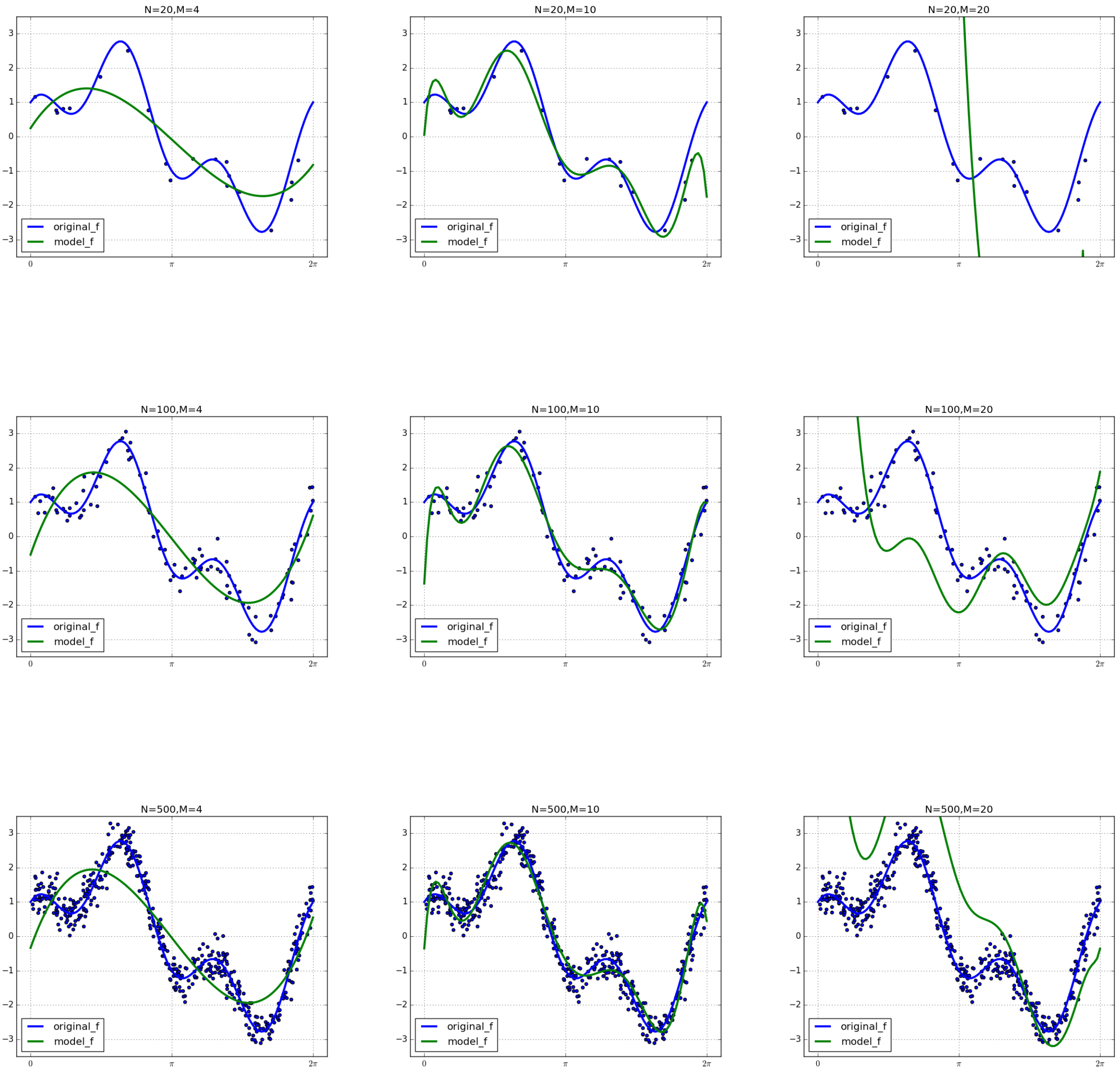
$\alpha = 10^{-10}$, $\beta = 1.0/(0.3)^2$ に固定して実験を行った.
この β は元のデータの分散が $(0.3)^2$ であることに由来する. α については適当.

1.4.1 多項式基底

基底 $\Phi(x)$ に $\phi_i(x) = x^i$ を選んだ. 「多項式曲線フィッティング」で行ったものと等しい.

M \ N	20	100	500
4	0.86	0.79	0.77
10	0.55	0.47	0.39
20	320	5.92	3.41

表 1: E_{RMS} の N,M との関係 (多項式基底)



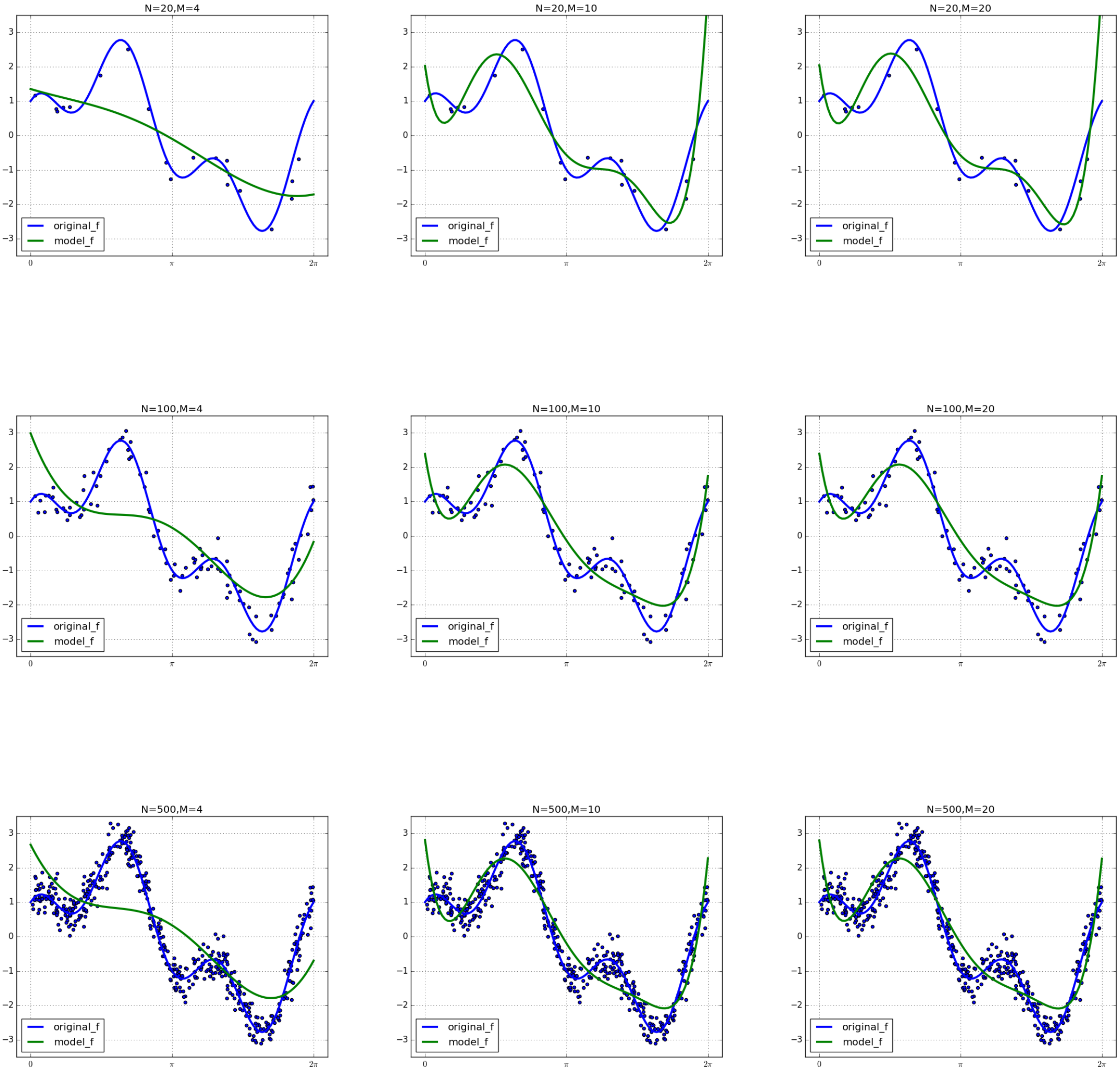
1.4.2　ガウス基底

基底 $\Phi(x)$ に $\phi_i(x) = exp\{-\frac{(x-\mu_i)^2}{2s^2}\}$ をもちいる．またここでは μ は区間 $[0, 2\pi]$ を M 個に等分する区間の中心を用い,s には x の分散を用いた．

$$mu = [(m+0.5)*(2*pi/(M+1))\ for\ m\ in\ range(M)] \quad s = (2*pi)/M$$

M＼N	20	100	500
4	1.03	0.98	0.97
10	0.75	0.62	0.63
20	0.77	0.62	0.63

表 2: E_{RMS} の N,M との関係 (ガウス基底)

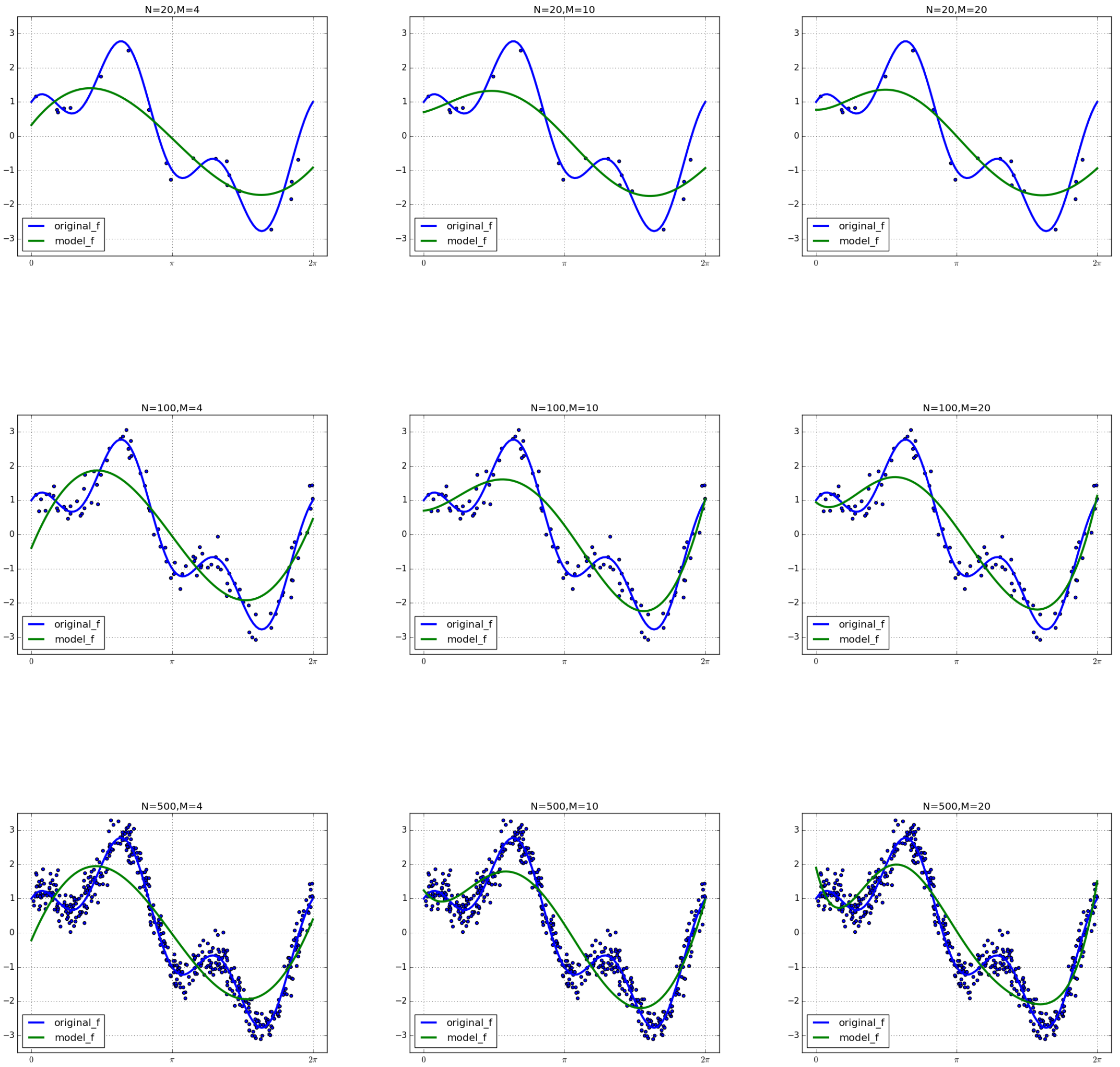


1.4.3 シグモイド基底

基底 $\Phi(x)$ に $\phi_i(x) = \sigma(\frac{x-\mu_i}{s})$ をもちいる. ただし,$\sigma(a) = \frac{1}{1+e^{-a}}$

M \ N	20	100	500
4	0.85	0.79	0.77
10	0.83	0.71	0.70
20	0.82	0.68	0.64

表 3: E_{RMS} の N,M との関係 (シグモイド基底)

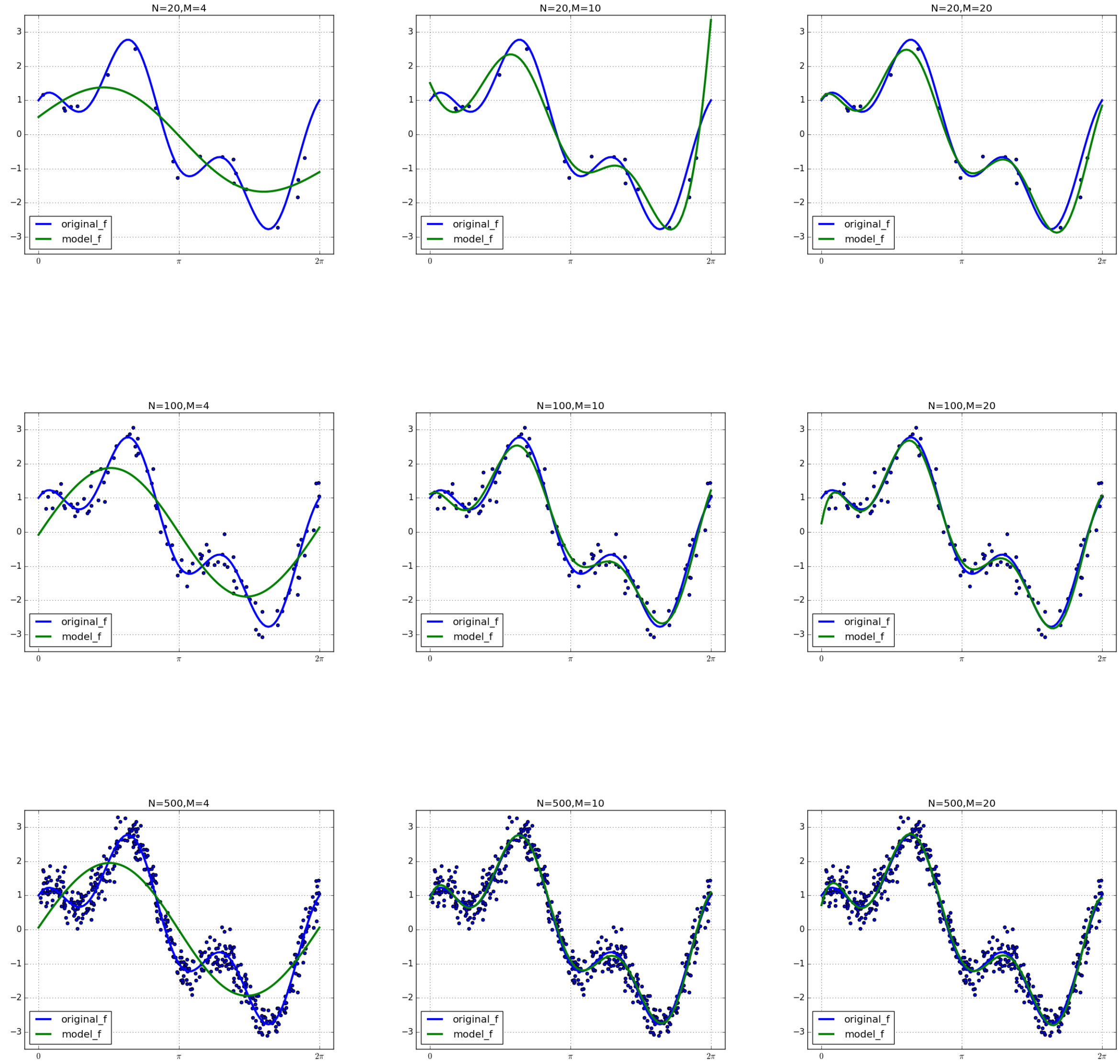


1.4.4 tanh 基底

基底 $\Phi(x)$ に $\phi_i(x) = tanh(\frac{x-\mu_i}{s})$ をもちいる.

M \ N	20	100	500
4	0.85	0.79	0.78
10	0.56	0.31	0.27
20	0.36	0.31	0.28

表 4: E_{RMS} の N,M との関係 (tanh 基底)



1.5 まとめ

多項式基底ではあまりうまくいかなかった (特に M が大きいとき). 考えるに, 事前分布で, 各要素を一律に扱っていることが原因ではないだろうか.
元の関数が三角関数の組み合わせだけあって, tanh 基底が一番うまくいった.
gauss,sigmoid,tanh 基底などでは μ のとり方につい考える必要はありそう.

2 3.3.1 予測分布

2.1 問題設定

新たな入力 x に対する目標値 t の予測分布を考えたい.

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta) d\mathbf{w} \quad (3.57)$$

となる.

2.2 アルゴリズム

上の積分を計算すると (平方完成),

$$p(t|x, \mathbf{t}, \alpha, \beta) = N(t|\mathbf{m}_N \boldsymbol{\phi}(x), \sigma_N^2(x)) \quad (3.58)$$

ここで,

$$\sigma_N^2(x) = \frac{1}{\beta} + \boldsymbol{\phi}(x)^T S_N \boldsymbol{\phi}(x) \quad (3.59)$$

2.3 コード

予測分布の平均, 分散を求めるプログラム (test1.py)

```
new_x=pi
new_t=original_f(new_x)
print("x=",new_x,"t=",new_t)

"""データセットの作成"""
for N in [20,100,500]:
        x=data[:N,0]
        t=data[:N,1]
        for M in [4,10,20]:
                """Wの最適化"""
                A=np.zeros((M,M))
                W=np.zeros(M)
                P=np.zeros((N,M))
                I=np.identity(M)

                mu=[(m+0.5)*(2*pi/(M+1)) for m in range(M)]
                s=(2*pi)**2/12


                for n in range(N):
                        for m in range(M):
                                P[n,m]=tanh_basis(x[n],m,mu[m],s)
                                #polynomial_basis(x[n],m)
                                #sigmoid_basis(x[n],m,mu[m],s)

                alpha=10**(-10)
                beta=1.0/(0.3)**2


                S_N=inv(alpha*I+beta*np.dot(P.T,P))
                m_N=beta*np.dot(S_N,np.dot(P.T,t))

                phi=np.zeros(M)
                phi=[tanh_basis(new_x,m,mu[m],s) for m in range(M)]

                z_m=np.dot(m_N,phi)
                z_s=1.0/beta+np.dot(phi,np.dot(S_N,phi))

                print(N,"\t",M,"\t%.3f"%z_m,"\t%.3f"%z_s)
```

2.4 結果

$x = \pi$ についての予測分布を考えた. (正解は $t = -1$ である)

2.4.1 多項式基底

M \ N	20	100	500
4	-0.08	-0.06	-0.04
10	-0.85	-0.61	-0.70
20	6.89	-2.21	1.48

表 5: 予測分布の平均

M \ N	20	100	500
4	0.10	0.09	0.09
10	0.12	0.10	0.09
20	0.15	0.10	0.09

表 6: 予測分布の精度

2.4.2 ガウス基底

M \ N	20	100	500
4	-0.10	0.24	0.33
10	-0.57	-0.13	-0.20
20	0.58	-0.13	-0.20

表 7: 予測分布の平均

M \ N	20	100	500
4	0.11	0.09	0.09
10	0.12	0.10	0.09
20	0.12	0.10	0.09

表 8: 予測分布の精度

2.4.3 シグモイド基底

M \ N	20	100	500
4	-0.07	-0.05	-0.04
10	0.02	0.26	0.24
20	0.00	0.21	0.05

表 9: 予測分布の平均

M \ N	20	100	500
4	0.10	0.09	0.09
10	0.11	0.09	0.09
20	0.11	0.09	0.09

表 10: 予測分布の精度

2.4.4 tanh 基底

M \ N	20	100	500
4	-0.04	-0.02	-0.02
10	-0.80	-0.73	-0.91
20	0.94	-0.86	0.92

表 11: 予測分布の平均

M \ N	20	100	500
4	0.10	0.09	0.09
10	0.12	0.10	0.09
20	0.12	0.10	0.09

表 12: 予測分布の精度

2.5 まとめ

精度は 0.3^2 以上になるのだが, 学習がうまくいっていないものでもあまり大きくならなかった.