

3.1 線形基底関数モデル

平成 28 年 9 月 11 日

概要

PRML の「3.1 線形基底関数モデル」についての実装と考察

目次

1 問題設定	2
2 3.1.1 最尤推定と最小二乗法	2
2.1 アルゴリズム	2
2.2 コード	3
2.3 結果	4
2.3.1 多項式基底	4
2.3.2 ガウス基底	5
2.3.3 シグモイド基底	6
2.3.4 tanh 基底	6
2.4 まとめ	7
3 3.1.4 正則化最小二乗法	8
3.1 アルゴリズム	8
3.2 コード	8
3.3 結果	8
3.3.1 多項式基底	8
3.3.2 ガウス基底	9
3.3.3 シグモイド基底	10
3.3.4 tanh 基底	11
3.4 まとめ	12
4 3.1.3 逐次学習	12
4.1 アルゴリズム	12
4.2 コード	12
4.3 結果	13
4.3.1 多項式基底	13
4.3.2 ガウス基底	13
4.3.3 シグモイド基底	14
4.3.4 tanh 基底	15
4.4 まとめ	16

1 問題設定

データセット $D = \{(x_n, t_n) | n = 1, \dots, N\}$ があるとき, このデータセットをよく表す関数を作りたい(回帰).

そこで用いるのが, 以下の線形結合

$$y(x, W) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) \quad (3.2)$$

$$= \sum_{j=0}^{M-1} w_j \phi_j(x) = \mathbf{w}^T \boldsymbol{\phi}(x) \quad (3.3)$$

ここで基底関数 $\phi_j(x)$ の選び方は複数存在し, $\phi_j(x) = x$ のとき線形回帰, $\phi_j(x) = x^j$ のとき多項式線形回帰, ガウス基底 $\phi_j(x) = \exp\{-(x - \mu_j)^2/2s^2\}$ などいろいろある(詳しくは後述).

この線形結合でデータセットを表すためには関数とデータ点との誤差を小さくする必要があり, ここでは二乗和誤差の最小化(尤度関数最大化)または, 正則化二乗和誤差最小化(事後分布最大化)によってこれを達成する.

2 3.1.1 最尤推定と最小二乗法

2.1 アルゴリズム

目標変数 t とモデル関数 $y(x, \mathbf{w})$ との間にガウスノイズが含まれるとすると

$$p(t|x, \mathbf{w}, \beta) = N(t|y(x, \mathbf{w}), \beta^{-1}) \quad (3.8)$$

となる. データセットに対してこれを考えると

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

と尤度関数を定義することができる.

この尤度関数の対数つまり対数尤度関数は

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) &= \sum_{n=1}^N \ln N(t_n|\mathbf{w}^T \boldsymbol{\phi}(x_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln (2\pi) - \frac{\beta}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(x_n)\}^2 \end{aligned} \quad (3.11)$$

となり, この最大化のため \mathbf{w} で微分したものを $\mathbf{0}$ とすると

$$\mathbf{0} = \sum_{n=1}^N t_n \boldsymbol{\phi}(x_n)^T - \mathbf{w}^T \left(\sum_{n=1}^N \boldsymbol{\phi}(x_n) \boldsymbol{\phi}(x_n)^T \right) \quad (3.14)$$

となり, よって

$$\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad (3.15)$$

ここで $N \times M$ 行列 Φ は $\phi_{i,j} = \phi_j(x_i)$ を要素に持つ.

2.2 コード

プロットの部分は省略した (test.py)

```
"""基底関数の定義"""
def polynomial_basis(x,j):
    return x**j
def gauss_basis(x,j,mu,s):
    if j==0:
        return 1
    else:
        return np.exp(-(x-mu)**2/(2*s**2))
def sigmoid_basis(x,j,mu,s):
    if j==0:
        return 1
    else:
        return 1/(1+np.exp(-(x-mu)/s))
def tanh_basis(x,j,mu,s):
    if j==0:
        return 1
    else:
        return np.tanh(-(x-mu)/s)

"""Wの最適化"""
print("tanh_basis")
for N in [20,100,500]:
    for M in [4,10,20]:
        x=data[:N,0]
        t=data[:N,1]
        P=np.zeros((N,M))
        W=np.zeros(M)
        mu=[m*(2*pi/M) 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)

#W=(P^tP)^-1P^t T ムーアペンローズの疑似逆行列 pinv(P)を用いる
W=np.dot(pinv(P),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+=W[m]*tanh_basis(x,m,mu[m],s)
    return sum

"""表示"""
#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=%d\tM=%d\tE=%.3f" % (N,M>Error()))

```

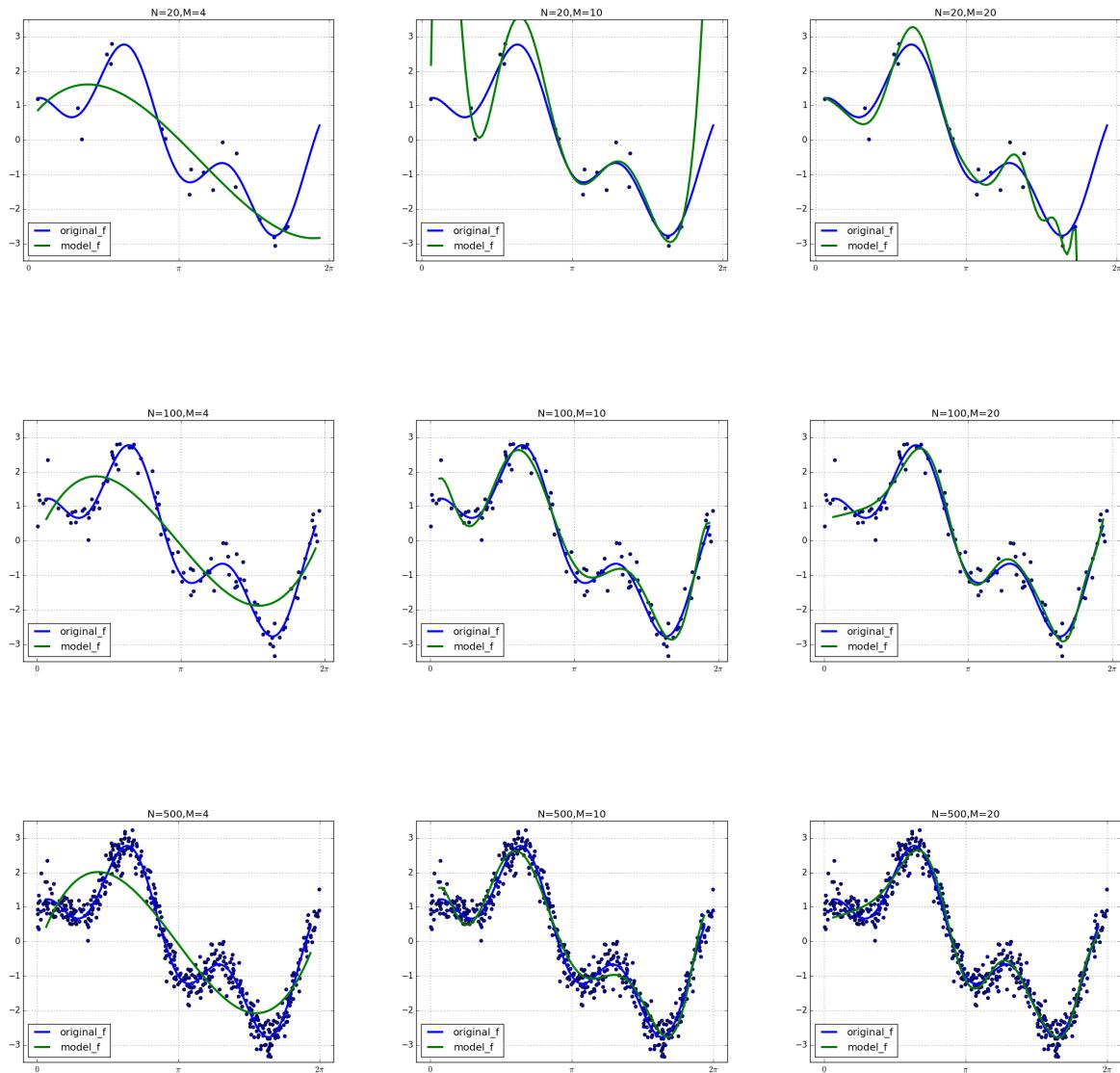
2.3 結果

2.3.1 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	1.23	0.82	0.81
10	6.18	0.45	0.40
20	5321	0.62	0.31

表 1: E_{RMS} の N, M との関係 (多項式基底)



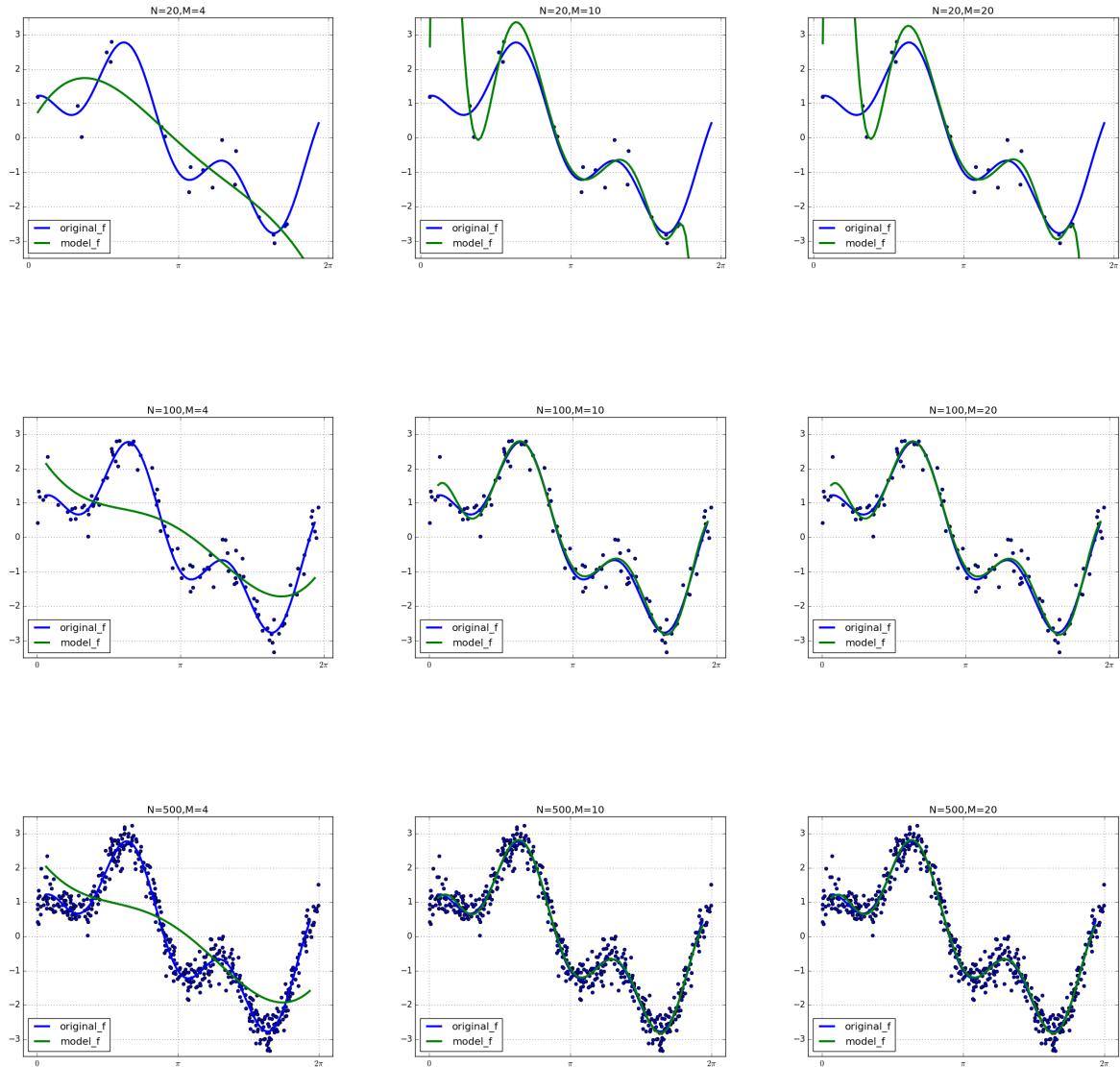
2.3.2 ガウス基底

基底 $\Phi(x)$ に $\phi_i(x) = \exp\left\{-\frac{(x-\mu_i)^2}{2s^2}\right\}$ 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。

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

M \ N	20	100	500
4	1.60	1.13	1.14
10	23.7	0.29	0.27
20	32.0	0.29	0.27

表 2: E_{RMS} の N, M との関係 (ガウス基底)

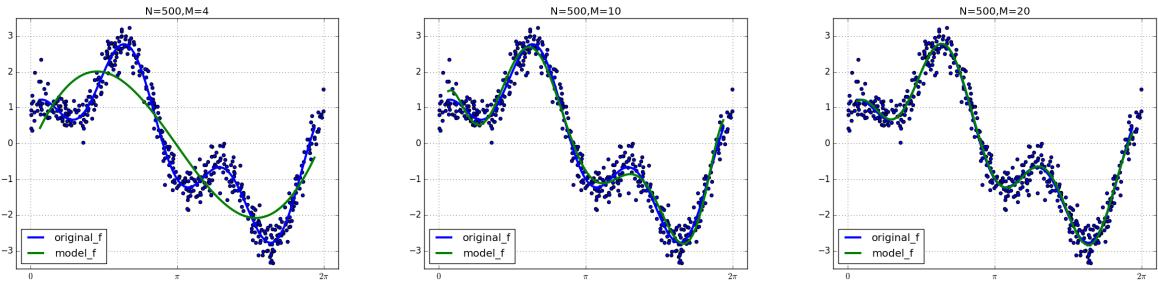
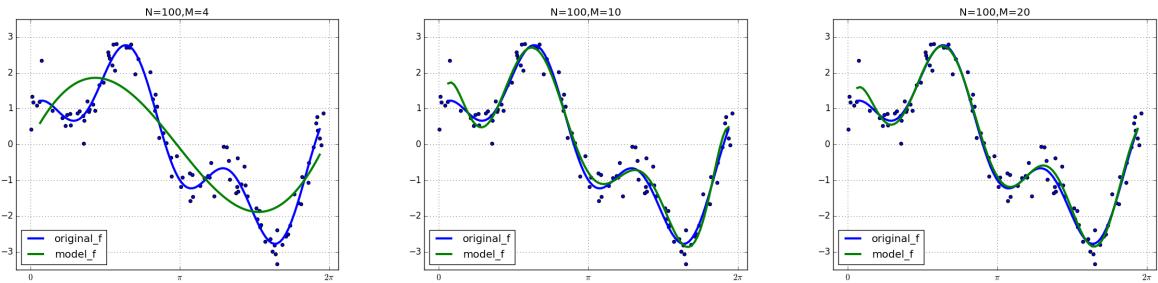
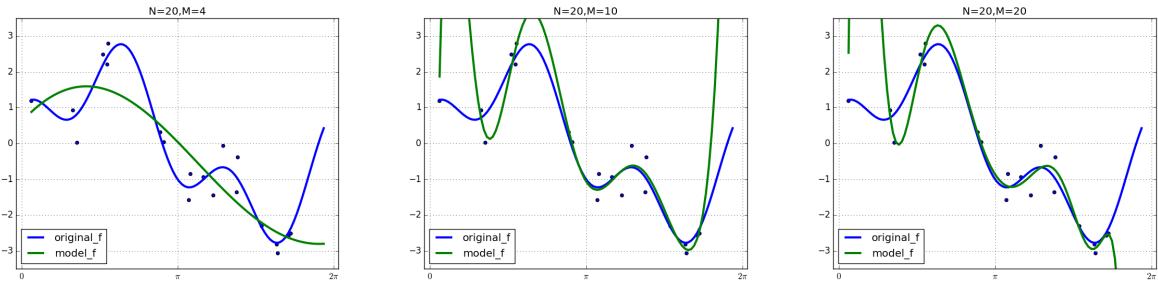


2.3.3 シグモイド基底

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	1.22	0.81	0.89
10	6.88	0.37	0.34
20	26.6	0.28	0.26

表 3: E_{RMS} の N, M との関係 (シグモイド基底)

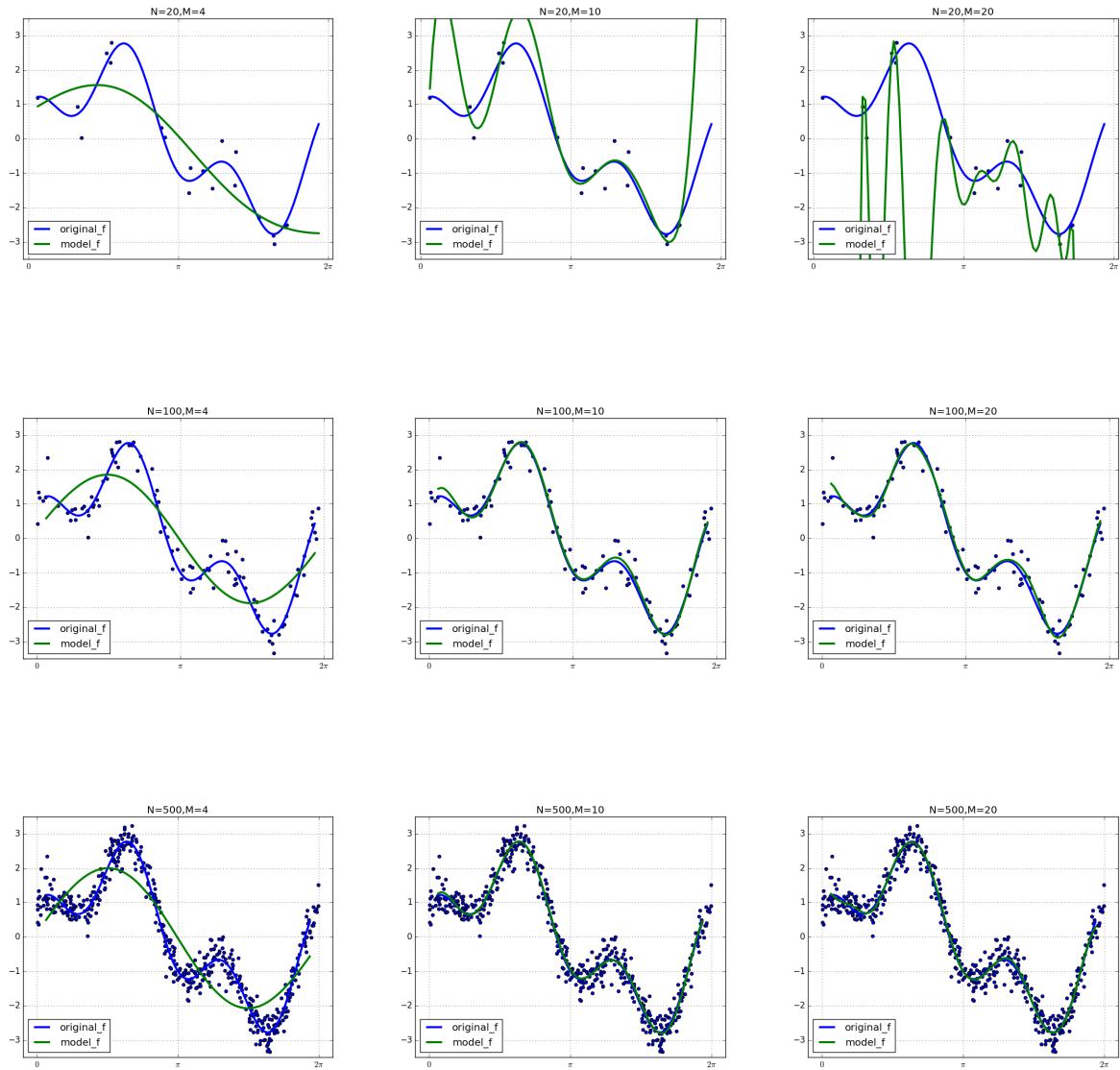


2.3.4 tanh 基底

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	1.19	0.78	0.77
10	5.82	0.28	0.27
20	2200	0.32	0.27

表 4: E_{RMS} の N, M との関係 (tanh 基底)



2.4 まとめ

データ数が少ないときに基底関数による差があった。データ数が多いときでは基底関数による差はあまりなかった。

gauss, sigmoid, tanh 基底などでは μ のとり方について考える必要はありそう。

3 3.1.4 正則化最小二乗法

3.1 アルゴリズム

2.2.2 節と同様に正則化二乗和誤差関数の最小化も行うことができ

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \quad (3.27)$$

を \mathbf{w} で微分したものを $\mathbf{0}$ とすると

$$\mathbf{w}_{ML} = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad (3.28)$$

が出る。

3.2 コード

最小二乗法のコードから一部変更する (test1.py).

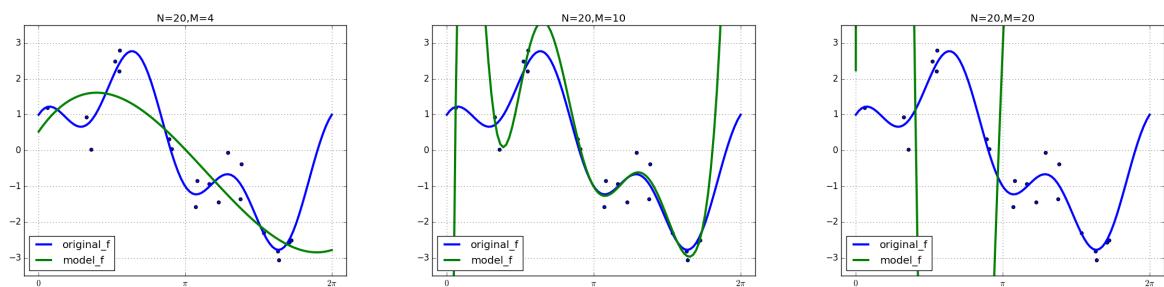
```
I=np.identity(M)
lam=10**(-10)
Q=inv(lam*I+np.dot(P.T,P))
W=np.dot(Q,np.dot(P.T,t))
```

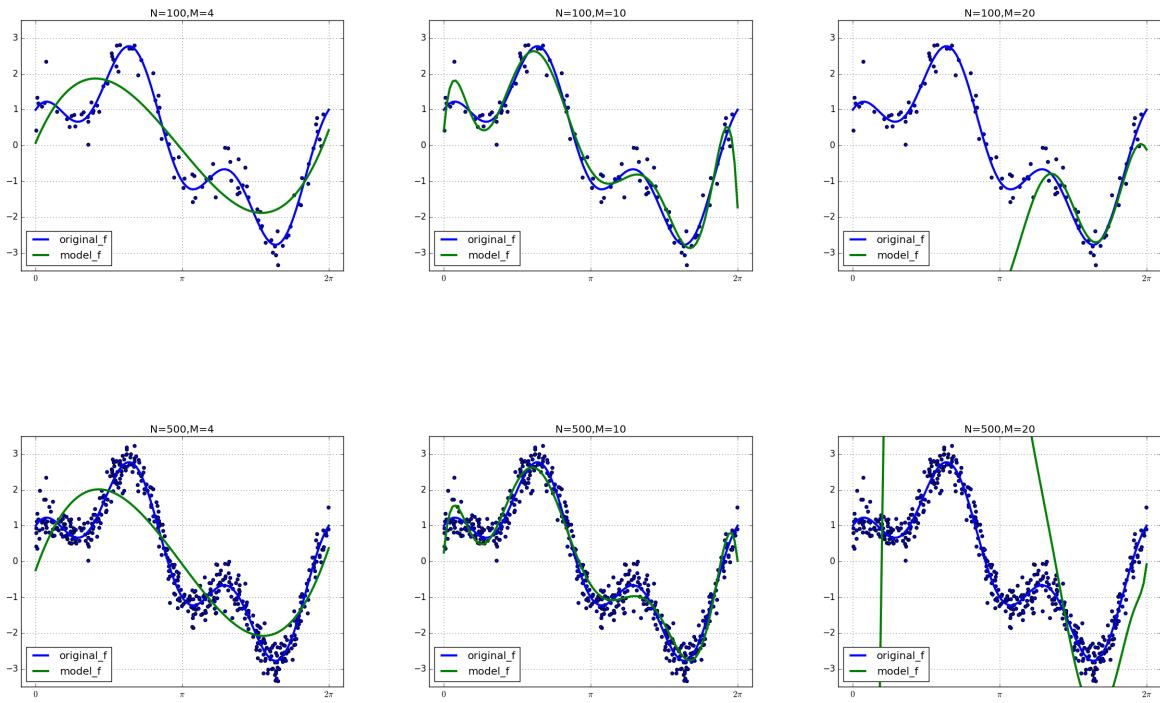
3.3 結果

3.3.1 多項式基底

M \ N	20	100	500
4	1.23	0.82	0.81
10	6.16	0.45	0.40
20	788	16.5	23.9

表 5: E_{RMS} の N, M との関係 (多項式基底)

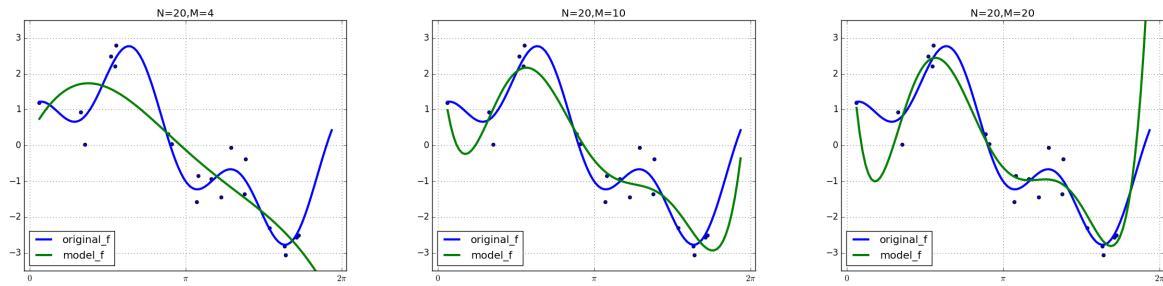


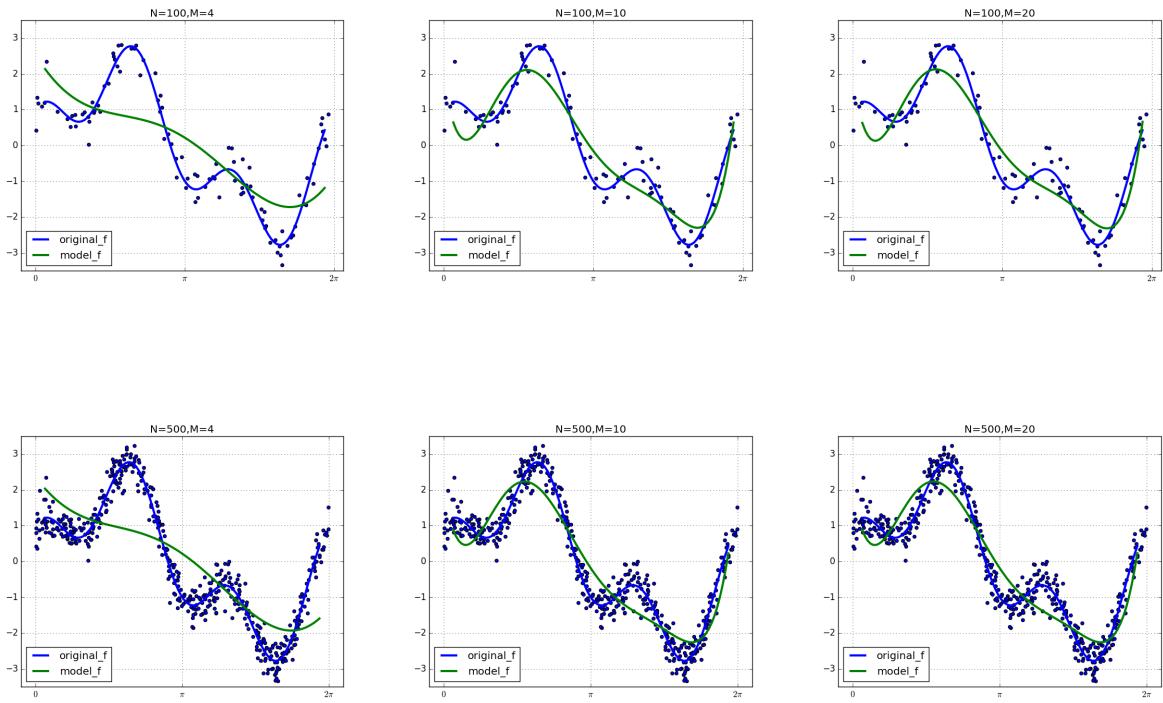


3.3.2 ガウス基底

M \ N	20	100	500
4	1.60	1.13	1.14
10	0.80	0.69	0.63
20	2.10	0.70	0.63

表 6: E_{RMS} の N, M との関係 (ガウス基底)

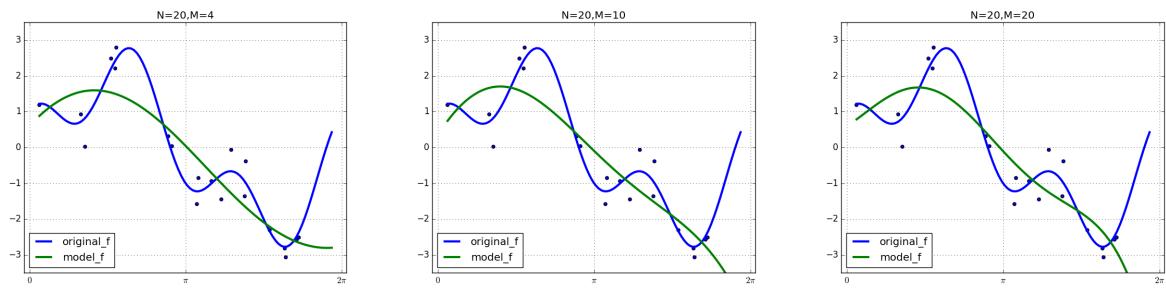


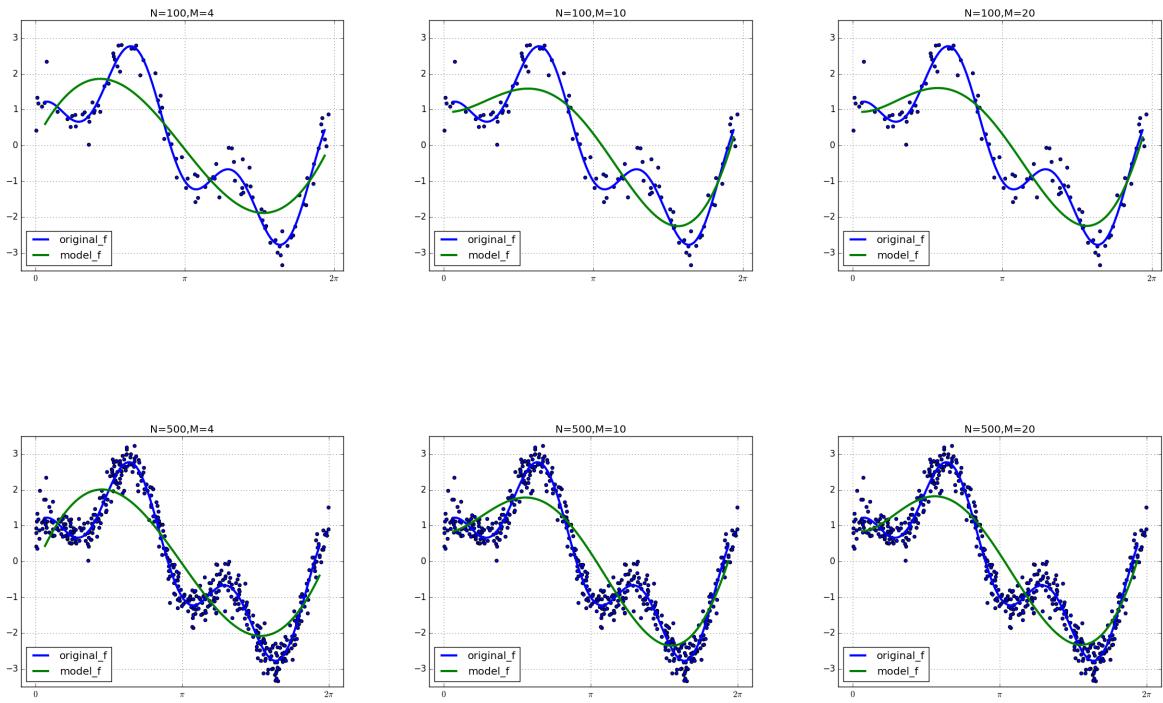


3.3.3 シグモイド基底

M \ N	20	100	500
4	1.22	0.81	0.79
10	1.89	0.75	0.72
20	1.91	0.74	0.71

表 7: E_{RMS} の N, M との関係 (シグモイド基底)

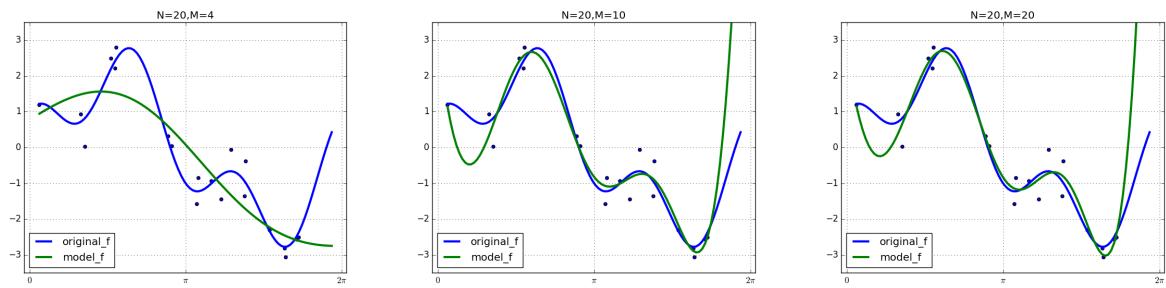


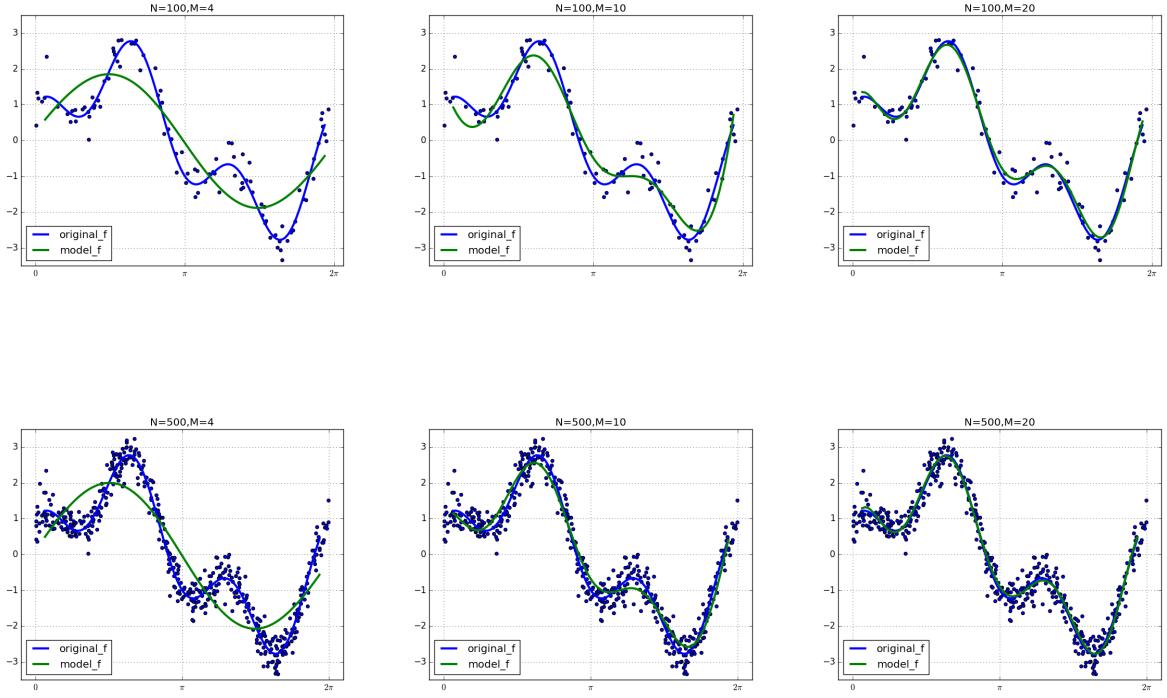


3.3.4 tanh 基底

M \ N	20	100	500
4	1.19	0.78	0.77
10	2.68	0.53	0.37
20	4.00	0.30	0.28

表 8: E_{RMS} の N, M との関係 (tanh 基底)





3.4 まとめ

基底関数を多項式にしたとき、不具合が生じている。
 正規化係数 $\lambda = 10^{-10}$ と非常に小さくとったが、過学習を抑制しすぎている気がする。
 一般化として lasso なども試してみたい。

4 3.1.3 逐次学習

4.1 アルゴリズム

確率的勾配降下法ではパラメータベクトル \mathbf{w} を

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$

で更新する。ここで、 η は学習率パラメータである。

誤差関数に二乗和誤差を用いた場合

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta(t_n - \mathbf{w}^{(\tau)T} \phi_n) \phi_n$$

となる。

4.2 コード

アルゴリズムの本体のみを記す (LMS.py).

```

eta=10**(-3)
for n in range(N):
    for m in range(M):
        P[n,m]=gauss_basis(x[n],m,mu[m],s)

norm=1

```

```

roop=0
while norm>10**(-6) or roop<N:
    n=roop%N
    W_old=W
    W=W_old-eta*(t[n]-np.dot(W_old,P[n,:]))*P[n,:]
    norm=np.linalg.norm(W-W_old)
    roop+=1

```

4.3 結果

4.3.1 多項式基底

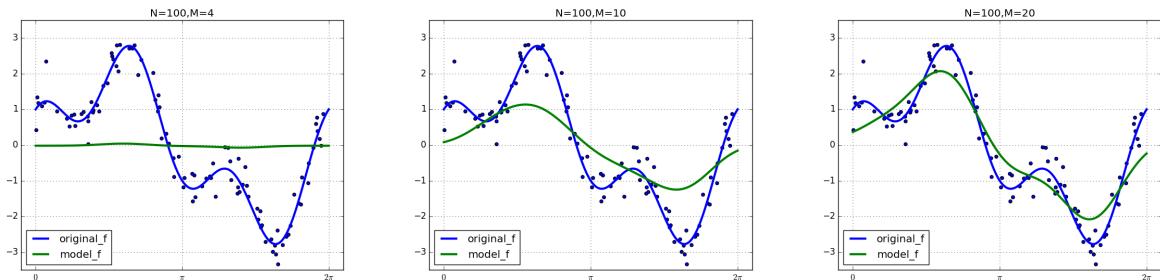
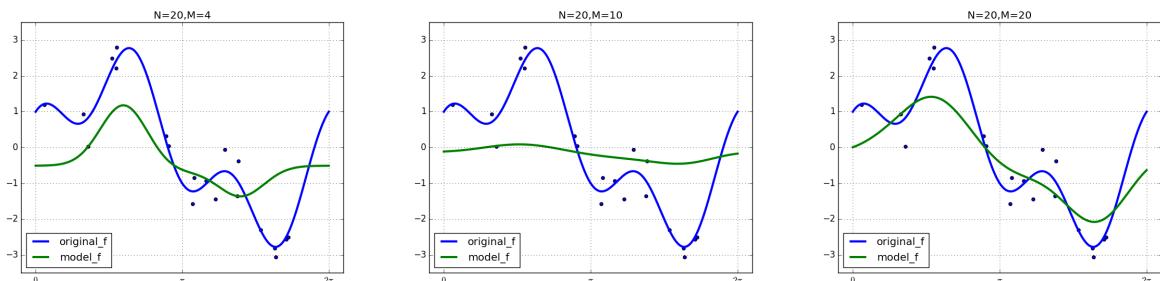
上手くいかなかった、全てに対して誤差 = nan が返ってきた。

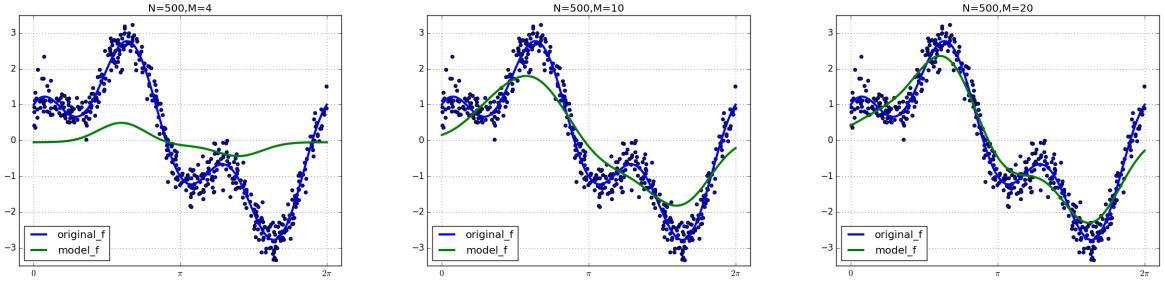
4.3.2 ガウス基底

$s = 0.5$ とした。

M \ N	20	100	500
4	1.20	1.62	1.42
10	1.53	0.98	0.70
20	0.89	0.59	0.52

表 9: E_{RMS} の N, M との関係 (ガウス基底)



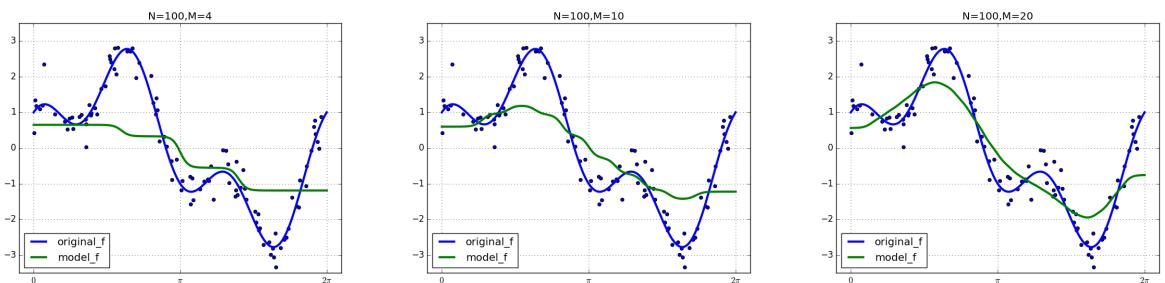
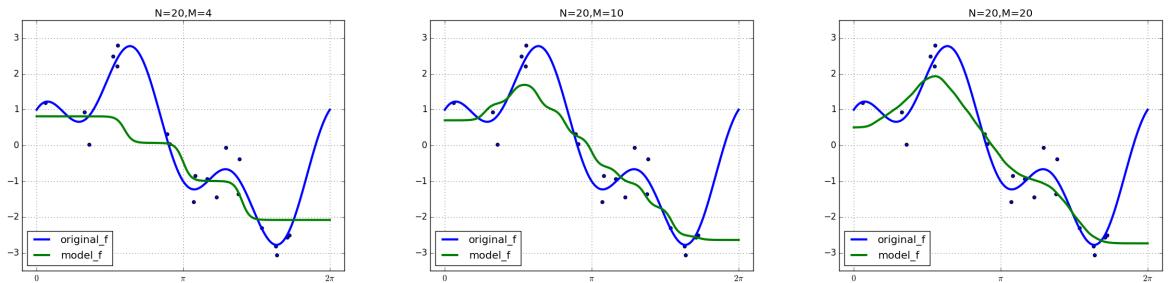


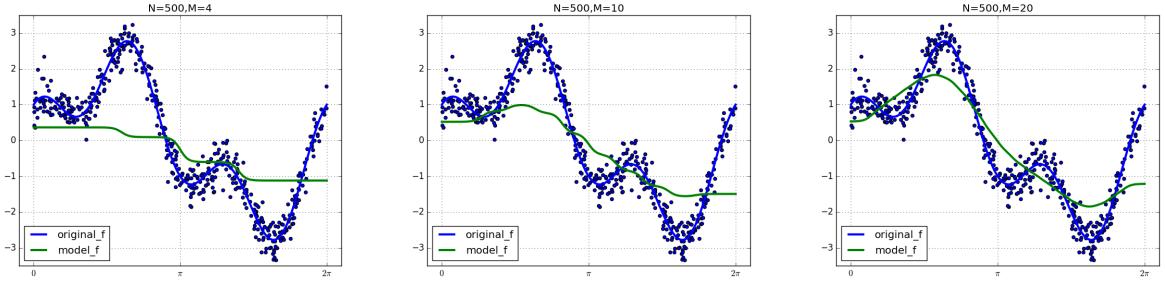
4.3.3 シグモイド基底

$s = 1/2$ とした。

M \ N	20	100	500
4	1.34	1.22	1.35
10	1.40	1.00	1.08
20	1.11	0.70	0.78

表 10: E_{RMS} の N, M との関係 (シグモイド基底)



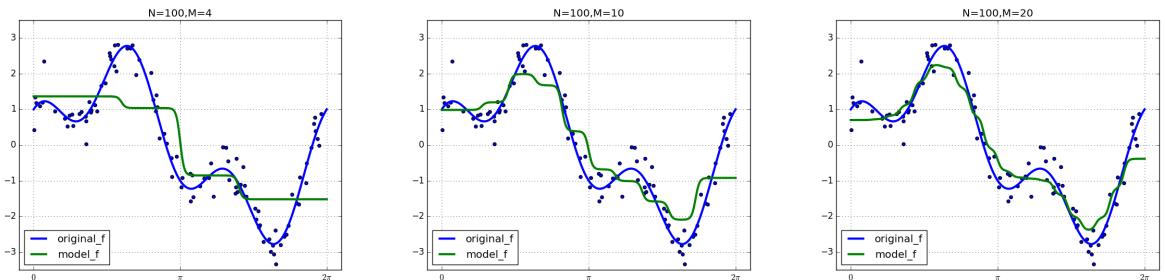
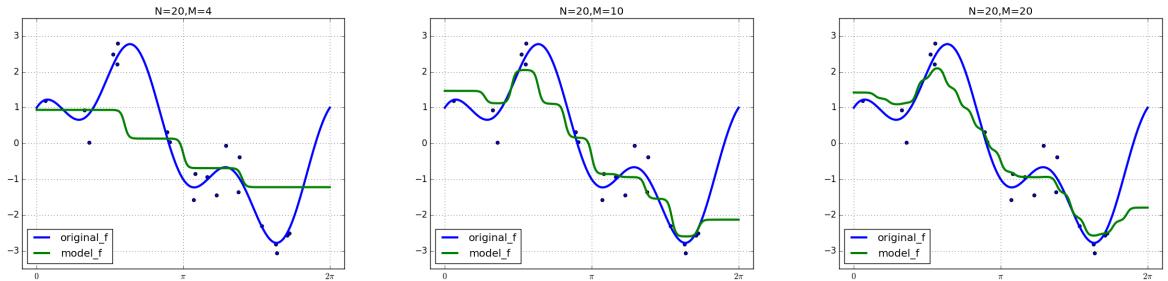


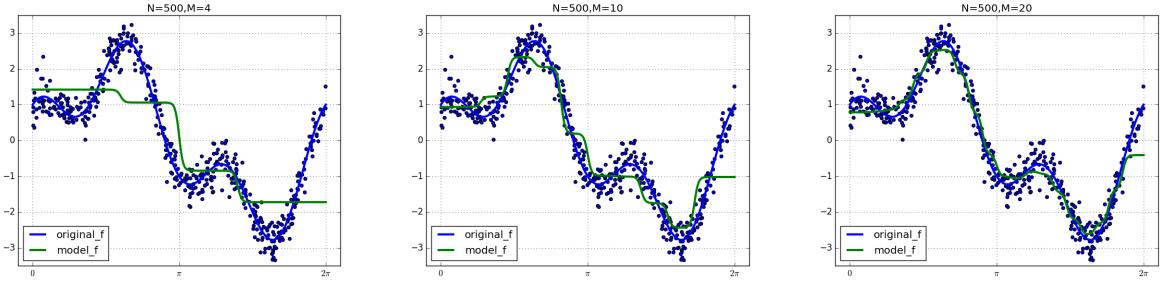
4.3.4 tanh 基底

$s = 1/12$ とした。

M \ N	20	100	500
4	1.24	0.99	1.00
10	0.99	0.70	0.61
20	0.87	0.49	0.40

表 11: E_{RMS} の N, M との関係 (tanh 基底)





4.4 まとめ

多項式基底では上手くいかなかった。バッチ処理の方が安定している。
 ガウス基底などでは, s の値を最適化したい。
 ステップサイズは, 適宜変える必要があるようだ。収束するよう保証するような値でなければ
 ならない。