1.1 例:多項式曲線フィッテイング

平成 28 年 10 月 30 日

概 要

PRML の「1.1 例:多項式曲線フィッテイング」についての実装と考察

目 次

1　問題設定

訓練集合として, N 個の観測値 x_n を並べた $\mathbf{x}= (x_1, x_2, \ldots, x_N)^T$ と, そのぞれぞれに対応する観測値 t_n を並べた $\mathbf{t}= (t_1, t_2, \ldots, t_N)^T$ が与えられたとする. ただし, この観測値には確率的な誤差が含まれているものとする. このとき, この観測値を与えた関数の形を知りたいが, ここでは次のような多項式で予測性能の良いものを探すことにする.

$$\begin{aligned} y(x, \mathbf{w}) &= w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M \\ &= \sum_{j=0}^{M} w_j x^j \quad (1.1) \end{aligned}$$

ここで, $\mathbf{w} = (w_0, w_1, \ldots, w_M)^T$ はパラメータで, 誤差関数を最小化することで最適化する. この誤差関数には主に次の二乗和誤差が用いられる.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 \quad (1.2)$$

また, もう一つのパラメータ M についても最適化する.

2　アルゴリズム

まずは M を固定し (1.2) で与えた二乗和誤差の最小化について考える.
$E(\mathbf{w})$ を w_i で微分すると

$$\begin{aligned} \frac{dE(W)}{dw_i} &= \sum_{n=0}^{N} \left\{ \sum_{j=0}^{M} w_j x_n^j - t_n \right\} x_n^i \\ &= \sum_{j=0}^{M} \left\{ \sum_{n=0}^{N} x_n^{i+j} \right\} w_j - \sum_{n=0}^{N} t_n x_n^i \\ &= 0 \end{aligned}$$

これは, 各 $i (i = 0, 1, \ldots, M)$ で成り立つため, これをまとめると

$$A\mathbf{w} = T$$

となる. ただし, A は $M+1$ 次正方行列, T は $M+1$ 次元ベクトルで,

$$A_{i,j} = \sum_{n=1}^{N} x_n^{i+j}, \ T_i = \sum_{n=1}^{N} x_n^i t_n$$

である.

多項式曲線フィッティング

1. 観測値 $\{x_n, t_n\}$ から A, T を計算する.
2. 線形方程式 $A\mathbf{w} = T$ をとくことで, パラメータ $\mathbf{w}$ を求める.

3 コード

二乗和誤差の最小化による多項式曲線フィッテイング (test.py)

```
#Wの最適化
A=np.zeros((M+1,M+1))
W=np.zeros(M+1)
T=np.zeros(M+1)

#二乗和誤差の最小化
for i in range(M+1):
        for j in range(M+1):
                for n in range(N):
                        A[i,j]+=trainx[n]**(i+j)
        for n in range(N):
                T[i]+=trainx[n]**i*traint[n]

#線形方程式(AW=T)を解くことでパラメータWを求める
W=np.linalg.solve(A,T)

#求まったパラメータからモデル関数を作り
def model_f(x):
        temp=0
        for m in range(M+1):
                temp+=W[m]*x**m
        return temp
```

4 データ数 N, パラメータ M について

4.1 各 N, M に対する二乗和誤差

データ数 N は {5,10,50,100,500}, パラメータ M は {2,4,10,20,50} のそれぞれに対して二乗和誤差 E の評価を行う.

M \ N	5	10	50	100	500
2	0.65	3.98	25.02	65.66	339.73
4	0.02	2.55	11.94	26.21	162.29
10	0.00	0.00	3.80	8.24	36.83
20	0.00	0.00	99.44	5.06	25.08
50	0.00	0.00	1.81	9.20	25.05

表 1: E と N,M の関係

まず, M が大きくなるほど E は小さくなる傾向があるが, これは M が大きくなるに従って表現できるモデルが複雑になるためであると考えられる. また, 二乗和誤差 E がデータ数 N について単調増加することが分かるが, これは単純に和の数が多くなるためである. このため, N が異なるときに E を比較するのは妥当ではなくなる. ここで, 次の形の平均二乗平方根誤差 E_{RMS} を導入する.

4.2 平均二乗平方根誤差の利用

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N} \quad (1.3)$$
$$= \sqrt{\frac{1}{N}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2}$$

ただし, $\mathbf{w}^*$ は二乗和誤差 E を最小にする $\mathbf{w}$ である. そして, この二乗平方根誤差 E_{RMS} は目的変数 t と同じ尺度であることが分かる.
これを用いて, 平均二乗平方根誤差 E_{RMS} と N,M の関係を考察する.

M \ N	5	10	50	100	500
2	0.51	0.89	1.00	1.15	1.17
4	0.09	0.71	0.69	0.72	0.81
10	0.00	0.00	0.39	0.41	0.38
20	0.00	0.00	1.99	0.32	0.32
50	0.00	0.00	0.25	0.31	0.32

表 2: E_{RMS} と N,M の関係

E_{RMS} についても M に対する減少傾向はみられる. N に対しては依存関係が弱まっていることが分かる. また, よくモデル化されているものでも誤差が 0.3 より小さくならないが, これは観測値に含まれるガウスノイズの分散に近づくものと考えられる.

4.3 ホールドアウト集合の使用

次に凡化性能を見る. ここではホールドアウト集合として新たな 50 個のデータを用いて誤差関数を評価する. また, 以下では誤差関数の評価には E_{RMS} とホールドアウト集合を用いることにする.

M \ N	5	10	50	100	500
2	1.15	1.12	1.15	1.20	1.17
4	1.18	0.84	0.78	0.79	0.77
10	5.37	55.31	0.43	0.47	0.39
20	22.75	15.90	4.91	0.32	0.27
50	443.21	89932	0.84	0.39	0.27

表 3: E_{RMS} の N,M との関係

モデル化が不十分であるものは誤差が大きくなっているが, 極端に大きくなったものは過学習を起こしていると考えられる. また, 誤差が小さくなっているものもあるが, これはデータに依存しただけである. ただ, 依然として 0.3 付近で収束している.

4.4　各 N, M に対するモデル関数のプロット

そして, 各 N,M に対して得られたモデルをプロットすると

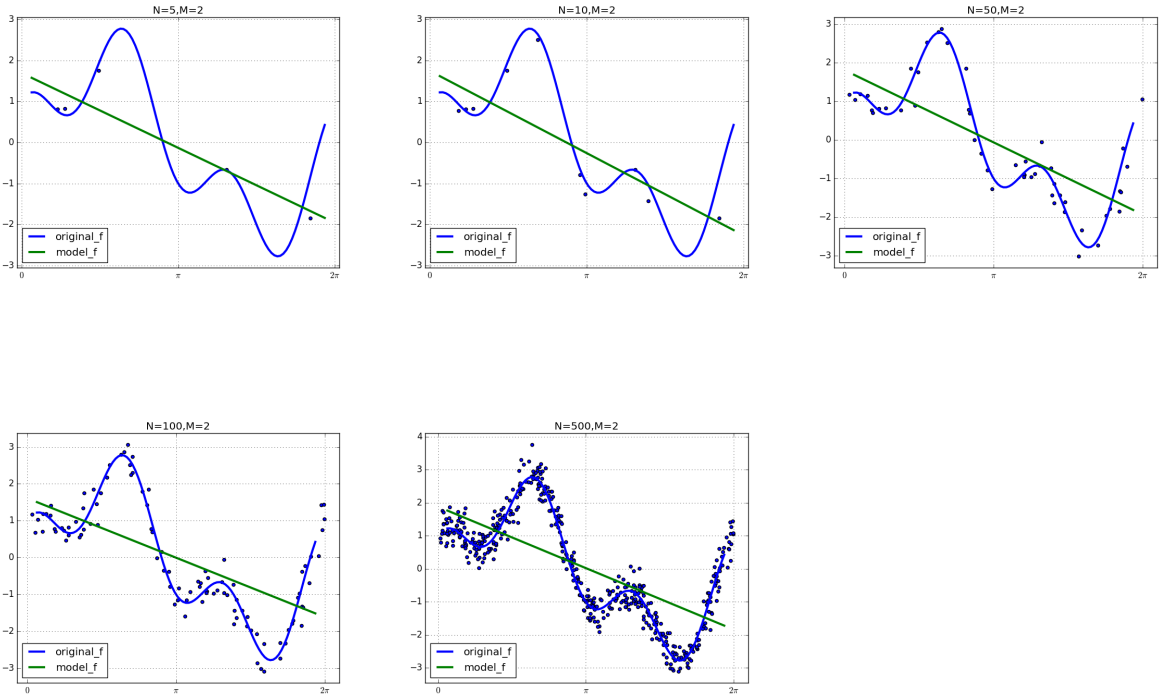


図 1: M=2 のときの多項式曲線フィッテイング

M=2 のとき N によらず大体同じで, データを表現するには不十分なモデルである.

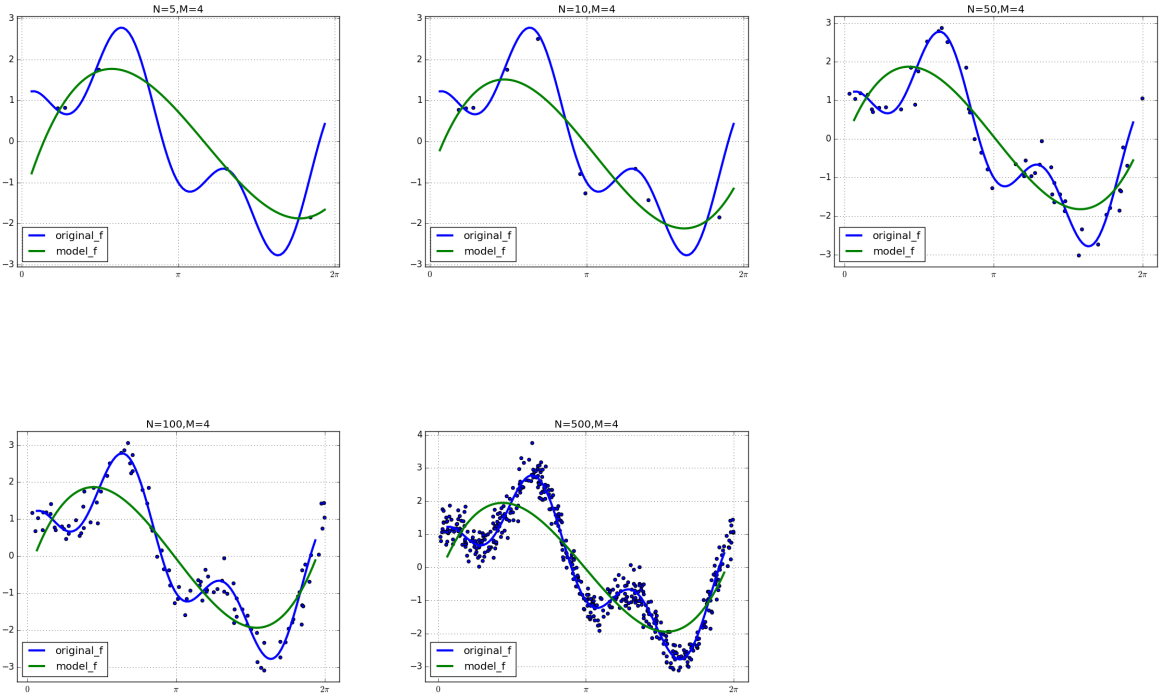


図 2: M=4 のときの多項式曲線フィッテイング

M=4 のとき N が一定量を超えると大体同じ．データ数が少なすぎると問題である．

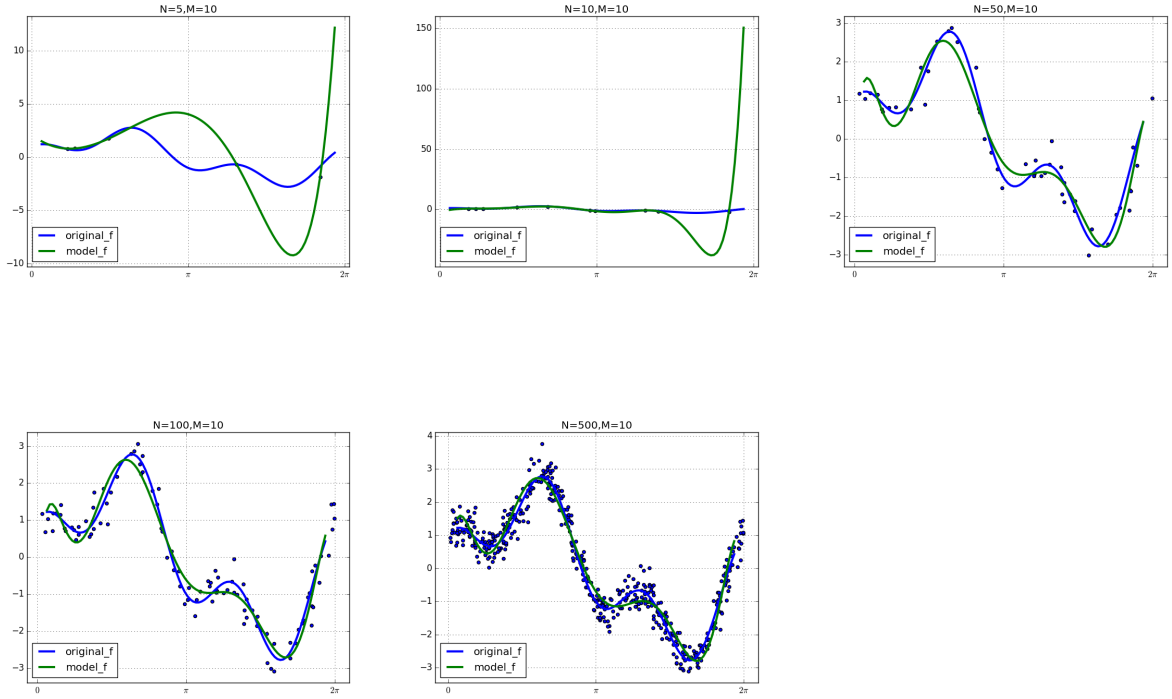


図 3: M=10 のときの多項式曲線フィッテイング

M=10 のとき N が一定量を超えれば, しっかりモデル化できていて, モデル化には M はそのデータを表現するのに必要な量が決まっていてそれを超える必要があるように思える．

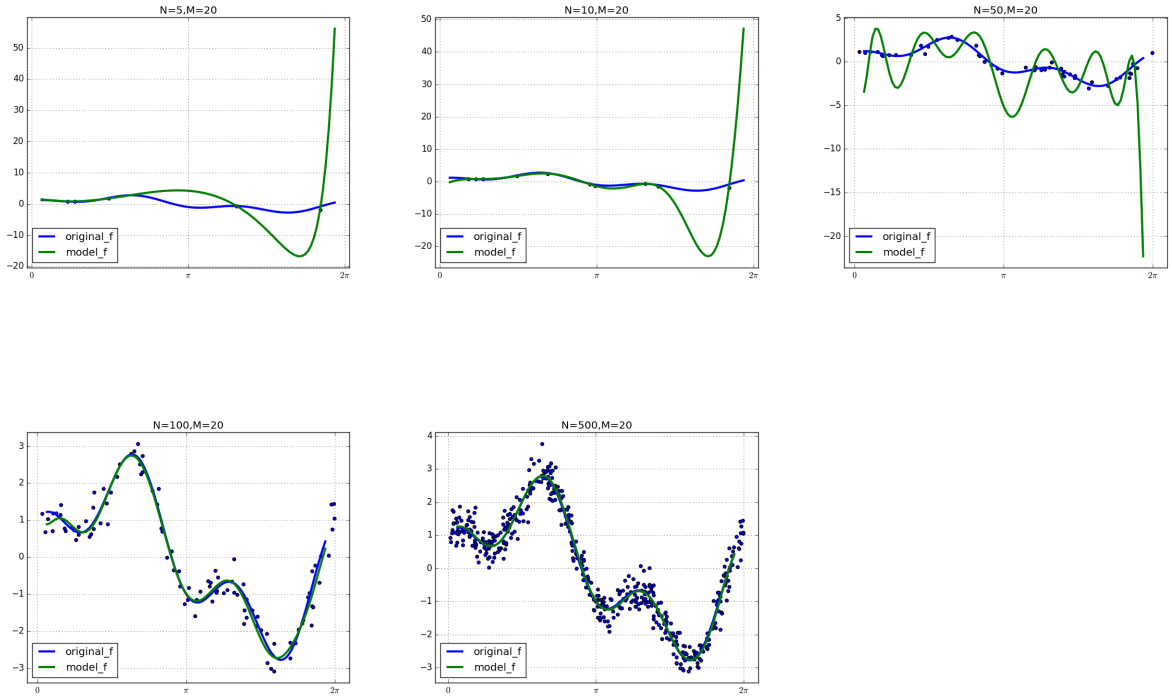


図 4: M=20 のときの多項式曲線フィッテイング

M=20, N=50 では過学習を起こしていることが分かる. また, M=20,N=5,10 のとき区間の端では極端にずれているが, これはデータが区間の端にないことが原因である.

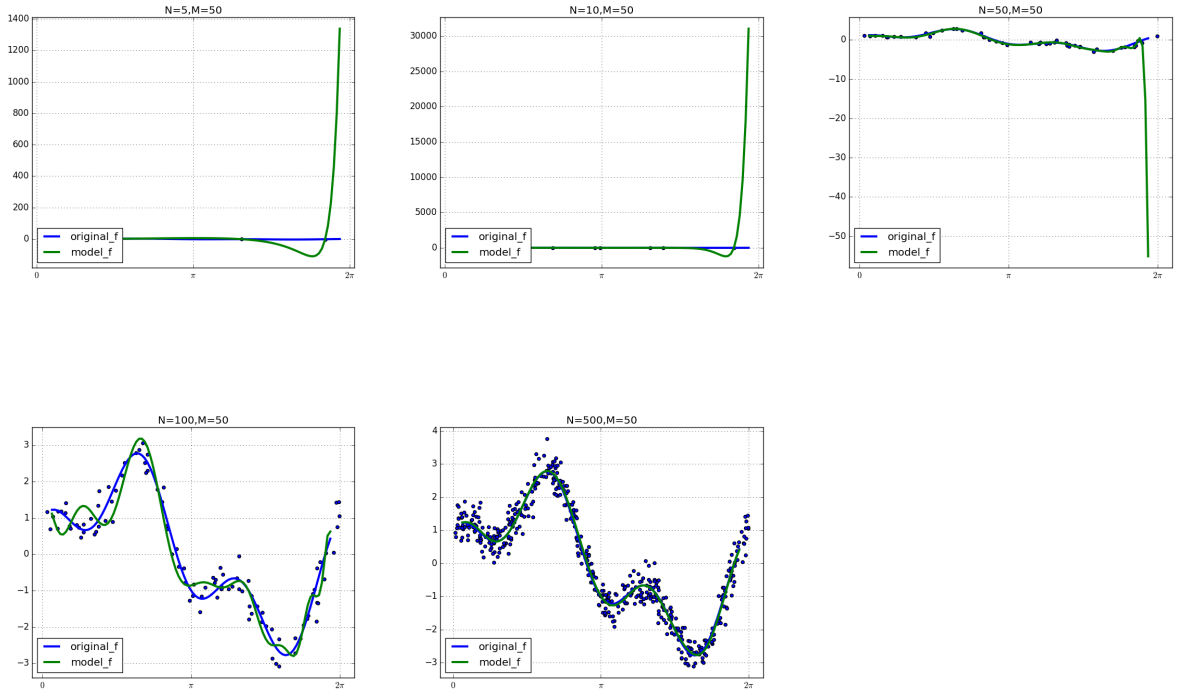


図 5: M=50 のときの多項式曲線フィッテイング

図から学習の程度を考察してみる. 1 は不十分, 2 は最適, 3 は過学習とすると.

M \ N	5	10	50	100	500
2	1	1	1	1	1
4	1	1	1	1	1
10	3	3	2	2	2
20	3	3	3	2	2
50	3	3	3	3	2

表 4: 学習の程度

まず, モデルは十分に複雑でなければ学習は不十分となり, 複雑すぎると過学習を起こす. そしてデータ数は多ければ多いほど過学習を起こしにくくなることが分かる. そして, データ数が多いとき予想以上の成果を見せた.

5 正規化二乗和誤差

M=20,N=50 は分かりやすく過学習を起こしているので, このときに次の形の正規化二乗和誤差関数を使って, どのような変化があるかを考察する.

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \quad (1.4)$$

ここで, 正規化係数 λ はパラメータで 0,1e-50,1e-10,1,5,50 で試してみる.

λ	0	1e-50	1e-10	1	5	50
E_{RMS}	4.907	0.280	0.273	0.295	0.301	1.643

表 5: E_{RMS} と λ の関係

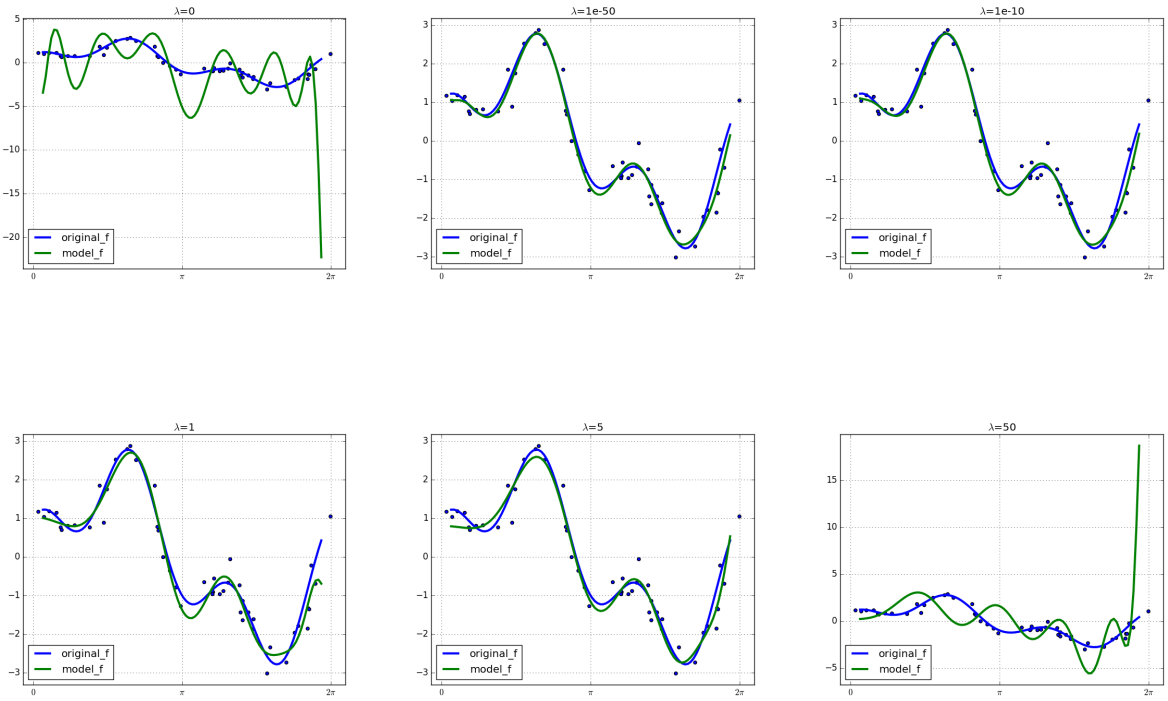


図 6: E_{RMS} と λ の関係

正規化係数 λ はかなり小さくても効果を発揮することが分かった. 逆に, 大きすぎると学習の妨げになりそうである.

6 思いついたこと

6.1 M が大きいときの w の要素について

M が大きいとき **w** のいくつかがとても小さくなっている. このことからその成分を 0 にしてみた.
M=50, N=500 のときに対して 10e-10 以下の要素を 0 にすると

```
W=
[  1.18772302e+00  -5.74513872e-01   8.84769501e+00  -3.11610550e+01
   4.02921329e+01  -2.41159213e+01   7.04137841e+00  -8.01722578e-01
  -4.52652190e-02   1.31234208e-02   8.11939389e-04  -1.26616240e-04
   5.33994625e-06  -9.87193242e-06   2.44168475e-07   1.90581072e-07
   7.56015177e-09   4.01187747e-09   2.31357673e-10  -4.55590088e-10
   4.52012093e-11   1.96015108e-12  -1.08281690e-12   1.38535970e-14
```

```
  -4.07393992e-15   7.86368481e-16   8.63539790e-16  -2.92625061e-17
  -5.30615445e-18   8.23558549e-18  -1.30040441e-18   1.73495154e-19
  -4.99605881e-20  -2.97806955e-21  -3.20092147e-22   6.85636594e-23
   3.54800151e-23   2.74296225e-24   2.44810028e-26  -1.14816334e-25
  -6.44204121e-27  -2.64901874e-28   2.39378222e-28  -2.14256742e-29
  -5.33222624e-30   1.20661623e-30  -1.02193953e-31   3.55519400e-32
  -3.99659766e-33   5.34007516e-35]
E= 0.274169379341
が
W=
[  1.18772302e+00  -5.74513872e-01   8.84769501e+00  -3.11610550e+01
   4.02921329e+01  -2.41159213e+01   7.04137841e+00  -8.01722578e-01
  -4.52652190e-02   1.31234208e-02   8.11939389e-04  -1.26616240e-04
   5.33994625e-06  -9.87193242e-06   2.44168475e-07   1.90581072e-07
   7.56015177e-09   4.01187747e-09   2.31357673e-10  -4.55590088e-10
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00]
E= 57733.5908491
```

に代わるが, 良い結果にならなかった.
また, この結果を見ると 20 番目以降の要素はすべて 10e-10 以下であることが分かるが, このことから M=20 としてよいのではないかと思われる. 実際 N=500 のとき M=20 と M=50 の場合では $E_{RMS} = 0.39$ と $E_{RMS} = 0.27$ でモデル関数の見た目はほぼ変わらない. さらに過学習を抑えるという意味でよいほうにはたらくと思う. ただし, ここでは 1e-10 以下の要素を 0 にしたが, この値の最適化という問題は残る.

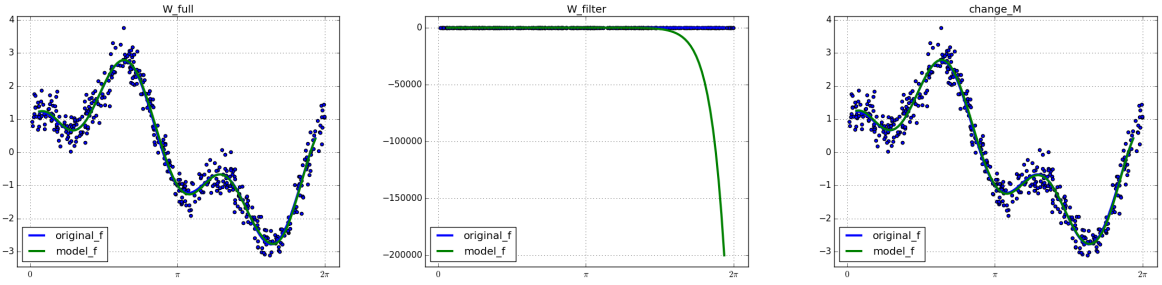


図 7: W についての考察にかかわる多項式曲線フィッテイング

6.2 N が小さいときにデータをかさましする方法

観測値に対して, 新たに分散の小さいガウスノイズを加えて新たにデータを作る方法について試してみる. ただ, N が小さすぎるとその観測値は信頼できないので, ここでは M=20 とし, データ数

N=10 からデータ数を増やしてみる．ここでは, x,t それぞれに分散 0.1 のガウスノイズを加えた．

N	10	20	50	100	500	1000
E_{RMS}	15.90	139.15	7.55	154.96	1.88	2.67

表 6: E_{RMS} とかさましデータ数 N との関係

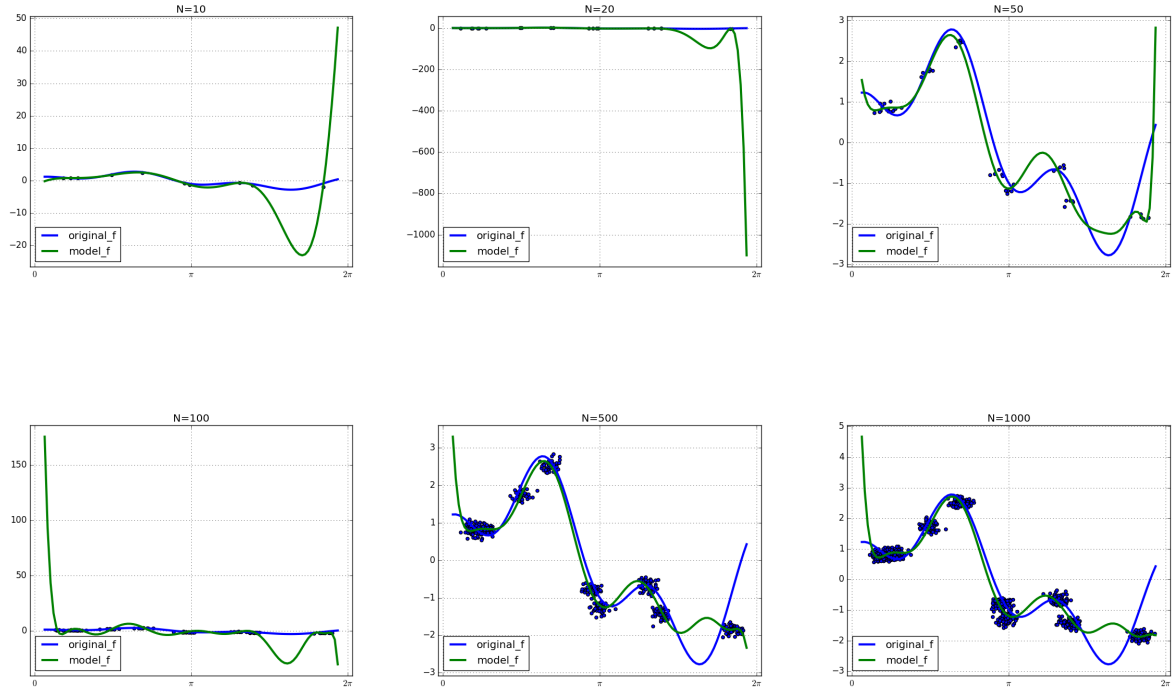


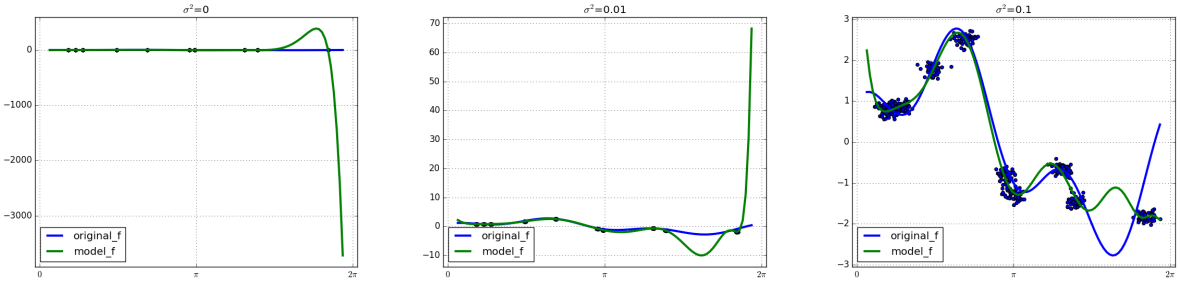
図 8: 人工データでデータ数 N をかさましたときの多項式曲線フィッテイング

N=500,1000 のときを見るとデータのかさましは一定の効果があるように思える．ただ, この場合にも右下の部分は表現できていない．しかし, この部分はモデル化のしようがないように思える．

また, 加えるガウスノイズの大きさについてであるが, これについても比較してみる

σ^2	0	0.01	0.1	0.3	(0.4,0.1)	(0.1,0.4)
E_{RMS}	1682	68.28	1.00	0.80	0.79	6.91

表 7: E_{RMS} とかさましデータ数 N との関係



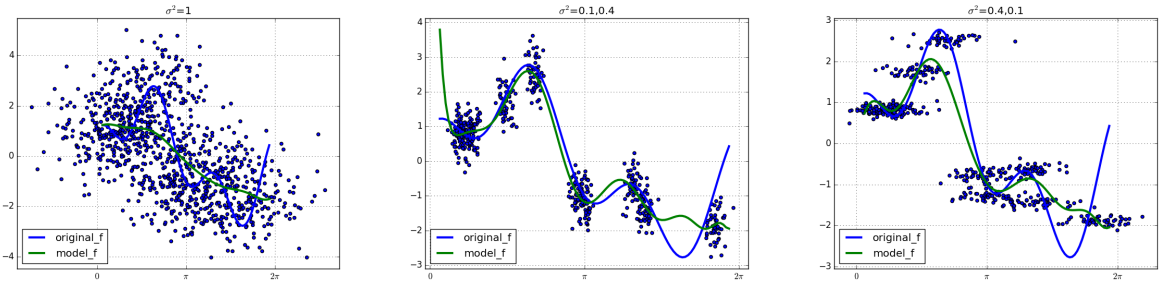


図 9: 人工データでデータ数 N をかさましたときの多項式曲線フィッテイング

分散が小さすぎると, 元データに依存したままで意味はなく, 大きくしすぎるとモデルはなだらかになりすぎる. さらに, 等方的なものが良いか否かはデータに依存するだろう.

7 多次元への拡張

7.1 $(x_1, x_2) \to t$

モデルを次のように定義し.

$$y(x_1, x_2, \mathbf{w}) = \sum_{j=0}^{M} w_{1,j} x_1^j + w_{2,j} x_2^j$$

これより, 二乗和誤差を $\mathbf{w}_i$ で微分すると,

$$A_1 \mathbf{w}_1 = T_1 \quad A_2 \mathbf{w}_2 = T_2$$

これより, 係数 $\mathbf{w}_1, \mathbf{w}_2$ が決まる.

7.2 $x \to (t_1, t_2)$

モデルを 2 こ作ればいい.

7.3 $(x_1, x_2) \to (t_1, t_2)$

上二つを組み合わせればよい.

8 まとめ

1. モデルサイズ M はデータ点数 N によって相対的に決まると考えられる.
2. 正規化二乗和誤差を用いることは一定の効果があった. ただ, 正規化係数を決定する式が必要であると考える.
3. M の大きさを W の値から制限する方法はなかなかいい感じ. もう少し練る必要があるように感じる.
4. データをかさましする方法は一定の効果はあるものの, 学習の負担を増すことになる. ほかの方法で (例えば, データ自体の価値に重みを与える) 効率的な方法が必要であろう.