6.1 双対表現

平成 29 年 3 月 29 日

概 要

PRML の「6.1 双対表現」についての実装と考察

目 次

1 問題設定 **2**

2 アルゴリズム **2**

3 コード **2**

4 結果 **3**

4.1 差の絶対値 3
4.2 差の絶対値の二乗 4
4.3 差の絶対値の累乗 5
4.4 ガウスカーネル 7

5 まとめ **8**

1　問題設定

データセット $D = \{(x_n, t_n)|n = 1, \ldots, N\}$ 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).

2　アルゴリズム

今考える線形回帰は以下の正則化二乗和誤差関数の最小化でパラメータを求める.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \quad (6.2)$$

$J(\mathbf{w})$ の W についての勾配を 0 とすると

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N}a_n\boldsymbol{\phi}(\mathbf{x}_n) = \Phi^T\mathbf{a} \quad (6.3)$$

ここで $\mathbf{a}$ は a_n を要素に持つベクトル

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\} \quad (6.4)$$

これから, パラメータ $\mathbf{w}$ のかわりに $\mathbf{a}$ について考え直す.

$$\begin{aligned} J(\mathbf{a}) &= \frac{1}{2}\sum_{n=1}^{N}\{(\Phi^T\mathbf{a})^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}(\Phi^T\mathbf{a})^T(\Phi^T\mathbf{a}) \\ &= \frac{1}{2}\mathbf{a}^T\Phi\Phi^T\Phi\Phi^T\mathbf{a} - \mathbf{a}^T\Phi\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\Phi\Phi^T\mathbf{a} \quad (6.5) \end{aligned}$$

ここで次で定義されるカーネル関数を要素に持つグラム行列について考える.

$$K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^T\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \quad (6.6)$$

つまり, $K = \Phi\Phi^T$. これを用いて

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^TKK\mathbf{a} - \mathbf{a}^TK\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^TK\mathbf{a} \quad (6.7)$$

また (6.4) より

$$\mathbf{a} = (K + \lambda I)^{-1}\mathbf{t} \quad (6.8)$$

となり, モデルは

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(K + \lambda I)^{-1}\mathbf{t} \quad (6.9)$$

ここで, $\mathbf{k}(\mathbf{x})$ は $k(\mathbf{x}_n, \mathbf{x})$ を要素に持つベクトル.

3　コード

アルゴリズムの本体だけ記す (test.py)

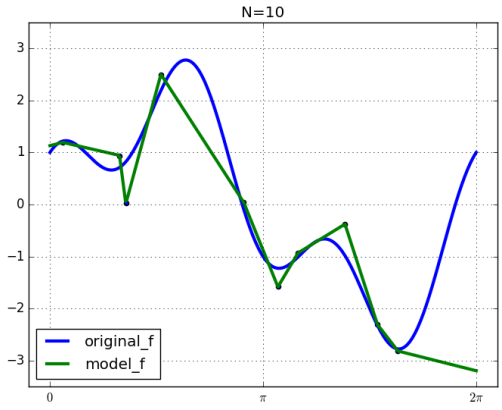
```
"""カーネル関数の定義"""
def gauss(x,z,s):
        return np.exp(-(x-z)**2/(2*s**2))
def norm(x,z):
        return (x-z)**2
def Abs(x,z):
        return abs(x-z)

"""wの最適化"""
print("abs")
for N in [10,30,50,100]:
        x=data[:N,0]
        t=data[:N,1]
        K=np.zeros((N,N))
        s=2*(2*pi)/N

        for n in range(N):
                for m in range(N):
                        K[n,m]=Abs(x[n],x[m])

        lam=10**(-3)
        I=np.identity(N)
        a=dot(inv(K+lam*I),t)

        #求まったパラメータからモデル関数を作り
        def model_f(z):
                k=[Abs(x[n],z) for n in range(N)]
                return dot(k,a)
```

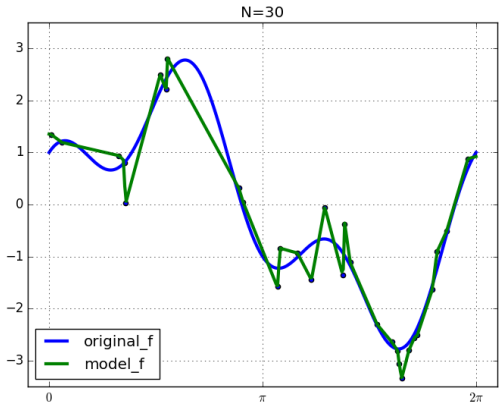
4 結果

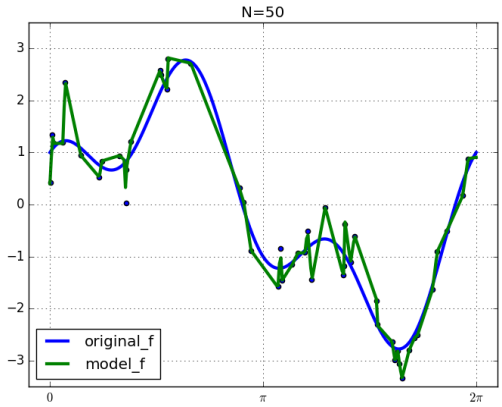
4.1 差の絶対値

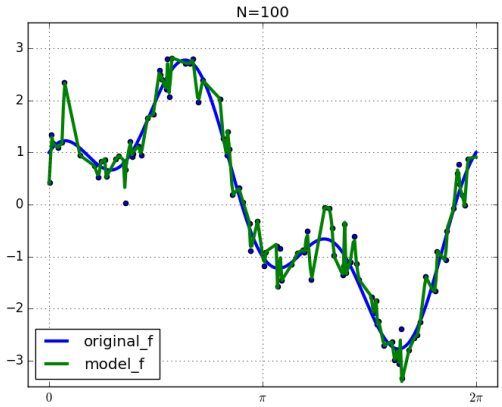
カーネル関数として差の絶対値を選んだ.

$$k(x,z) = |x - z|$$









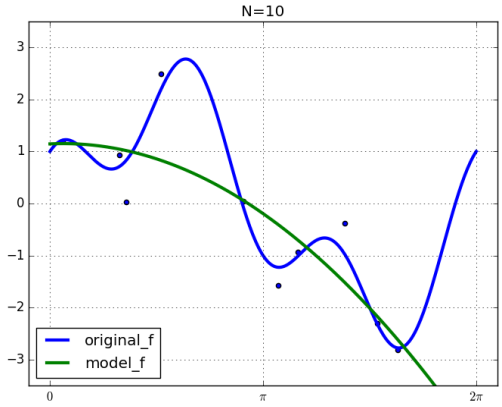
N	10	30	50	100
E	1.19	0.49	0.41	0.40

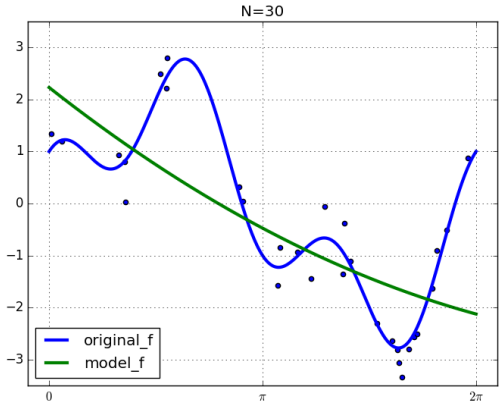
表 1: N と E の関係

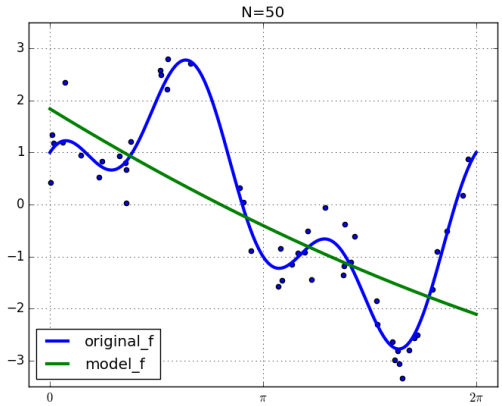
4.2 差の絶対値の二乗

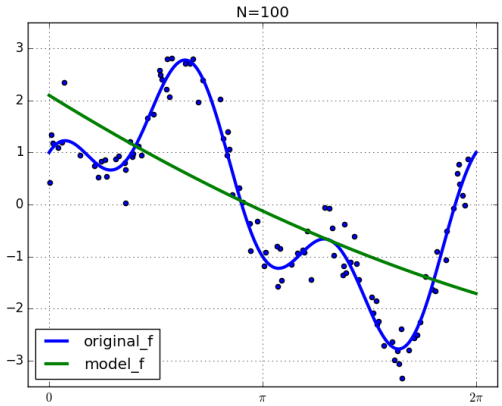
カーネル関数として差の絶対値の二乗を選んだ.

$$k(x, z) = |x - z|^2$$









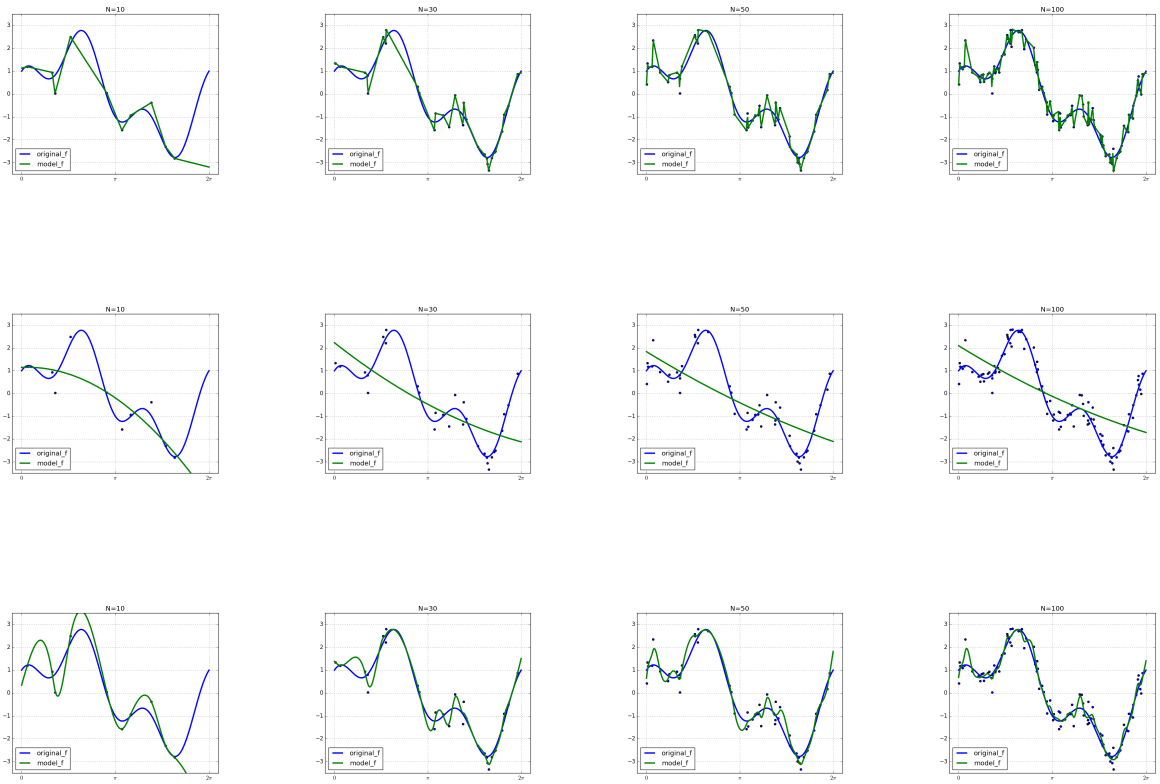
N	10	30	50	100
E	1.70	1.37	1.35	1.27

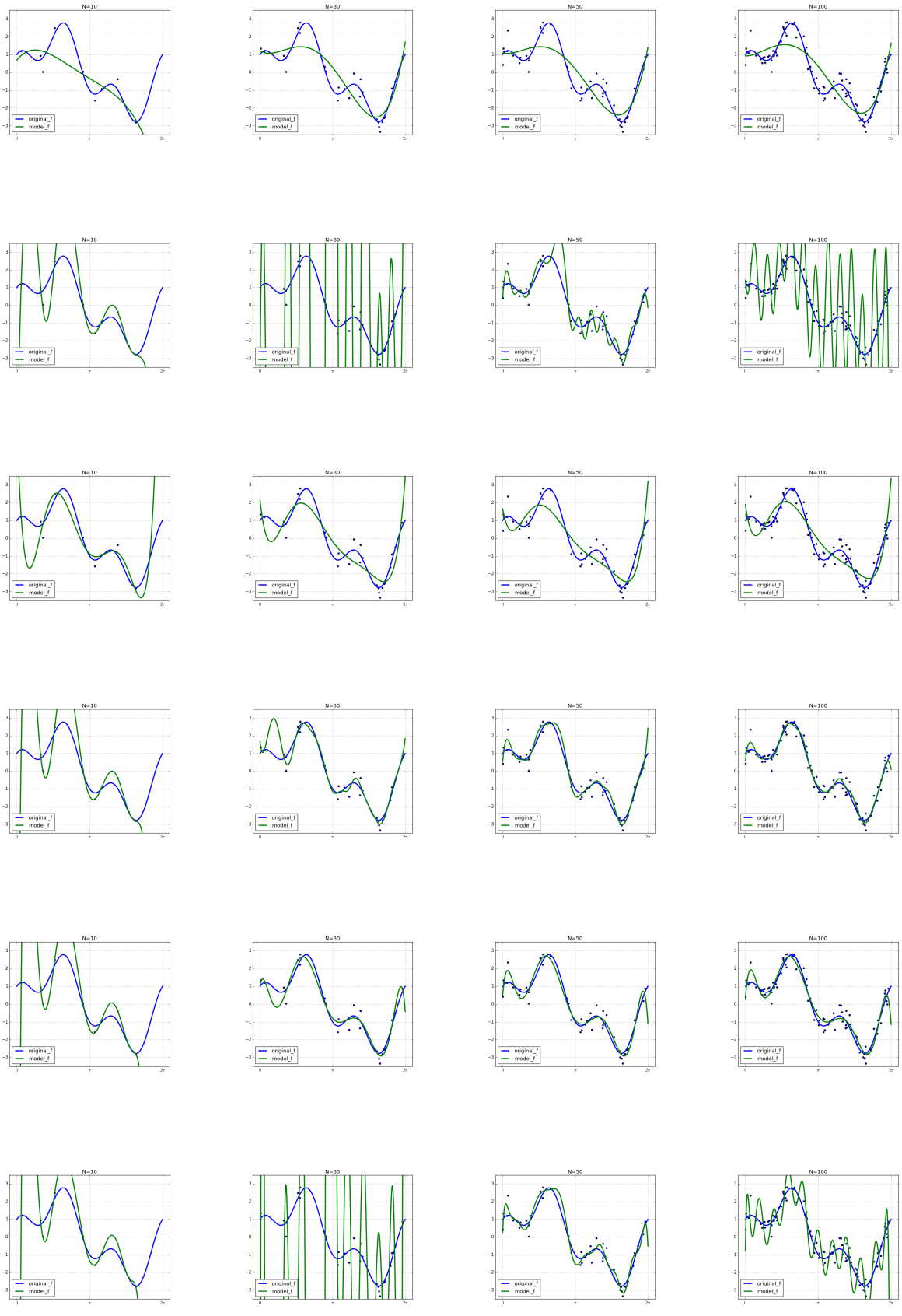
表 2: N と E の関係

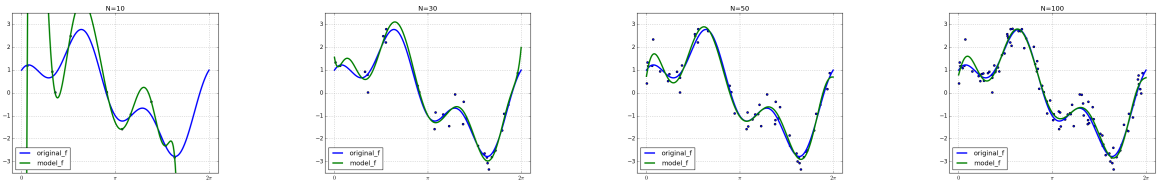
4.3 差の絶対値の累乗

カーネル関数として差の絶対値の累乗を選んだ.

$$k(x,z) = |x-z|^q$$




N=10
N=30
N=50
N=100
original_f
model_f



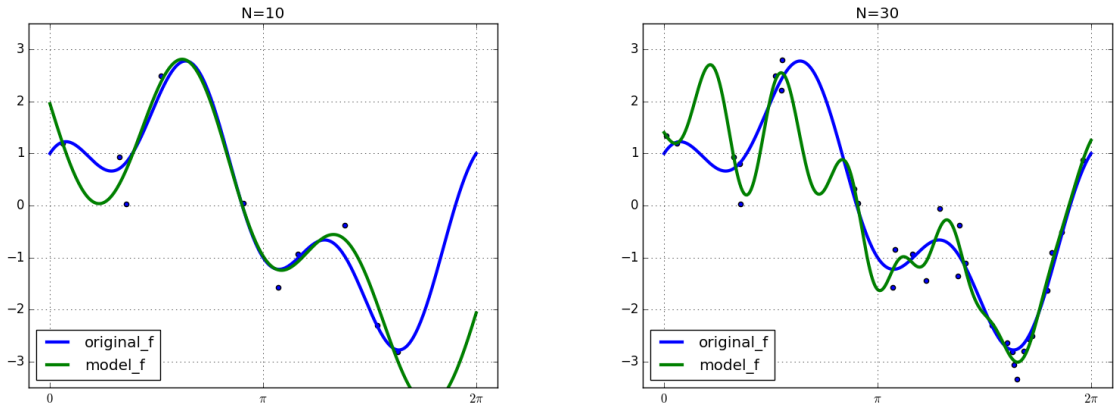
q \ N	10	30	50	100
1	1.19	0.49	0.41	0.40
2	1.70	1.37	2.35	1.27
3	1.82	0.46	0.43	0.33
4	2.96	2.95	2.87	2.87
5	4.87	35.21	0.69	3.79
6	3.42	0.78	0.76	0.72
7	19.06	0.74	0.38	0.31
8	51.99	0.52	0.45	0.42
9	131.52	63.03	0.39	2.22
10	390.07	0.35	0.32	0.30

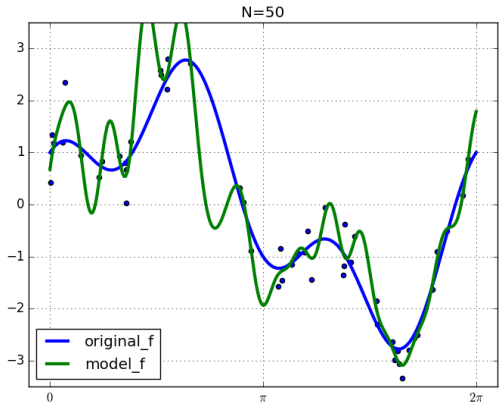
表 3: N と E,q の関係

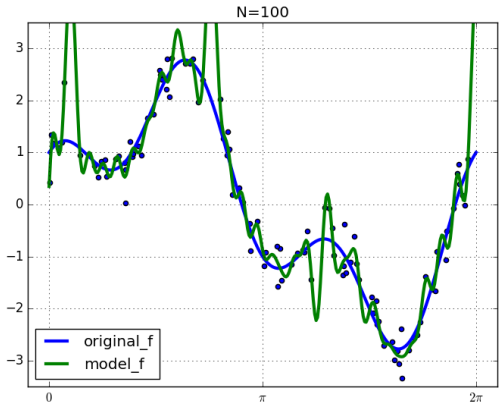
4.4 ガウスカーネル

カーネル関数としてガウスカーネルを選んだ.

$$k(x,z) = exp(-(x-z)^2/\sigma^2)$$







N	10	30	50	100
E	1.13	1.04	0.99	0.83

表 4: N と E の関係

5　まとめ

カーネルを用いるということはデータへ大きく依存するということで, 過学習しやすいのではないかと思う. データ点の少ないときはなかなかいい感じではあった. また, カーネル関数を制御するパラメータの最適化も必要であると考える.