カーネル法

はじめに

カーネル法とは主にパターン認識で使われる手法である．DL(深層学習)がもてはやされる以前には盛んに研究がなされていた．筆者が今現在もそうであるように「カーネル法は難解」とするエンジニアが大勢ではなかろうか？ 本稿では，筆者に理解が及ぶ程度の簡単なカーネル法を説明する．

簡単な線形識別器(線形回帰)

カーネル法以前に簡単な線形識別器について述べる．最も簡単な線形識別器とは(1)である．

$$f = W^{\top}X \quad (1)$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \quad (2)$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} \quad (3)$$

p: 次数(パラメータ数 or 変数の数)

入力Xと重みW(2)の内積により評価値fを得る．ここで，$X(3)$はp次元のデータであるが，XをFと併せてnケ取得したとする(4)．

$$X_{np}W = F \quad (4)$$

$$X_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix}$$

$$X_{np} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} {X_1}^{\top} \\ {X_2}^{\top} \\ \vdots \\ {X_n}^{\top} \end{bmatrix}$$

$$F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

n: データの個数

i: i番目に取得したデータ

(4)を満たす重みWを特定できれば，入力に対する評価値を決定できる．機械学習の過程に突き合わせてみる．

①学習データ(X, F) をnケ取得

②Wを特定(学習完了)

③任意の入力Xに対して評価値fを算出(機能の運用)

Wを算出する②だが，(4)へ最小 2 乗法を適用して(5)とする．

$$X_{np}{}^{\top} X_{np} W = X_{np}{}^{\top} F \quad (5)$$

$$n \geq p$$

簡単な例として(1)を 1 次式(6)としてみる．

$$f = wx \quad (6)$$

パターン認識では任意の入力点(x, f)が 2 つの領域A, Bのいずれにあるか，ということがある(**図 1**)．

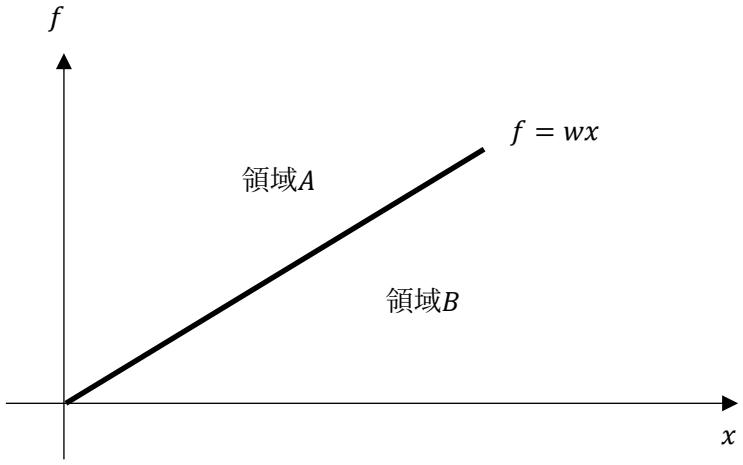

図 1. 領域分割

重みwにより直線が特定されれば，領域の評価が可能となる．なお，1 次式にオフセット(定数)が必要であれば，Xを 2 次元とし，$x_1 = 1$とすれば(7)，(8)となる．

$$X = \begin{bmatrix} 1 \\ x_2 \end{bmatrix} \quad (7)$$

$$f = w_1 + w_2 x \quad (8)$$

(1)からも明らかなように, 1 次元につき一本の直線(1 つの重み)が対応するのみである. 線形識別機では直線(平面)による粗い分割となる.

カーネル法(カーネル回帰)

本稿では，カーネル法の詳細は専門書に譲るとして，カーブフィッティングを題材としようと思う．カーブフィッティングでは近似曲線(理論式など)の数式があり，カーブの形状が決まっている．一方，カーネル法ではこのようなひな形は存在しない．あえて述べるのであればカーネル関数というものがある．本稿では一般に使われるカーネル関数であるガウシアンカーネル(9)を用いる．

$$k(V_1, V_2) = \exp\left(-\frac{\|V_1 -, V_2\|^2}{2\sigma^2}\right) \quad (9)$$

次にW(3)に相当するAを(10)で表記する．次数が学習データの個数であることに注意する．

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad (10)$$

X(1)の次数pとして，$n \geq p$である．ここで(1)に相当するのが(11)である．

$$f = A^{\mathsf{T}} K(X) = \sum_{i=1}^{n} a_i k(X_i, X) \quad (11)$$

$$K(V) = \begin{bmatrix} k(X_1, V) \\ k(X_2, V) \\ \vdots \\ k(X_n, V) \end{bmatrix}$$

Xの代わりに$K(X)$となっているが，実のところAは(12)に由来する．

$$W = {X_{np}}^{\mathsf{T}} A \quad (12)$$

つまり，AはWの次元を$p \to n$と拡張したものである．カーネル回帰ではデータの個数nがベースになり，線形回帰とは異なる．最後に(5)に相当するのが(13)である．

$$K^{\mathsf{T}} K A = K^{\mathsf{T}} F \quad (13)$$

$$K = \begin{bmatrix} k(X_1, X_1) & k(X_1, X_2) & \cdots & k(X_1, X_n) \\ k(X_2, X_1) & k(X_2, X_2) & \cdots & k(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(X_n, X_1) & k(X_n, X_2) & \cdots & k(X_n, X_n) \end{bmatrix} \quad (14)$$

K: グラム行列

ところで，ガウシアンカーネルによるK(14)は対称行列となるため最小２乗法は不要である(15)．

$$KA = F \quad (15)$$

カーネル関数は対称かつ半正定値なる要件があり(15)でよいが，因果律がある系では非対称カーネル関数も議論される．この場合，(15)は適さないだろう．

今一度，(11)だが，そのイメージを示す(**図２**)..

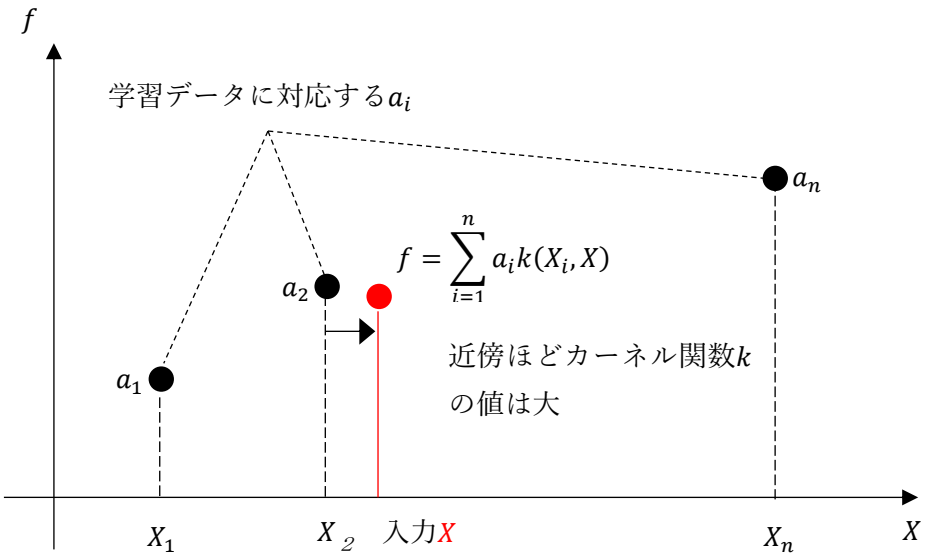

図２　カーネル関数の重ね合わせ

ガウシアンカーネル(9)から明らかなように，カーネル関数は，入力と学習データの位置が近いほど，その関数値が大きくなる．線形回帰に当てはめると，a_iは重みというよりf_iとした方が解かりやすい．カーネル関数にはいくつかの効用があるが，筆者には荷が勝ちすぎる．ここでは列挙するにとどめる．

＊非線形への対応

＊次元拡張

＊非対称カーネルの応用

過学習の抑制

過学習を抑制する方法がある(16).

$$(K + \lambda I)A = F \quad (16)$$

λにより調整することができる. 後述のシミュレーションで確認する.

カーネル法のシミュレーション

カーブフィッティング(近似曲線)のシミュレーションを実施する.

条件:

ガウシアンカーネル: σ=0.3

(15)の求解: ガウスの消去法

学習データ

Xを 1 次元として学習データを用意する(**表 1**, **図 3**).

x	f
0.15	0.1
0.25	0.05
0.3	0.07
0.38	0.2
0.45	0.5
0.55	0.9
0.75	0.87
0.9	0.4
1	0.3
0.15	0.33

表 1　学習データ

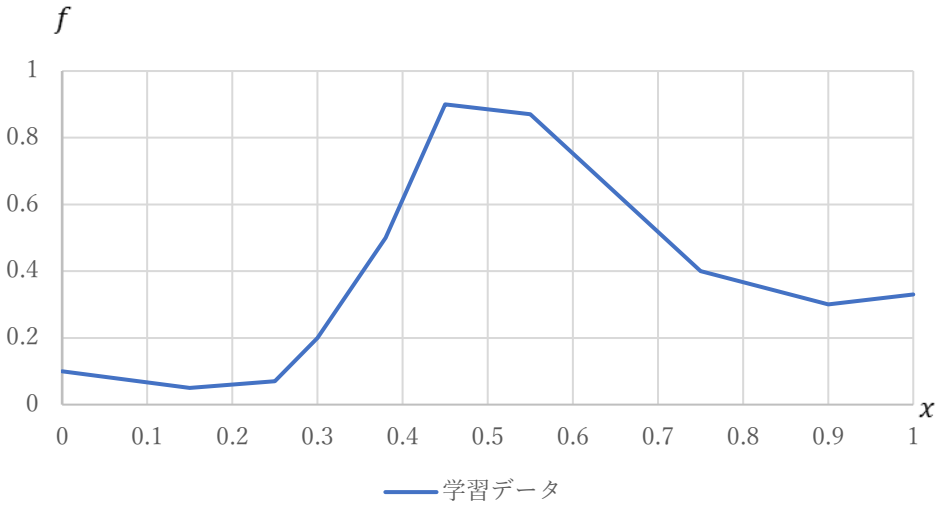

図 3　学習データ

シミュレーション結果

シミュレーション結果を示す(**表 2**, **図 4**).

x	f	f: カーネル法(11)
0.15	0.1	0.1
0.25	0.05	0.05
0.3	0.07	0.07
0.38	0.2	0.2
0.45	0.5	0.5
0.55	0.9	0.9
0.75	0.87	0.87
0.9	0.4	0.4
1	0.3	0.3
0.15	0.33	0.33

表 2　カーネル法(11)の結果

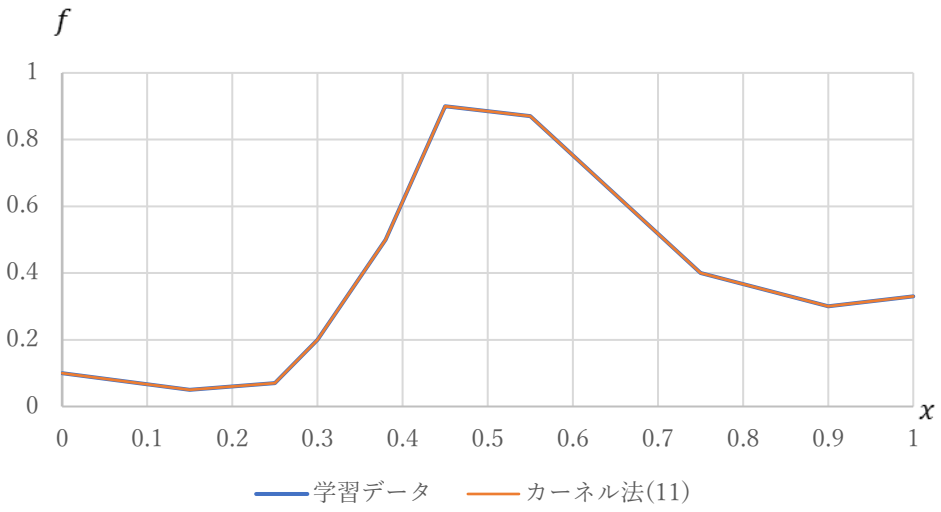

図 4　カーネル法(11)の結果

学習データと一致する．これは学習データと同じxで評価したからである．そこでxを 0.01 刻み($\Delta x = 0.01$)でプロットした結果を示す(**図 5**).

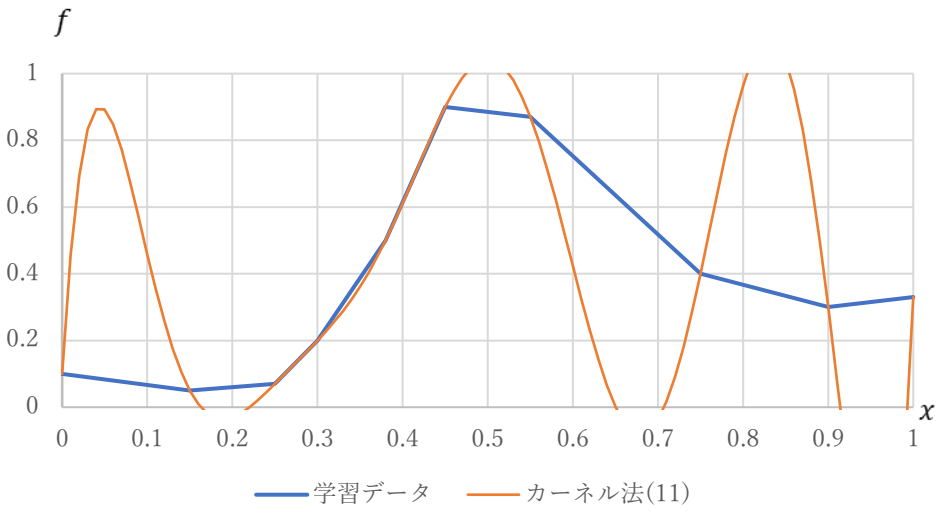

図 5　カーネル法(11)の結果($\Delta x = 0.01$)

学習データと同じxでは一致するがその他の部分では学習データのカーブとはかけ離れている．これが過学習と呼ばれる状態である．この例のように，全ての学習データを一致させるには，極端な曲線でしかフィッティングできないのである．

過学習の抑制のシミュレーション

過学習を抑制するには(16)を用いる．このシミュレーション結果を示す(**図 6**)．

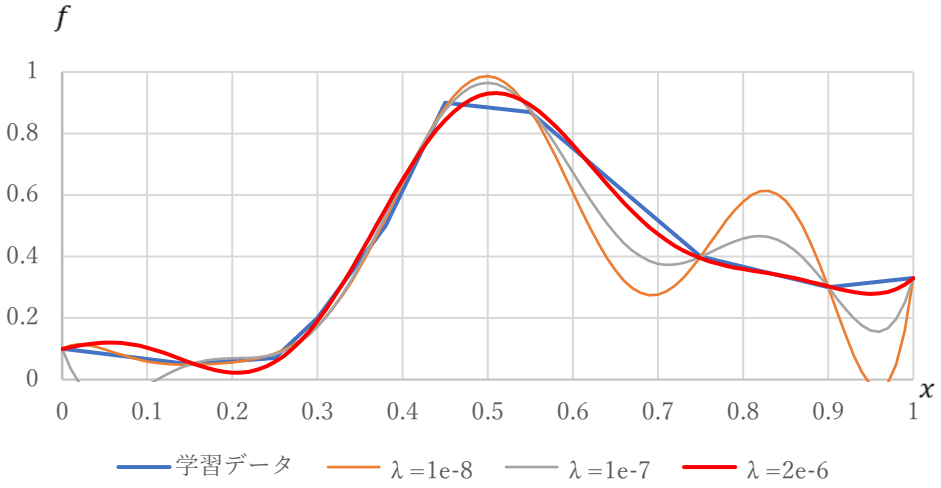

図 6　過学習の抑制($\Delta x = 0.01$)

λの強度を上げることで過学習が抑制されることが確認できる．さらにカーネル関数の重ね合わせ(11)について，個々のカーネル関数を示す(**図 7**)．これは**図 6** の赤線，λ=2.0e-6 のデ

ータである．

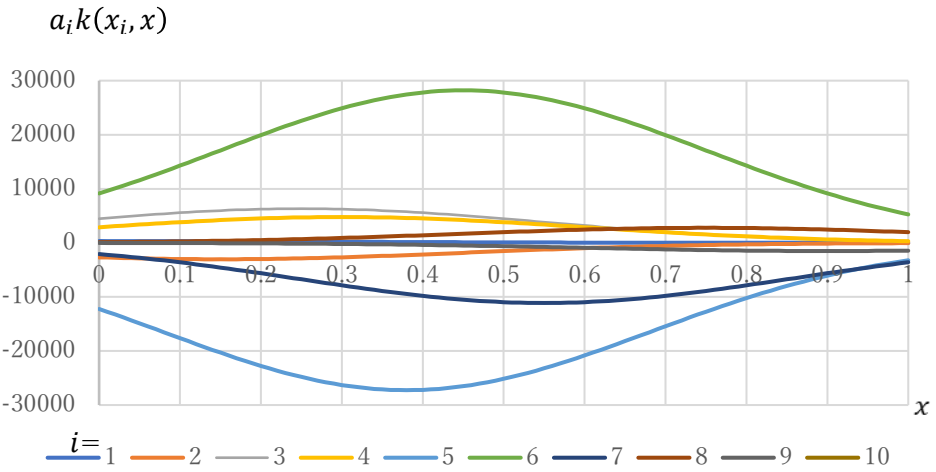

図 7　カーネル関数の重ね合わせ($\Delta x = 0.01, \lambda = 2.0e-6$)

その他のカーネル関数のシミュレーション

カーネル関数として下記(17)～(19)を試行する．

＊ラプラシアン

$$k(V_1, V_2) = \exp\left(-\frac{\|V_1 -, V_2\|}{\sigma}\right) \quad (17)$$

＊ローレンツ関数

$$k(V_1, V_2) = \frac{\sigma}{\|V_1 -, V_2\|^2 + \sigma^2} \quad (18)$$

＊sinc 関数

$$k(V_1, V_2) = \frac{\sin\left(\frac{\pi}{\sigma}\|V_1 -, V_2\|\right)}{\frac{\pi}{\sigma}\|V_1 -, V_2\|} \quad (19)$$

sinc 関数(19)は関数値が零(ゼロ)となる位置を σ として定義した．なお，ローレンツ関数と sinc 関数は見かけないカーネル関数である．これらが対称かつ半正定値という要件を満たすかどうかである．シミュレーションによる検証では，sinc 関数のグラム行列Kは要件-半正定値を満たしていない．sinc 関数の形状から想定されるとはいえ，要件に関わらず機能することもある．ここでは筆者の興味ということで容赦いただきたい．3 つのカーネル関数の結果を示す(**図 8～図 10**)．

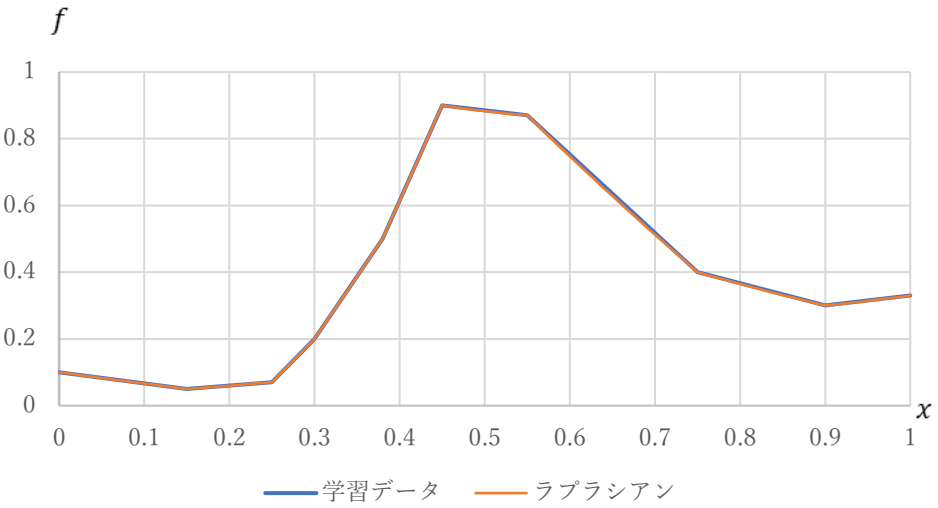

図 8　ラプラシアン($\Delta x = 0.01, \sigma = 0.7, \lambda = 0$)

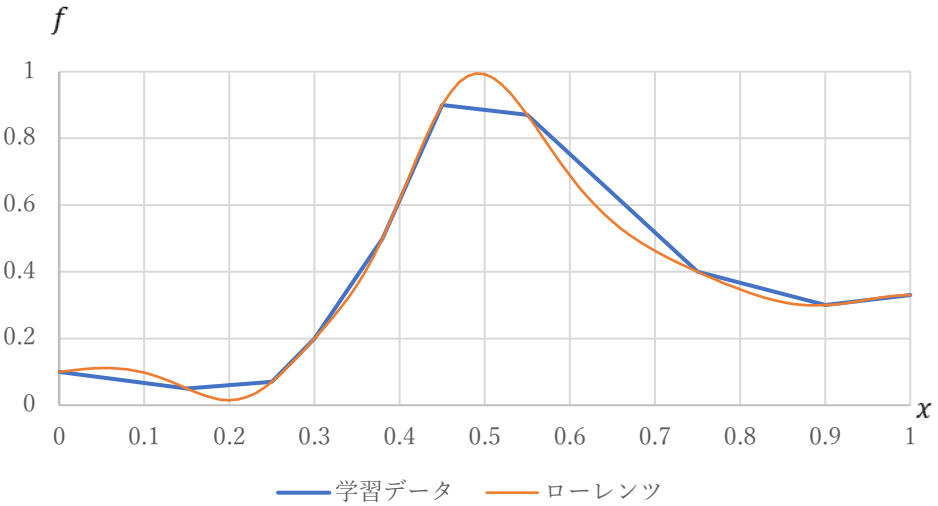

図 9　ローレンツ関数($\Delta x = 0.01, \sigma = 0.2, \lambda = 2e-6$)

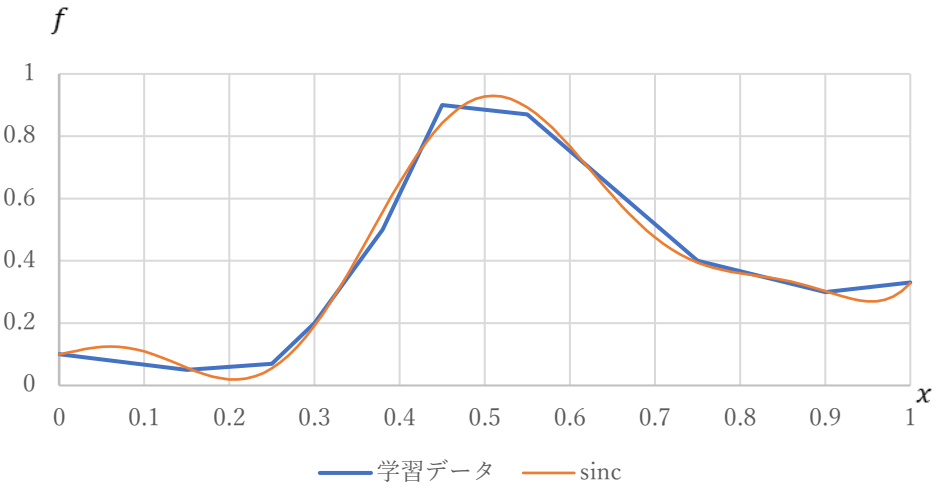

図 10　sinc 関数($\Delta x = 0.01, \sigma = 0.27, \lambda = 2e-6$)

ラプラシアンは線形補間のような結果である．ローレンツ関数はガウシアンと似た形状である．やや下方に凹みがみられるカーブとなっている．さらに，カーネル関数の要件を満たさない sinc 関数も本シミュレーションでは機能する様子である．sinc 関数は標本化定理に由来するものである．sinc 関数の形状からして，カーブの微細な表現が可能(分解能向上)ということもあるかもしれない．

コメント

ここまで，カーネル法の使い方に重点を置いた記述とした．本稿のシミュレーション程度ではスプライン補間を使えば十分だが，学習には丁度よい題材である．過学習の抑制を適用しない場合，スプライン補間と同様，学習データで指定される点を通る補間となる(**図 5** 参照)．

過学習については多項式フィッティングと似た部分があるように思う．多項式フィッティングではデータ数の増大と共に次数を上げる必要がある．**図 5** は多項式フィッティングでもみられる事象である．また，多項式フィッティングで粗く近似するのなら次数を落とせばよく，これに相当するのが過学習の抑制という感触である．

筆者の場合，本稿シミュレーションのように入力データを常に 0〜1 で正規化するのが常である．これにより σ や λ の数値の変動を，ある程度は避けることができる．また，PCA(主成分分析)を扱うような場合，確率を 0〜1 で表現するということがある．

カーネル法では学習データの規模の増大により，計算コスト（実行時間）に難があるように思う．他に，規模の増大によりガウスの消去法も実用とならない．直接法による求解は手順の進行とともに桁落ち誤差が問題となる．やはり，行列全体でバランスをとることができる反復法，共役勾配法が適するのかと思う．