論　説

多次元補間多項式関数を用いた識別関数の一構成法

吉　田　　要

目　　次

Ⅰ．はじめに

与えられた特徴データを既知のクラスに分類することは，画像処理のみならず多くの分野で広く研究が進められている。適当な処理を施すことで，判断しにくいデータから，意味のある知識を抽出することが可能となることがある。このような問題は一般にパターン識別問題という。なかでも，2 クラスへの識別は，多くの場面で必要となることがある。例えば，SVM（Support Vector Machine）［1］を用いて，企業の半年後の倒産可能性の予測［9］は，当該企業への投資に対する重要な判断因子となる。また，技術開発は多大の投資を必要とすることから，開発の実施や続行を決定するために，過去のデータから何らかの意味のある情報を抽出して判断の根拠とすることは重要で，各種の方法が提案されている［5, 6, 7, 8］。筆者らも，技術評価支援システムの開発に向けて，幾何計画法［2, 13, 14］を利用した識別関数の構成［11］や，識別関数のオーバーフィッティンッグ問題の解消［12］について提案をした。

識別関数の能力を測るには，蓄積データのうち，学習用データとして採用しなかったデータに対して得られた識別関数を用いて，クラス判定をすることで評価されることが一般的である。この検証データに対する識別結果を評価して，将来得られる未知のデータに対しても，同様な結果を得るであろうと推測されることが多い。しかし，重要な意思決定をする場合は，識別関

数がデータに対してどの程度影響を受けるかを，前もって調べておくことは重要である。識別関数がデータによって変化する度合いが無視できない場合は，識別関数に対して十分な期待をおくことができない。

このような問題を解決するには，蓄積したデータから，学習データを変化させて，各ケースで得られる識別関数がどのようになるかを観察することが重要である。

また，データが時系列的に与えられるとき，蓄積データは過去のデータであり，時間の影響を考慮しなくてよい場合は問題は生じないが，企業のデフォルト予測において，時間的変動を考慮することでよい結果が得られたとの報告［4］など，従来の特性とは異なったデータが現れる可能性がある場合は，新たに発生するデータを識別する関数は時間変化の影響を考慮する必要がある。本論文では，識別関数と学習データとの影響について簡単な事例を用いて，データセットの選択による影響について考察をし，データ依存の強い識別関数を避けることで，識別関数の頑健性を向上させることを試みる。

Ⅱ．記号の定義

便宜上，以下の記号を定めておく

A と B の直積集合

$A \times B = \{(a, b) \mid a \in A, b \in B\}$

$R^m = R \times .. R$　　R の m 個の直積 m 次元空間

$\boldsymbol{x} = (x_1, \ldots, x_n)^t$: n 次元ベクトル　$\boldsymbol{x} \in R^n$，t : 転置

クロネッカーの δ

$$\delta_{ij} = \begin{cases} i = j \Rightarrow 1 \\ i \neq j \Rightarrow 0 \end{cases}$$

Lagrange 関数

1 次元の場合：

$l_i(x_j) = \delta_{ij}$

多次元の場合：

$L_i(\boldsymbol{x})$: 多次元 Lagrange 関数

$||\cdot||$ ノルム　: $R^n \to R_+$ 正の実数の集合

$\boldsymbol{x}^A = (\boldsymbol{x}^{\boldsymbol{a}_1}, \ldots, \boldsymbol{x}^{\boldsymbol{a}_m})^t$

ただし，

$A = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m)^t$

$\boldsymbol{a}_i = (a_{i1}, .., a_{in})^t \in R^n$

$$\boldsymbol{x}^{\boldsymbol{a}_j} = \prod_{i=1}^{n} x_i^{a_{ji}} = x_1^{a_{j1}} x_2^{a_{j2}} .. x_n^{a_{jn}}$$

多項式関数

$$f(\boldsymbol{x}) = \sum_{j=1}^{m} c_j \boldsymbol{x}^{\boldsymbol{a}_j} = \langle \boldsymbol{c}, \boldsymbol{x}^A \rangle$$　　ただし〈　，　〉は内積

また，$\boldsymbol{a}_i$ が整数の要素からなるとき

$\boldsymbol{a}_i \in I^n = I \times I .. \times I = \{(m_1, .., m_n) \mid m_i \in I, i = 1, .., n\}$

$f(\boldsymbol{x})$ を整多項式という。ただし，I は整数の集合

Ⅲ．識別関数と補間法

識別関数の構成は種々な方法があるが，2 クラスを分類する為のパターン識別ということができ，分布されたデータが超平面で分離できる場合，線形識別可能という。

特徴空間の次元が n のとき，超平面は空間を 2 分割することが出来，一方の半空間に属している集合をクラス A，他方をクラス B として扱うことが出来る。（図 1 参照）

ここで，H は超平面であり，$n-1$ 次元の線形多様体である。すなわち，

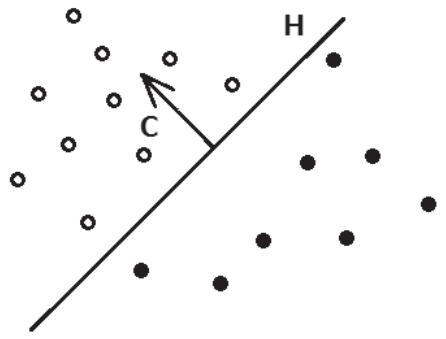

図 1　分離超平面

$$H = \{\boldsymbol{x} \mid \langle \boldsymbol{c}, \boldsymbol{x} \rangle = d, d \in R, \boldsymbol{c}, \boldsymbol{x} \in R^n\}$$

この内積が正であれば，ベクトル $\boldsymbol{x}$ は，ベクトル $\boldsymbol{c}$ の矢印側に存在しているので，H によって，内積の正負の問題に帰着できるとき，線形分離可能という。

このようなベクトル $\boldsymbol{c}$ と d を見つけることが線形識別問題であり，H を記述する式が識別関数である。

1. 最適化による識別関数の構成

サポートベクターマシン（Support Vector Machine）は，お互いの識別領域を分離する領域（マージン）を最大化するもので，識別能力を高めることが期待できる。（図 2 参照）　図で，H1，H2 は超平面であるが，この平行な平面間の距離をマージンという。そして，その中間が分離超平面であり，識別関数に対応する。超平面を決定するベクトル **a**，**b**，**c** をサポートベクター（支持ベクター）という。SVM は線形分離できないようなデータセットに対して，より高次元の特徴空間に写像して線形分離を行い逆写像によって非線形識別関数を得る方法であるが，高次元化する空間の次元を考慮しない場合には，サポートベクターは，学習データの全てあるいは大

多数をサポートベクターとすることが起こりやすく，対象データセットへの依存度の高い識別関数となり，いわゆるオーバーフィッティングを起こすことになる。オーバーフィッティングを起こした識別関数は，将来得られる未知のデータに対して，識別能力が低下する恐れが高くなり，頑健性が損なわれることが懸念される。このことを避ける為には，高次元化する場合でも，次元数を極力抑えた識別関数を構成する方法の開発が重要となる。最適化法はマージンを最大化できる利点と同時にオーバーフィッティングを起こすかどうか事前に判定できないという欠点を持つ。このため，支持ベクトルの個数をあらかじめ固定して識別関数を構成できれば，オーバーフィッティングを避けるうえでは有効な方法と考えられる。

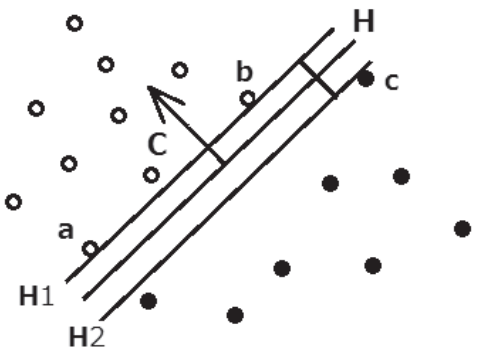

図 2　超平面とマージン

2. 多次元補間関数による識別関数の構成

補間法は離散データを連続な関数で近似するもので，与えられた点でのデータがない場合でも，近似関数から値を得る方法である。特に，1 次元のデータに対してはデータの特徴などから各種の方法が開発されている。ここでは，補間法をパターン識別に適用することを検討する。

(1) Lagrange 補間 [10]

代表的で，比較的扱いやすい Lagrange 法を検討する。

① 1 次元の Lagrange 補間

n 次の Lagrange 補間関数による近似関数は $f(x)$ は次のようになる。

$$f(x)=\sum_{i=1}^{n} f_i \cdot l_i(x)$$

この $l_i(x)$ を Lagrange 補間関数といい，次のようになっている。

$$l_i(x)=\frac{\prod_{j\neq i}^{m}(x-x_j)}{\prod_{j\neq i}^{m}(x_i-x_j)}$$

離散点での関数値は，その点でのデータと一致し，

$$f(x_i)=f_i \qquad i=1,..,m$$

となり，離散点でのデータが得られている。

例１　１次元 Lagrange 補間関数

次の，４個のデータに関して近似した関数は図３のようになる．

x	1	2	3	4
f	4	2	3	5

表１　４点の Lagrange 補間

このとき，Lagrange 関数　l_1，l_2，l_3，l_4 はそれぞれ次のようになる。

$$l_1(x)=\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)}$$

$$l_2(x)=\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)}$$

$$l_3(x)=\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)}$$

$$l_4(x)=\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}$$

図４から図７に見るとおり，隣り合った点での関数は，離散点以外では相殺するようになっている。このことから，離散点間は区間幅にもよるが比較的滑らかに近似できることになる。

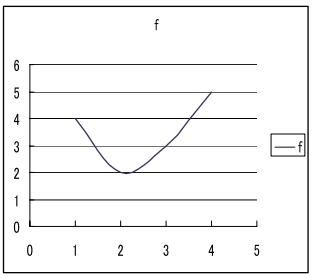

図３　近似関数によるグラフ

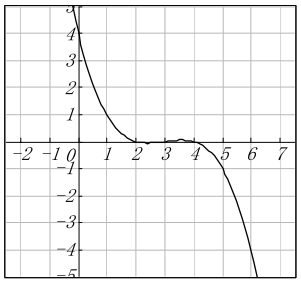

図４　$l_1(x)$ のグラフ

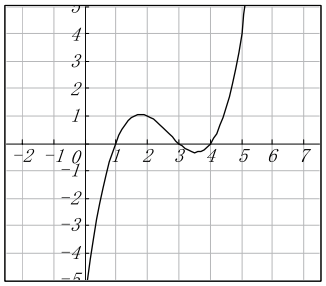

図５　$l_2(x)$ のグラフ

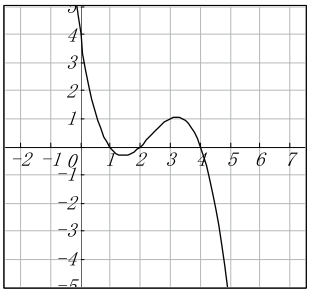

図６　$l_3(x)$ のグラフ

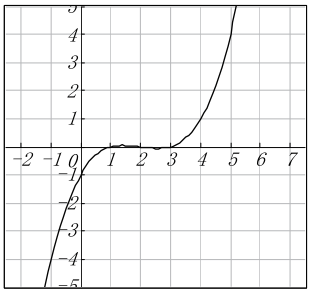

図７　$l_4(x)$ のグラフ

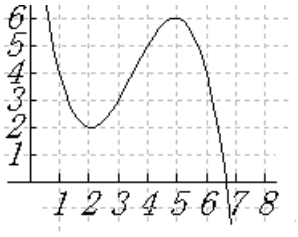

図８　l_1, l_2, l_3, l_4の合成グラフ

この補間関数 $f(x)$ は

$$f(x)=\sum_{i=1}^{n} f_i \cdot l_i(x)=4\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)}+2\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)}$$
$$+3\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)}+5\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}$$

となり，この関数は図示すると，図 4 から図 7 が $f(x)$ の第 1 項から第 4 項である。これらを合成した関数の合成図は図 8 である。区間 1 から 4 までの補間区間は，図 3 と比較すると，よい近似になっているのが見て取れる。

もし，離散点での値がクラスによって正と負に分かれるのであれば，$f(x) = 0$ を識別関数とすることができる。ところで，1 次元のラグランジュ関数は，隣り合った関数同士は，離散点と異なるところでは，相殺しあう形になっており，近似関数が比較的滑らかになるが，多次元に於いては，隣接する離散点は 2 点ではないので，この関数を多次元に拡張するときに留意すべき点となる。

例 2　1 次元 Lagrange 補間関数を利用した識別例

次のように，7 個のデータのうち，1，2，5，6 が A クラス，1，4，7 が B クラスとなっているとき，関数値をそれぞれ 1 と −1 とすれば，6 次の近似関数はつぎのようになるので，$f(x) = 0$ が識別関数となる。

x	1	2	3	4	5	6	7
f	1	1	−1	−1	1	1	−1

表 2　A, B クラスの例

図 9 では，グラフが X 軸と交わるところがクラスの境界と考えることが出来る。X 軸より上が A，下が B クラスと判定することが出来る。

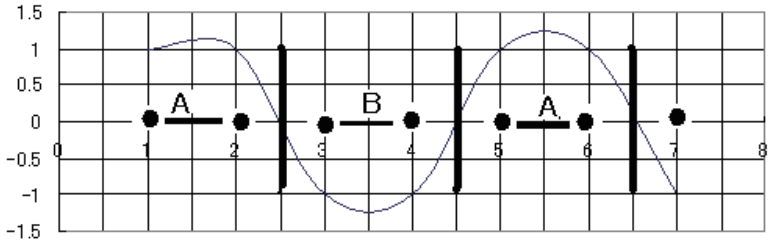

図 9　近似関数とクラスの識別

②多次元への拡張

Lagrange 補間関数の多次元への拡張を考える。多次元の場合に拡張するためには，ベクトルをスカラ化する必要があり，適当な関数により写像することを考える。

ベクトル間の距離を表すノルムを導入しスカラ化した次の L 関数を構成する。

$$L_i(\boldsymbol{x}) = \frac{\prod_{j \neq i}^{n} ||\boldsymbol{x} - \boldsymbol{x}_j||}{\prod_{j \neq i}^{m} ||\boldsymbol{x}_i - \boldsymbol{x}_j||}$$

各ベクトル $\boldsymbol{x}_i$ に対して，1 変数の Lagrange 関数と同様に次の関係が成立する。

$$L_i(\boldsymbol{x}_j) = \delta_{ij}$$

このことから，多変数の場合の補間関数 $f(\boldsymbol{x})$ は

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} f_i L_i(\boldsymbol{x})$$

と表すことができ

$$f(\boldsymbol{x}_j) = f_j \qquad j = 1, .., m$$

となる。以上のことから，

$f(\boldsymbol{x}) = 0$ を識別関数と考えて，正負の判定によってクラスを推定することが可能となる。1次元の Lagrange 関数と異なり，L 関数は常に非負なので，離散点以外では相殺されない。また，1次元と異なり，隣の離散点が高々 2 個ということにもならない。このことは識別関数を構成する場合は考慮しておくべき点である。以上を考慮して，検証データを使った結果を利用して，識別関数を改良することで識別能力の向上をはかる方法について提案する。

ノルムとして，ユークリッドノルムと最大ノルムついて構成してみる。各ノルムの近傍は，ユークリッドノルムが超球，最大ノルムは超立方体になる。

これらの使い分けは検証データに対する識別成績の優れた結果を出した方を選択する。

$$||\boldsymbol{x}-\boldsymbol{y}||_E = \{\sum_{\mathrm{i}=1}^{\mathrm{n}} (x_i-y_i)^2\}^{\frac{1}{2}}$$

$$||\boldsymbol{x}-\boldsymbol{y}||_\infty = \max_{\mathrm{i}} |x_i-y_i|$$

例 3　3 点での識別例

$\boldsymbol{x}_1 = (1, 3, 2)^t$，$\boldsymbol{x}_2 = (5, 7, 1)^t$，$\boldsymbol{x}_3 = (2, 3, 5)^t$

に対して，それぞれ A, B, A クラスとする。また，$\boldsymbol{x} = (x, y, z)$ とする。

$$L_{1E1}(\boldsymbol{x}) = \frac{||\boldsymbol{x}-\boldsymbol{x}_2||_E \cdot ||\boldsymbol{x}-\boldsymbol{x}_3||_E}{||\boldsymbol{x}_1-\boldsymbol{x}_2||_E \cdot ||\boldsymbol{x}_1-\boldsymbol{x}_3||_E} = \frac{\{(\mathrm{x}-5)^2+(\mathrm{y}-7)^2+(\mathrm{z}-1)^2\}^{\frac{1}{2}} \cdot \{(\mathrm{x}-2)^2+(\mathrm{y}-3)^2+(\mathrm{z}-5)^2\}^{\frac{1}{2}}}{(4+16+1)^{\frac{1}{2}} \cdot (1+0+9)^{\frac{1}{2}}}$$

$$= \frac{[\{(\mathrm{x}-5)^2+(\mathrm{y}-7)^2+(\mathrm{z}-1)^2\}^{\frac{1}{2}} \cdot \{(\mathrm{x}-2)^2+(\mathrm{y}-3)^2+(\mathrm{z}-5)^2\}]^{\frac{1}{2}}}{5\sqrt{26}}$$

$$L_{\infty 1}(\boldsymbol{x}) = \frac{||\boldsymbol{x}-\boldsymbol{x}_2||_\infty \cdot ||\boldsymbol{x}-\boldsymbol{x}_3||_\infty}{||\boldsymbol{x}_1-\boldsymbol{x}_2||_\infty \cdot ||\boldsymbol{x}_1-\boldsymbol{x}_3||_\infty} = \frac{\max\{|x-5|, |y-7|, |z-1|\} \cdot \max\{|x-2|, |y-3|, |z-5|\}}{4 \cdot 3}$$

$L_{E1}(\boldsymbol{x}_1) = 1$，$L_{E1}(\boldsymbol{x}_2) = L_{E1}(\boldsymbol{x}_3) = 0$

$L_{\infty 1}(\boldsymbol{x}_1) = 1$，$\mathrm{L}_{\infty 1}(\boldsymbol{x}_2) = \mathrm{L}_{\infty 1}(\boldsymbol{x}_3) = 0$

L_{E2}，L_{E3}，$L_{\infty 2}$，$L_{\infty 3}$ についても，同様なので離散点では δ 関数となっているのが分かる。この例では，L_∞では 3 変数の高々 2 次関数である。

$\boldsymbol{x} = (3, 3, 3)$ では，

$L_E(\boldsymbol{x}) = 0.60302 - 0.135932 + 0.54100 = 1.00809$

$L_\infty(\boldsymbol{x}) = 2/3 - 1/3 + 2/3 = 1$

でともに A クラスと判定される。

例 4　iris データの多次元 Lagrange 関数による識別（付録 1 参照）

iris データに対して，最大ノルムを用いた拡張 Lagrange 法で識別関数を求める。

学習データは 10 個で，バージニカとバーシクルから，それぞれ 5 個ずつ選んでいる。検証データは学習データを含め 98 個である。case1, case2 のヒット率はそれぞれ，49, 92 となっている。

	1	2	3	4	5	6	7	8	9	10	ヒット数
case1	1	1	1	1	1	−1	−1	−1	−1	−1	49
case2	1	1	1	1	1	−12	−12	−12	−12	−12	92

表 3　多次元 Lagrange 関数による識別

case1 では A クラスの関数値を 1，B クラスの関数値を −1 として求めた。ついで，この識別関数の関数値を 1 と −12 とした場合を case2 とした。case2 では識別率が大きく上がっているのが判る。識別関数の構造は同じなので，若干の変更によって，改良できたことになる。付録 1 の表の第 1，10 列はクラスの真の識別値。7，9 列は case1 case2 の予測値である。

2 ～ 4，11 ～ 14 まではデータ，6，8，14，16 は識別関数の出力で，7，9，15，17 は判定クラスが正しいときは 1 を誤判定のときは 0 としている。

ところで，識別関数の頑健性を維持するためには学習データを少なくし，よい識別能力を有する関数を構成できることが望まれる。

しかしながら，学習データを少なくすると，検証データが多くなり，識別能力が低くなるという相矛盾することが生じる。これを解消するためには，後の例で見るように，学習データの選択を変化させて，最も良いケースを元に識別関数の改良を検討することが有効と思われる。

以下は，最適化法を用いて改良する方法である。

クラスを決める識別値は符号判定になるので，

$x_i \in A$　ならば　$f(x_i) > 0$

$x_i \in B$　ならば　$f(x_i) > 0$

となるように，係数ベクトルを決めることにすれば，1，−12 のケースよりもよりよい識別率が期待できる。この問題は，

各離散点での，拡張 Lagrange 関数の x_q における p 番目の項の値 $L_{\infty p}(x_q)$

を要素とする行列

$$Z=\begin{pmatrix} L_{\infty 1}(\boldsymbol{x}_1) & L_{\infty 2}(\boldsymbol{x}_1) & .. & L_{\infty m}(\boldsymbol{x}_1) \\ L_{\infty 1}(\boldsymbol{x}_2) & L_{\infty 2}(\boldsymbol{x}_2) & .. & L_{\infty m}(\boldsymbol{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ L_{\infty 1}(\boldsymbol{x}_s) & L_{\infty 2}(\boldsymbol{x}_s) & .. & L_{\infty m}(\boldsymbol{x}_s) \end{pmatrix}$$

$$Z\cdot\begin{pmatrix}\boldsymbol{f}_1\\ \boldsymbol{f}_2\end{pmatrix}=\begin{pmatrix}\boldsymbol{a}\\ \boldsymbol{b}\end{pmatrix}$$

ここで，f_1，f_2 は学習データのクラスA，Bに対応する関数値であり，
a，b は全データのクラスA，Bに対応するものである。ここで，

$$\min_i a_i > \max_j b_j$$

なるように f が決定できれば良い。

$\max s_1 - s_2$

f, s

sub. to　$Z\cdot f_1 \geqq s_1(1, 1,.., 1)^t$

$Z\cdot f_2 \leqq s_2(1, 1,..1)^t$

ただし，$\boldsymbol{f}=(f^t{}_1, f^t{}_2)^t$, $\boldsymbol{s}=(s_1, s_2)^t$

(2) 多次元多項式補間

n 次元空間を張ることが出来る基底として多次元の多項式を考える。

2つのクラスを判定するのに，ϕ の関数値の正負で判定することを考える。単項式の線形結合で識別関数を構成すれば，係数 $\boldsymbol{c}$ を決定する問題として考えることができる。

$\boldsymbol{x}\in R^n : \phi(\boldsymbol{x})=(\phi_1(\boldsymbol{x}),.., \phi_p(\boldsymbol{x}))^{\mathrm{t}}\in R^p$

すなわち，学習データの属するクラスA，Bに対して，それぞれのクラスに属する識別値をそれぞれ 1，-1 とする。この値は，クラス毎に全ての離散点で変更するのであれば，方程式は線形なので k，$-k$ としても識別値の符号は変化しないので改良につながらないが，クラスごとに変更すれば改良が期待できる。

$\phi(\boldsymbol{x})=\langle \boldsymbol{c}, \phi(\boldsymbol{x})\rangle$

とすることで，識別関数を求める問題は，係数 $\boldsymbol{c}$ を求める問題になる。

すなわち，

$(\phi(\boldsymbol{x}_1),.., \phi(\boldsymbol{x}_p))^t=\boldsymbol{\sigma}=(\sigma_1,.., \sigma_p)^t\in I_1^p$

ただし，

$\sigma_i = 1$ or -1　　$I_1=\{1, -1\}$

で

$$\begin{pmatrix} \phi(\boldsymbol{x}_1)^t \\ \vdots \\ \phi(\boldsymbol{x}_p)^t \end{pmatrix} \cdot \boldsymbol{c} = A \cdot \boldsymbol{c} = \boldsymbol{\sigma}$$

より，A が正則であれば

$\boldsymbol{c} = A^{-1} \cdot \boldsymbol{\sigma}$

となり計算できる。

① 整多項式による識別関数の構成

整数巾を持つ多項式を用いて識別関数を構成する。

入力変数の次元が n で次数が m の場合の異なる項の総数 P_m は次のようになる。

$$P_m = \frac{(n+m-1)!}{(n-1)! \cdot m!}$$

よって，m 次までで異なる項の総数 S_m は

$$S_m = \sum_{k=1}^{m} P_k = \sum_{k=1]}^{m} \frac{(n+k-1)!}{(n-1)! \cdot k!}$$

となる。学習データの個数を，v とすると，必要な多項式の次数 m は

$S_{m-1} < V \leqq S_m$

なる関係を満たせばよい。以上のことから，計算例題のデータに対して，ベクトルの次元数 n は，$n = 4$ なので，3 次の同次式による項 20 個を用いて識別式を作るとすれば，学習データの個数は 20 個となる。また, 3 次以下の全ての項を使う場合は, 学習データの個数は 34 となり, これらで構成した識別関数による結果を示す。

例 5　同次式をもちいた整多項式型の識別関数の構成（付録 2 参照）

2 クラスからそれぞれ 10 個ずつを学習データとして選んでいる。

この同次式の 20 個の単項式は次のようになっている。変数は $\boldsymbol{x} = (z,\ x,\ e,\ v)$ としている。

$\boldsymbol{\Phi}(\boldsymbol{x}) = (z^3, z^2x, z^2c, z^2v, x^2z, x^3, x^2c, x^2v, c^2z, c^2x, c^3, c^2v, v^2z, v^2x, v^2c, v^3)^t$

よって識別関数 Φ は，この係数ベクトルと単項関数より

$\Phi(\boldsymbol{x}) = \langle \boldsymbol{c}, \Phi(\boldsymbol{x}) \rangle$

となる。学習データを含めた検証データは　バーシクル 50　バージニカ 49 の 99 個[3]である。

表 5（付録 2）は整多項式型の識別関数を求めた場合であるが，99 個のデータからなるデータセットに対して 20 個の学習データを複数回選択した場合の識別率を示している。

5 ケースの識別ヒット数は, 51, 73, 62, 62, 85 また, 34 個の学習データでは, 2 ケースを扱っているが, 識別ヒット数は 68 と 75 となっている。学習するデータは 20 個の場合より, カバー

率が高いので，識別能力はあがり安定してくるはずであるが，このケースだけでは其の効果は顕著ではない。仮に全データを学習データとすれば，全データを識別できる。その分，頑健性が落ちると考えられる。この計算事例では，20 個の学習データの第 5 のケースを採用することが望ましい，識別率は 85% 強である。

係数ベクトル c は

c ＝

(0.01507，0.02735，0.015995，−0.04399，−0.06791，0.019202，0.092623，−0.071258，
0.024618,013446，−0.010655，−0.00496，−0.066804，0.066564，0.044086，−0.018717，
−0.124922，0.152668，0.073229，−0.127735)

となっている。(付録 2 を参照)

これに対して，3 次式以下の全ての異なる項から構成した識別関数では，学習データの個数は 34 個で，2 ケースを作成して識別関数を作った結果，68，75 であった。

	z	x	c	v	zz	zx
case1 68	201.2503	−121.578	794.3851	−321.643	−2.87937	−5.98613
case2 75	−44.6521	21.1187	5.058322	−8.06418	−0.54973	−1.02535
zc	zv	xx	xc	xv	cc	cv
10.69422	−4.84354	1.641662	−27.2456	15.09343	−14.5307	6.465909
0.719235	2.20433	0.539165	0.121623	−1.36371	0.211182	−0.60383
vv	zzz	zzx	zzc	zzv	xxz	xxx
−1.54175	−0.04326	0.156559	−0.19689	0.043584	0.01674	−0.02167
0.537871	−0.01539	0.015777	−0.02484	0.020244	−0.02182	0.006533
xxc	xxv	ccz	ccx	ccc	ccv	vvz
0.083103	−0.01745	−0.0727	0.223441	0.059159	−0.00579	0.094517
0.000379	−0.01923	0.035634	−0.00993	−0.00662	0.004115	−0.01877
vvx	vvc	vvv	zxc	zxv	zcv	xcv
−0.10591	−0.05154	0.034855	0.179827	−0.09273	−0.14306	0.054435
0.024196	0.011436	−0.00951	0.039057	0.024646	−0.06144	−0.0028

表 6　3 次以下の次数による識別（上段，下段は case1，case2 の係数ベクトル c）

20 個の場合より，学習データの選択の仕方による影響は少ないと思われる。しかしながら，このケースを含めても，頑健性と識別率の観点から第 5 ケースが優れており，学習データの個数と同時にデータの選択も重要であることがわかる。

②指数可変型多項式による構成

整多項式で識別関数を構成する方法について示したが，指数部が整数でない場合を許せば，より一般的な識別関数を構成することができる。

ここでは，指数部も変数として扱うことを検討する。すなわち，識別関数Φを

$\Phi\ (\boldsymbol{c}, A, \boldsymbol{x}) = \langle\ \boldsymbol{c}, \boldsymbol{x}^{A}\ \rangle$

とする。ここで，

$\boldsymbol{x} \in R^{n}$，$\boldsymbol{c} \in R^{m}$ で，

$$\boldsymbol{x}^{\boldsymbol{a}i} = \prod_{j=1}^{n} x_j^{a}ij$$

$A = [\ \boldsymbol{a}^{t}{}_{1}, \boldsymbol{a}^{t}{}_{2}, \ldots, \boldsymbol{a}^{t}{}_{m}\ ]$

$\boldsymbol{x}^{A} = (\boldsymbol{x}^{\boldsymbol{a}}{}_{1}, \ldots\ldots, \boldsymbol{x}^{\boldsymbol{a}}{}_{m})^{t}$

ついで，

$\boldsymbol{\Phi} = (\Phi\ (c, A, x_1)\ , \ldots\ \Phi\ (c, A, x_m))^{\mathrm{t}}$

とすることで，次の等式を得る。

$\boldsymbol{f}(\boldsymbol{c}, A) = \Phi - \boldsymbol{b} = \boldsymbol{0}$

これを満足する，$\boldsymbol{c}$ と A を Newton 法によって求めることが出来る。

Newton 法は次のような反復計算法で行える。

変数を書き直して，

$$\boldsymbol{y} = (\boldsymbol{c}^{t}, A)^{t}$$

とすると，結局次の連立方程式の解を求めることになる。

$$\boldsymbol{f}(\boldsymbol{y}) = \boldsymbol{0}$$

この式に対して，反復式は次のようになる。

$\boldsymbol{y}_{n+1} = \boldsymbol{y}_{n} - J\ (\boldsymbol{y}_{n})^{-1}\boldsymbol{f}(\boldsymbol{y}_{n})$

ここで，$J\ (\boldsymbol{y})$ は $\boldsymbol{f}$ の Jacobi 行列である。

すなわち，

$$\begin{pmatrix} \boldsymbol{c}_{n+1} \\ A_{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}_{n} \\ A_{n} \end{pmatrix} -$$

$$\begin{pmatrix} x_1^{a1} & .. & x_1^{am} & c_1 x_1^{a1}\ [\ lnx_1, .., lnx_n\ ] & . .. .. & c_m x_1^{am}\ [\ lnx_1, .., lnx_n\ ] \\ . & . & . & . & . .. .. . & \\ x_m^{a1} & . & . & c_1 x_m^{a1}\ [\ lnx_1, .., lnx_n\ ] & . .. .. & c_m x_m^{am}\ [\ lnx_1, .., lnx_n\ ] \end{pmatrix} \cdot\ ^{-1}\boldsymbol{f}(\boldsymbol{c}_n, A_n)$$

ここで，

$c_i x_j^{ai}\ [\ lnx_1, .., lnx_n\ ] = (\ c_i x_j^{ai}\ log_e x_1\ , .., c_i x_j^{ai}\ log_e x_n\ )$

である。

解が存在し，初期点が解の近傍で選択される場合は 2 次収束し，高速に求められることが知られている。

しかし，初期点の選び方によっては。解に収束しないことがあり，適用に当たっては工夫が必要である。$\boldsymbol{b}$ によっては，解が存在しないこともあるので，

$\|\Phi - \boldsymbol{b}\|$

を最小化する問題に変えることで，解の存在保証を得ることができるが，この場合はクラスの判定が保証されなくなることがある。

解の存在と収束性の吟味は今後のこととしたい。

例 6　Newton 法による計算例

クラス				
1	7	3.2	4.7	1.4
1	6.4	3.2	4.5	1.5
1	6.9	3.1	4.9	1.5
1	5.5	2.3	4	1.3
1	6.5	2.8	4.6	1.5

表 7　学習データ

y0	f0	y1	f1	y2	f2	y3	f3
1	146.392	0.056539	6.928884	0.058667	2.306671	0.114103	0.244415
1	137.24	0.974026	6.439897	0.503249	2.12857	−0.15295	0.197835
1	156.2165	0.993194	7.453803	0.867438	2.498083	0.600797	0.294845
1	64.78	1.010153	2.568517	1.189824	0.73618	1.273574	−0.14104
1	124.58	0.979173	5.763483	0.60109	1.88262	0.051099	0.134181
y4	f4	y5	f5	y6	f6	y7	f7
0.348876	−0.60056	0.991584	−0.1909	0.976158	0.024277	0.99887	1.90E−05
−0.41157	−0.6033	0.62161	−0.18241	−0.14343	0.021826	−0.0033	−4.00E−05
0.106248	−0.59718	−0.16145	−0.20102	0.036158	0.027423	0.00078	9.60E−05
0.583175	−0.60457	−0.87958	−0.17843	0.204321	0.020733	0.00474	−6.00E−05
−0.26703	−0.60637	0.404245	−0.17266	−0.09223	0.019213	−0.0021	−9.00E−05

表 8　計算結果

なお，このケースでは，対数をとることで解を簡単に見つけることができるので，Newton 法を適用する必要はないが，計算の例としてあげた。単項関数が 1 個のみなので，識別関数は正負のどちらかのみが構成できる。両符号をとる場合は，複数個の単項関数が必要となる。

3. アルゴリズム

識別関数がデータに依存して，判別率が異なってくることは十分予想される。

すなわち，学習するデータが検証データを含めた全データに占める割合や，どの部分を学習データとして選択し，識別関数を構成するかは構成した関数の識別の能力やデータに対する頑健性などに与える影響は少なくない。また，判別すべき対象が重要であれば，識別関数の構成は重要である。

次の諸点は，識別関数を構成するときに配慮しなければならない項目と考えられる。

- データセットの性質や偏りを調べておく。
- 複数の方法で，識別関数を構成し，識別能力の高いと思われる方法を選ぶ。
- 識別関数の構成法を選択すれば，その方法に用いる学習セットを複数の取り方で選択し，検証データで識別能力を評価する。評価のもっとも高いものを選ぶ。
- 未知のデータに対する識別能力の頑健性を保証するために，オーバーフィッティングに留意する。

アルゴリズム

Step1. データセットの収集

Step2. データクリーニング（不適なデータ除去）

Step3. 識別関数の構成法の選択

Step4. 学習データセットの選択

学習セットの選択回数が N（事前設定 etc）以上であれば Step3 へ

Step5. 識別関数の構成

Step6. 検証データで関数の評価

評価が低ければ Step4 へ

一定水準以上あれば停止

Ⅳ．おわりに

本論文では，識別関数の頑健性の維持と同時に識別率の向上について検討しいくつかの提案を行った。学習データの個数を抑制して識別関数を構成し，改良する方法は大変有効であると考えられる。実際，拡張 Lagrange 関数や整多項式形識別関数を構成し，計算例題を扱ったが，改良によって識別率の大幅な向上がみられた。今後は，より多くのケースを扱うことによって，本論での提案法が有効であることの検証を行うことや，ユークリッドノルムによる比較や，他の多項式形の可能性，Newton 法の解の存在や収束の為の初期点などの検討，さらにこれらを実装したシステムの作成などが今後の課題として残された。

Ⅴ．参考文献

［1］Burgs, C.:A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2, (1998)

［2］Duffin,R.J.,Peterson,E.L. and Zener,C.:
Geometric Programming Theory and Application; Jhon Wiley & Sons, 1967

［3］Fisher, R.A.: アイリスデータ　例えば
http://www.sci.kagoshima-u.ac.jp/~drs/ja/data/fishers_iris/

2005 年 7 月アクセス
［4］安道知寛，山下智志，「財務指標の時間依存を考慮した信用リスク評価モデル – デフォルト予測」
www.fsa.go.jp/frtc/seika/discussion/2004/20040406.pdf
［5］大澤良隆，油本暢勇「産業界における研究開発プロジェクト評価の一手法」748，754
［6］高森年「産業に役立つ技術評価の方法について」725，729
［7］真柄睦「CI に基づく技術評価」742，747 以上文献 4 所収
［8］計測と制御　VOL.43 NO.10 2004　　計測自動制御学会誌
［9］山口貴大「サポートベクターマシンによる倒産予測」2001
http://www-optima.amp.i.kyoto-u.ac.jp/research/paper/grad/yamaguchi.pdf
［10］山本哲郎 : 数値解析入門，サイエンス社，2009
［11］吉田要「技術評価支援システムの試作にむけて－識別関数の構成－
立命館経営学　Vol.44 No.4
［12］吉田要「技術評価支援システム試作」
－識別関数の構成オーバーフィッティング問題の解消－　Vol.49 No.1
［13］吉田　要：可変係数付き幾何計画問題の近似最適化法；計測自動 制御学会，
Vol.26，No.8，1990
［14］吉田要：幾何計画法の識別関数の構成への応用，
平成 16 年度三重地区計測制御研究会講演論文集，2004

付録 1　iris データの多次元 Lagrange 関数による識別［3］

					case1		case2							case1		case2	
1	5.7	3	4.2	1.2	1.026	1	1.02302	1	*−1*	6.7	3.3	5.7	2.5	11.09	0	−30.2	1
1	5.7	2.8	4.5	1.3	7.745	1	−12.048	1	*−1*	6.7	3	5.2	2.3	17.81	0	−10.2	1
1	6.3	3.3	4.7	1.6	11.34	1	−12	1	*−1*	6.3	2.5	5	1.9	10.03	0	−13.8	1
1	4.9	2.4	3.3	1	1051	1	−12.025	1	*−1*	6.5	3	5.2	2	9.142	0	−3.73	1
1	6.6	2.9	4.6	1.3	45.63	1	−12.033	1	*−1*	6.2	3.4	5.4	2.3	13.3	0	−19.3	1
1	5.2	2.7	3.9	1.4	184.3	1	−11.833	1	*−1*	5.9	3	5.1	1.8	7.351	0	−8.66	1
1	5	2	3.5	1	1114	1	5.15379	1	*−1*	6	3	4.8	1.8	5.525	0	−6250	1
1	5.9	3	4.2	1.5	6.225	1	3.54647	1	*−1*	6.9	3.1	5.4	2.1	14.38	0	−0.45	1
1	6	2.2	4	1	203.8	1	661.122	1	*−1*	6.7	3.1	5.6	2.4	4.698	0	−16.1	1
1	5.7	2.8	4.5	1.3	7.745	1	7.42779	1	*−1*	6.9	3.1	5.1	2.3	46.95	0	5.154	1
1	6.3	3.3	4.7	1.6	11.34	1	129.13	1	*−1*	6.8	3.2	5.9	2.3	12.29	0	3.546	1
1	4.9	2.4	3.3	1	1051	1	642.804	1	*−1*	7.2	3.2	6	1.8	183.3	0	661.1	1
1	6.6	2.9	4.6	1.3	45.63	1	5.20503	1	*−1*	6.2	2.8	4.8	1.8	7.51	0	7.428	1
1	5.2	2.7	3.9	1.4	184.3	1	112.097	1	*−1*	6.1	3	4.9	1.8	6.001	0	129.1	1
1	5	2	3.5	1	1114	1	−23.452	1	*−1*	6.4	2.8	5.6	2.1	3.138	0	642.8	1
1	5.9	3	4.2	1.5	6.225	1	−8.6769	1	*−1*	7.2	3	5.8	1.6	204.6	0	5.205	1
1	6	2.2	4	1	203.8	1	−2.8989	1	*−1*	7.4	2.8	6.1	1.9	499.8	0	112.1	1
1	6.1	2.9	4.7	1.4	7.581	1	−6.6382	1	*−1*	7.9	3.8	6.4	2	10154	0	2.551	1
1	5.6	2.9	3.6	1.3	242.9	1	−17.599	1	*−1*	6.4	2.8	5.6	2.2	4.309	0	174.4	1
1	6.7	3.1	4.4	1.4	145.6	1	−9.0005	1	*−1*	6.3	2.8	5.1	1.5	9.529	0	27.27	1
1	5.6	3	4.5	1.5	9.928	1	−1.1467	1	*−1*	6.1	2.6	5.6	1.4	51.68	0	6.599	1
1	5.8	2.7	4.1	1	17.64	1	−12.998	1	*−1*	7.7	3	6.1	2.3	2813	0	14.01	1
1	6.2	2.2	4.5	1.5	44.55	1	−13.535	1	*−1*	6.3	3.4	5.6	2.4	15.82	0	10.69	1
1	5.6	2.5	3.9	1.1	68.04	1	−19.674	1	*−1*	6.4	3.1	5.5	1.8	4.877	0	52.16	1
1	5.9	3.2	4.8	1.8	10.36	1	−34.951	1	*−1*	7.6	3	6.6	2.1	5231	0	1.676	1
1	6.1	2.8	4	1.3	29.88	1	−173.78	1	*−1*	4.9	2.5	4.5	1.7	407.2	0	22.39	1
1	6.3	2.5	4.9	1.5	10.37	1	0.19018	0	*−1*	7.3	2.9	6.3	1.8	642.1	0	−0.45	0
1	6.1	2.8	4.7	1.2	10.73	1	−2.9367	1	*−1*	6.7	2.5	5.8	1.8	33.67	0	4.928	1
1	6.4	2.9	4.3	1.3	28.84	1	−13.325	1	*−1*	7.2	3.6	6.1	2.5	444.1	0	12.88	1
1	6.6	3	4.4	1.4	79.41	1	−133.07	1	*−1*	6.5	3.2	5.1	2	12.25	0	18.04	1
1	6.8	2.8	4.8	1.4	67.52	1	−337.03	1	*−1*	6.4	2.7	5.3	1.9	5.656	0	−3.51	0
1	6.7	3	5	1.7	26.06	1	−2621.5	1	*−1*	6.8	3	5.5	2.1	6.812	0	−4.79	0
1	6	2.9	4.5	1.5	3.183	1	−16.267	1	*−1*	5.7	2.5	5	2	5.659	0	2.146	1
1	5.7	2.6	3.5	1	490.4	1	−5.6672	1	*−1*	7.7	3.8	6.7	2.2	9327	0	336.4	1
1	5.5	2.4	3.8	1.1	148.4	1	−91.972	1	*−1*	7.7	2.6	6.9	2.3	15323	0	107.5	1
1	5.5	2.4	3.7	1	170	1	−846.72	1	*−1*	5.8	2.8	5.1	2.4	32.7	0	126.1	1
1	5.8	2.7	3.9	1.2	19.84	1	−27.699	1	*−1*	6.4	3.2	5.3	2.3	9.721	0	16.67	1
1	6	2.7	5.1	1.6	6.737	1	−14.619	1	*−1*	6.5	3	5.5	1.8	4.945	0	−8.26	0
1	5.4	3	4.5	1.5	14.95	1	−3848.4	1	*−1*	6	2.2	5	1.5	15.8	0	9.922	1
1	6	3.4	4.5	1.6	19.61	1	152.087	0	*−1*	6.9	3.2	5.7	2.3	12.05	0	10.42	1
1	6.7	3.1	4.7	1.5	58.65	1	−666.24	1	*−1*	5.6	2.8	4.9	2	7.293	0	3.933	1
1	6.3	2.3	4.4	1.3	47.7	1	−65.351	1	*−1*	7.7	2.8	6.7	2	9327	0	19.61	1
1	5.6	3	4.1	1.3	5.256	1	−443.07	1	*−1*	6.3	2.7	4.9	1.8	10.37	0	4.853	1
1	5.5	2.5	4	1.3	64.94	1	−4.2542	1	*−1*	6.7	3.3	5.7	2.1	6.108	0	48.56	1
1	5.5	2.6	4.4	1.2	16.61	1	−8.2382	1	*−1*	6.3	2.9	5.6	1.8	−0.97	1	11.62	1
1	6.1	3	4.6	1.4	5.09	1	−11.549	1	*−1*	7.1	3	5.9	2.1	−1	1	2.504	1
1	5.8	2.6	4	1.2	20.41	1	−8.7626	1	*−1*	6.3	3.3	6	2.5	−0.99	1	16.67	1
1	5	2.3	3.3	1	972	1	−6250.5	1	*−1*	6.5	3	5.8	2.2	−0.98	1	611.8	1
1	5.6	2.7	4.2	1.3	7.55	1	−12395	1	*−1*	5.8	2.7	5.1	1.9	−0.7	1	6.53	1
															49		92

表 4　識別関数とパラメータ変更による改良

付録２　３次の同次式を用いた識別関数 20 個の学習データの選択を変更した場合

				真値	予測値 1	判別	予測値 2	判別	予測値 3	判別	予測値 4	判別	予測値 5	判別
17	45	25	49	1	1	1	4.792319	1	−0.28317	−1	−5.8835	−1	1	1
15	50	22	60	1	1	1	−18.7096	−1	4.262889	1	1.938049	1	1	1
20	50	25	57	1	1	1	4.559651	1	0.040721	1	23.10823	1	1	1
20	49	28	56	1	1	1	14.42106	1	1.224151	1	−2.67709	−1	1	1
19	51	27	58	1	1	1	9.478641	1	1.970974	1	−1.78846	−1	1	1
18	48	30	60	1	1	1	1.353477	1	8.468497	1	−3.50016	−1	1	1
18	48	28	62	1	1	1	−1.15496	−1	−0.58632	−1	6.774247	1	1	1
18	49	27	63	1	1	1	−1.50925	−1	0.941439	1	4.434052	1	1	1
19	50	25	63	1	1	1	−13.2606	−1	23.17639	1	5.884246	1	1	1
15	51	28	63	1	1	1	0.822573	1	−3.58436	−1	3.352116	1	1	1
14	56	26	61	1	24.58565	1	1	1	17.41633	1	−41.3173	−1	1.77522	1
18	49	30	61	1	4.915529	1	1	1	6.598402	1	−3.90341	−1	1.129123	1
18	51	30	59	1	4.795281	1	1	1	6.857705	1	−2.38726	−1	1.619024	1
24	51	28	58	1	−55.6752	−1	1	1	−39.4624	−1	77.43702	1	4.345796	1
13	52	30	67	1	32.2816	1	1	1	12.34206	1	15.69994	1	−5.2111	−1
19	53	27	64	1	−10.4813	−1	1	1	0.287315	1	5.448932	1	3.261978	1
18	56	29	63	1	−29.6446	−1	1	1	3.20371	1	−8.45741	−1	3.163943	1
20	52	30	65	1	3.57644	1	1	1	1.537484	1	9.235689	1	2.506828	1
20	51	32	65	1	−12.1208	−1	1	1	9.142336	1	5.920958	1	2.227744	1
18	55	31	64	1	−6.25716	−1	1	1	6.267607	1	−2.08966	−1	2.744983	1
18	55	30	65	1	−7.69918	−1	3.788691	1	1	1	−4.14453	−1	2.509655	1
18	58	25	67	1	−51.1692	−1	−28.1932	−1	1	1	1.280207	1	−1.87504	−1
21	56	28	64	1	−12.4535	−1	8.28649	1	1	1	11.08952	1	1.234656	1
22	56	28	64	1	−2.30904	−1	6.93015	1	1	1	25.66671	1	1.448828	1
23	53	32	64	1	−53.8481	−1	7.392325	1	1	1	24.51976	1	4.682717	1
23	54	34	62	1	−81.2158	−1	29.69191	1	1	1	−21.1142	−1	1.664136	1
23	51	31	69	1	−72.093	−1	−38.9515	−1	1	1	125.9403	1	−13.2884	−1
21	55	30	68	1	4.459284	1	−0.02221	−1	1	1	14.5664	1	3.197574	1
21	54	31	69	1	1.120374	1	−3.2495	−1	1	1	19.33768	1	1.513033	1
22	58	30	65	1	−1.73756	−1	16.17084	1	1	1	1.492172	1	0.212134	1
16	58	30	72	1	37.41097	1	−1.5312	−1	−8.01875	−1	16.31739	1	1.121062	1
24	56	34	63	1	−66.9156	−1	36.26932	1	−8.15883	−1	−22.5652	−1	1.152426	1
24	56	31	67	1	−7.05295	−1	−4.3285	−1	−6.63515	−1	55.88365	1	4.333599	1
21	57	33	67	1	12.14425	1	8.191164	1	9.657032	1	−8.76416	−1	2.154265	1
23	57	32	69	1	4.863068	1	3.722402	1	1.320561	1	24.5923	1	3.866222	1
21	59	30	71	1	−16.2037	−1	2.105891	1	−0.54598	−1	6.448115	1	4.390495	1
25	60	33	63	1	−13.256	−1	44.7682	1	−34.724	−1	−24.1127	−1	−3.87119	−1
25	57	33	67	1	−45.7295	−1	12.83414	1	−10.1408	−1	37.59605	1	4.917478	1
23	59	32	68	1	12.02734	1	16.86102	1	2.52802	1	4.646663	1	2.14363	1
18	60	32	72	1	−8.78339	−1	4.614138	1	−3.97384	−1	1	1	2.511223	1
19	61	28	74	1	−15.9334	−1	−20.8839	−1	5.549975	1	1	1	4.223272	1
18	63	29	73	1	−2.00284	−1	−11.3252	−1	−14.5035	−1	1	1	−2.69411	−1
23	61	30	77	1	0.506509	1	−28.0489	−1	54.63128	1	1	1	1.597233	1
20	67	28	77	1	−51.5628	−1	−60.7029	−1	−0.14619	−1	1	1	−7.39327	−1
21	66	30	76	1	−79.5527	−1	−16.4739	−1	−7.77482	−1	1	1	−0.70068	−1
25	61	36	72	1	−12.4198	−1	20.16817	1	8.804232	1	1	1	3.828902	1
23	69	26	77	1	−241.863	−1	−149.079	−1	96.51174	1	1	1	−10.0614	−1
20	64	38	79	1	−2.23048	−1	−2.61428	−1	4.626661	1	1	1	−0.82611	−1
22	67	38	77	1	−14.8575	−1	−2.0901	−1	12.70707	1	1	1	6.267764	1
10	35	20	50	−1	12.74386	1	−5.77212	−1	7.628652	1	−1.23986	−1	−1	−1
10	33	23	50	−1	0.098946	1	−7.14012	−1	−0.22243	−1	0.192426	1	−1	−1

				真値	予測値 1	判別	予測値 2	判別	予測値 3	判別	予測値 4	判別	予測値 5	判別
10	33	24	49	−1	−0.00449	−1	−7.84808	−1	−2.03488	−1	−2.10279	−1	−1	−1
11	30	25	51	−1	13.66885	1	35.81363	1	5.872111	1	38.18029	1	−1	−1
10	37	24	55	−1	3.764459	1	−18.1991	−1	5.519612	1	2.121626	1	−1	−1
10	35	26	57	−1	8.876403	1	−9.56634	−1	10.00924	1	2.249394	1	−1	−1
11	38	24	55	−1	1.721104	1	−11.8121	−1	0.496562	1	0.551025	1	−1	−1
11	39	25	56	−1	0.169233	1	−15.3207	−1	−0.40159	−1	1.06408	1	−1	−1
13	40	23	55	−1	2.062086	1	−1.79748	−1	0.387337	1	−3.55577	−1	−1	−1
14	39	27	52	−1	−3.62039	−1	−0.84326	−1	4.955666	1	−0.71162	−1	−1	−1
13	40	25	55	−1	−1	−1	−5.52608	−1	−3.81125	−1	−0.85839	−1	−0.39419	−1
13	36	29	56	−1	−1	−1	17.94392	1	3.393236	1	20.95264	1	3.195954	1
12	39	27	58	−1	−1	−1	−11.2252	−1	−2.33042	−1	−0.27714	−1	−1.36658	−1
12	40	26	58	−1	−1	−1	−12.3208	−1	−1.90525	−1	−0.43693	−1	−1.31349	−1
10	41	27	58	−1	−1	−1	−19.4786	−1	8.986698	1	1.732339	1	−5.74596	−1
12	44	26	55	−1	−1	−1	2.755118	1	4.409202	1	5.306975	1	−2.29336	−1
13	42	26	57	−1	−1	−1	−8.07786	−1	−4.34732	−1	−0.73152	−1	−0.3761	−1
13	42	27	56	−1	−1	−1	−7.29255	−1	−1.66332	−1	−1.60274	−1	−1.82547	−1
13	41	28	57	−1	−1	−1	−9.2296	−1	−2.28851	−1	−2.75573	−1	−2.11912	−1
13	41	30	56	−1	−1	−1	−0.38281	−1	−4.24811	−1	2.010758	1	−7.61821	−1
12	42	30	57	−1	−9.11566	−1	−1	−1	−6.3596	−1	3.689639	1	−11.3357	−1
13	40	28	61	−1	1.980973	1	−1	−1	0.122558	1	5.302817	1	−3.51345	−1
10	50	22	60	−1	359.3294	1	−1	−1	27.23361	1	30.09842	1	−14.3618	−1
13	44	23	63	−1	56.3343	1	−1	−1	58.39198	1	−35.1308	−1	−7.03515	−1
13	45	28	57	−1	−4.89408	−1	−1	−1	2.310184	1	4.815323	1	−3.62829	−1
15	45	22	62	−1	37.37502	1	−1	−1	77.6363	1	−60.8399	−1	−8.56108	−1
15	45	30	54	−1	−9.50051	−1	−1	−1	−7.93073	−1	27.85204	1	−4.01804	−1
15	42	30	59	−1	−6.58978	−1	−1	−1	3.702281	1	3.46173	1	−0.39574	−1
15	45	30	56	−1	−1.26837	−1	−1	−1	1.10503	1	9.836702	1	−4.46828	−1
12	47	28	61	−1	10.70728	1	−1	−1	7.98672	1	8.973105	1	−4.71015	−1
13	43	29	64	−1	−0.92416	−1	−15.5593	−1	−1	−1	−0.28946	−1	−2.26251	−1
15	45	29	60	−1	0.722872	1	−6.65414	−1	−1	−1	−2.69132	−1	−0.82784	−1
14	46	30	61	−1	−2.26447	−1	−8.72265	−1	−1	−1	−1.13779	−1	−4.00974	−1
14	47	29	61	−1	−1.75835	−1	−6.31441	−1	−1	−1	0.716692	1	−2.23792	−1
15	49	25	63	−1	8.968241	1	−5.49217	−1	−1	−1	0.514671	1	2.612472	1
14	44	30	66	−1	0.398721	1	−7.43372	−1	−1	−1	2.406541	1	−3.76318	−1
15	46	28	65	−1	1.14001	1	−3.7164	−1	−1	−1	−3.44712	−1	−3.15669	−1
16	51	27	60	−1	−11.299	−1	3.014645	1	−1	−1	−2.38971	−1	1.934212	1
16	45	34	60	−1	−20.8502	−1	15.54535	1	−1	−1	13.20557	1	−8.75232	−1
14	44	31	67	−1	2.112818	1	−7.06863	−1	−1	−1	4.73894	1	−3.25646	−1
15	45	32	64	−1	−0.85177	−1	−8.16206	−1	−1.46857	−1	−1	−1	−1.86714	−1
18	48	32	59	−1	−12.6097	−1	4.262781	1	10.86006	1	−1	−1	−1.81564	−1
14	48	28	68	−1	17.76213	1	−14.4395	−1	7.602366	1	−1	−1	−1.67546	−1
13	54	29	62	−1	13.31992	1	34.29554	1	33.31729	1	−1	−1	4.318445	1
16	47	33	63	−1	−1.11036	−1	−3.27682	−1	3.891696	1	−1	−1	−4.71561	−1
15	47	31	67	−1	−2.81044	−1	−13.5938	−1	−5.99714	−1	−1	−1	−1.7775	−1
14	47	32	70	−1	−2.06649	−1	−24.2475	−1	−1.65491	−1	−1	−1	−2.51385	−1
15	49	31	69	−1	−2.05503	−1	−17.4559	−1	−6.46392	−1	−1	−1	−1.41999	−1
17	50	30	67	−1	−0.59082	−1	−4.71765	−1	−6.01177	−1	−1	−1	0.182001	1
13	56	29	66	−1	92.74217	1	37.351	1	35.45496	1	−1	−1	−3.0456	−1
					ヒット数	51		73		62		62		85

表 5　3 次の同次式のみで構成した場合