人 工 知 能 87−12
（1993. 3. 4）

ブール関数の多項式時間学習アルゴリズム

月本 洋
(株) 東芝 研究開発センター

本稿ではブール関数の近似学習のアルゴリズムを提示する。その手法の基本は古典論理のモデルであるブール代数を初等代数で書き直しそれを拡張して得られる疑似線形関数の空間にブール関数が属していると言う事実に注目して、与えられた例を重回帰分析して得られた回帰関数をブール関数で近似することである。このアルゴリズムの計算量は条件つきで変数の数の多項式時間になる。

A Polynomial-time Algorithm for Boolean Functions

Hiroshi Tsukimoto
Research & Development Center, Toshiba Corporation

This paper presents an approximate learning algorithm for Boolean functions. This algorithm is based on the fact that Boolean functions belong to the extended space of classical logical functions. The algorithm consists of multiple regression analysis and approximation of the function by a Boolean function. Computational complexity of this algorithm is polynomial time of the number of variables on some conditions.

1 はじめに

本稿ではブール関数のあるクラスではなくブール関数全体に対する近似学習のアルゴリズムを提示する。PAC学習はk-CNF等のブール関数のあるクラスの確率的近似学習についての多項式時間学習可能性を主に議論してきた[Valiant 84]。また[Haussler 89]はPAC学習モデルを一般化して値域を実数に拡張し損失関数を用いた議論をしている。本稿ではブール関数を学習するために損失関数を用いて実数値関数を求める（重回帰分析）が、この実数値関数を再びブール関数で近似することによってブール関数を求める。

その手法の基本は、古典論理のモデルであるブール代数を初等代数で書き直しそれを拡張して得られる疑似線形関数の空間にある内積を導入してその空間をユークリッド空間にするが、この空間にブール関数が属していると言う事実に注目して、与えられた例の{0,1}を[0,1]の{0,1}とみなしその例を重回帰分析して得られた回帰関数をユークリッド距離で最も近いブール関数で近似することである。このアルゴリズムの計算量は条件つきで変数の数の多項式時間になる。このアルゴリズムは著者が以前提案した論理の幾何的モデル[月本90]の応用である。

2 論理の幾何的モデル

論理の幾何的モデルは論理関数の空間に距離を導入したもので、関数解析の論理関数版と言える。詳細は[月本 90]を参照。

2.1 初等代数による古典論理のモデル

古典論理の代数モデルであるブール代数を初等代数で書き換え、古典論理の初等代数によるモデルを構築する。それを以下のようなL_1を定義し、それが古典論理のモデルになっていることを示すという手順で行なう。まずL_1を以下のように帰納的に定義する。

1. 変数はL_1の要素である。
2. もしxとyがL_1の要素ならば, $\tau(x\cdot y)$と$\tau(x+y-x\cdot y)$と$\tau(1-x)$もL_1の要素である。(以降この計算をτ計算と呼ぶ。)
3. L_1は上記の1と2を有限回適用して生成された関数の全体である。

但し、τは$\tau(f(x^n,y^n,...))=f(x,y,...)$, ($x,y,...\in\{0,1\}$) という写像である。

そして次の対応を与える。$F\wedge G\Leftrightarrow\tau(fg)$, $F\vee G\Leftrightarrow\tau(f+g-fg)$, $\overline{F}\Leftrightarrow\tau(1-f)$
上記のF,Gは命題であり、f,gはそれぞれ対応する関数である。上記の対応のもとでL_1は古典論理のモデルである。証明は[月本 90]を参照。

2.2 疑似線形関数への拡張

ここでは、モデルを疑似線形関数の空間(L)に拡張する。疑似線形関数とは$\tau(f)=f$を満たす関数である。2変数では$axy+bx+cy+d(a,b,c,d\in\mathbf{R})$となる。この拡張は$\tau(f^2)=f$をはずす事に相当する。$\tau(f^2)=f$は巾等律を意味する為、これは巾等律をはずす事を意味する。だが、τ計算により変数x及び$1-x$に関しては巾等律をはずさない事になるので、巾等

律をはずすのはリテラルに対応する式以外の式(非リテラル式)に対して行う事になる。これにより対象領域は、疑似線形関数 $f(\{0,1\}^n \rightarrow R)$ に拡張される。

また関数の定義域を {0,1} から [0, 1] へ拡張する。この拡張により、関数 f は $f:[0,1]^n \rightarrow R$ となる。

2.3 L をユークリッド空間にする

2.3.1 内積の定義

内積 $< f,g >$ を次の様に定義する。$< f,g >= 2^n \int_0^1 \tau(fg)dx$

2.3.2 ノルムの定義

ノルムの定義を次の様に行う。$\| f \| = \sqrt{< f,f >}$ L は上記の内積、ノルムに関し、有限な内積空間、即ちユークリッド空間になる。これは L が疑似線形関数から構成される事から簡単にわかる。

2.3.3 直交関数展開

ブール代数の原子を拡張した直交関数を次の様に定義する。(n 変数とする。)

$\phi_i = \prod_{j=1}^{n} e(x_j)$ $(i = 1 \sim 2^n, j = 1 \sim n)$, $e(x_j) = 1 - x_j$ または x_j

例えば、1変数の場合の正規直交系は $\{x, 1-x\}$ となり、2変数の場合の正規直交系は $\{xy, x(1-y), (1-x)y, (1-x)(1-y)\}$ となる。上記の様に定義した ϕ_i に対して、

$< \phi_i, \phi_j >= 0 \ (i \neq j)$

$= 1 \ (i = j)$

$f = \sum_{i=1}^{2^n} < f, \phi_i > \phi_i$

となる事は容易に確められる。この直交関数展開は、ブール代数の原子による展開を拡張したものと言える。$(< f, \phi_i >)$ を論理ベクトルと呼ぶ。

3 重回帰分析による学習アルゴリズム (原型)

3.1 アルゴリズム (原型)

以下にアルゴリズムの原型を示す。

1. ブール関数の例の値 {0, 1} を実数の閉区間 [0, 1] の {0, 1} と見なして重回帰分析を行なう。

2. 重回帰分析の結果である線形関数を直交展開して各直交関数の係数を求める。即ち論理ベクトルを求める。計算方法は本節の例 (後述) を参照。

3. 論理ベクトルをブールベクトル (ブールベクトルとは要素が 0,1 のベクトルである) で近似する。即ちユークリッド距離で最も近いブールベクトルを求める。重回帰分析で得られた関数 (f) に最も近いブール関数 (g)、即ち $\| f - g \|$ を最小にする g, を求めるのであるが、直交関数展開ができる (ユークリッド空間のベクトルとして表現できる) た

め、探索をする必要はない。f を直交関数展開して求めたベクトルを $\mathrm{f}(=(f_i))$ とし、g に対応するベクトルを $\mathrm{g}(=(g_i))$ とすると、最小にしたい量は $\sum(f_i-g_i)^2$ となる。各項 $(f_i-g_i)^2$ は独立なので各項を最小にすることが $\sum(f_i-g_i)^2$ を最小にすることと同じになる。g_i は $1,0$ のいずれかなので、$(f_i-g_i)^2$ を最小にする g_i は $f_i \geq 0.5$ ならば $g_i=1$ その他で $g_i=0$ となる。

4. このブールベクトルをブール関数に変換し簡約化する。

2変数の論理関数 $x \vee y$ で説明する。例は4例与えるとする。表1を参照。これを重回帰分析すると $0.66x+0.66y$ が得られる。$0.66x+0.66y = axy+bx(1-y)+c(1-x)y+d(1-x)(1-y)$ とおいて a,b,c,d を求めると論理ベクトル (a,b,c,d) は $(1.32,0.66,0.66,0)$ となる。これを最も近い古典論理ベクトルで近似すると $(1,1,1,0)$ となり、ブール関数は $xy \vee x\overline{y} \vee \overline{x}y$ となり、簡約化すると $x \vee y$ となる。

3.2 2変数の例

表1 $x \vee y$ の例

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

表2 2変数の例

関数	学習の結果	距離	残差平方和
0	0	0	0
xy	xy	0	0.33
$x\overline{y}$	$x\overline{y}$	0	0.33
$\overline{x}y$	$\overline{x}y$	0	0.33
$\overline{xy}$	$\overline{x}y$	1	1
x	x	0	0
y	y	0	0
$xy \vee \overline{xy}$	xy	1	1.33
$\overline{x}y \vee x\overline{y}$	xy	3	1.33
$\overline{x}$	$\overline{x}y$	1	1.33
$\overline{y}$	$x\overline{y}$	1	1.33
$x \vee y$	$x \vee y$	0	0.33
$x \vee \overline{y}$	x	1	1
$\overline{x} \vee y$	y	1	1
$\overline{x} \vee \overline{y}$	xy	4	2.33
1	$x \vee y$	1	1.33

2変数のブール関数全てに対して重回帰分析（例数=4）を行なう。その結果を表2に示す。表2より分かる通り、いくつかの関数は正解を学習できない。正解を学習できない関数は否定を含んでいるか、定数である(=1)。特に $\overline{x}y \vee x\overline{y}$ と $\overline{x} \vee \overline{y}$ は学習されたブール関数の正解との距離が3または4と学習結果が極めて悪い。この理由としては回帰式を $ax+by$ としているために 1. 定数項がない。2. 回帰式が肯定 (x,y) の線形結合であり、否定 $(\overline{x},\overline{y})$ の近似がしにく

い。等が考えられる。また学習されたブール関数と正解との距離と重回帰分析の残差平方和は相関が高いと考えられる。

4 重回帰分析について

ここでは2変数の実験の結果に基づいて重回帰分析の近似精度向上について議論する。

4.1 定数項について

定数項を追加することによって近似精度を上げることは可能であるが、例数が少ない場合にはこの定数項が存在することによって、例間の線形独立性がなくなり重回帰分析が出来なくなる場合がある。例数が多くて例間の線形独立性が失われないのならば定数項を付加して重回帰分析した方が近似精度が良くなる。従って定数項付きの重回帰分析は例数が多い場合に近似精度を上げる方法として有効であるが、本稿では定数項なしで議論する。

4.2 変数反転

また否定を含む場合は $x\prime = \overline{x}$ 等の変数変換によって否定を消すことが出来るがこれは学習するブール関数が否定を含んでいることをあらかじめ分かっていなければならないが、これは不可能である。しかしこれに近い手法が可能である。

表3 $\overline{x} \vee y$ の例

x	y	$\overline{x} \vee y$
0	0	1
0	1	1
1	0	0
1	1	1

表4 $x \vee y$ の例

x	y	$x \vee y$
1	0	1
1	1	1
0	0	0
0	1	1

例えば表3の例では変数 x が0の時の方が1の時より多くブール関数の値が1になっている。ここで変数を x から $\overline{x}$ に反転し新たに x とおくと変数 x が1の時の方が0の時より多く1になり(表4参照)、ブール関数 $x \vee y$ の例と同じになる。従ってこのように変数を反転して重回帰分析をしてブール関数を求めると $x \vee y$ が得られ変数を元に戻すと $\overline{x} \vee y$ が得られる。この手法を一般的に述べると、全ての変数についてその変数が0の時の方が1の時より多くブール関数の値が1になっている場合にはその変数を反転して重回帰分析してブール関数を求め、最後に再び変数を反転して最終的なブール関数を求めることになる。この変数反転により2変数の例では $xy \vee \overline{x}\overline{y}, \overline{x}y \vee x\overline{y}$ 以外は正解を得る。(但し、ブール関数1は例外とする。これは定数項がないから、正解にならない。)

変数反転でも良好に近似学習出来ない関数に関しては、回帰関数を線形関数から疑似線形関数に拡張して重回帰分析することを考える。これについては次節で述べる。

4.3 高次の項の追加

学習されたブール関数と正解との距離を残差平方和で評価する。線形関数による重回帰分析を行なって残差平方和が大きい場合は高次の関数項を追加して再び重回帰分析を行なうことにする。2変数で言えば回帰関数を線形関数 $ax+by$ に xy を追加して $axy+bx+cy$ とすることになる。3変数で言えば回帰関数を線形関数 $ax+by+cz$ に xy, yz, zx を追加して $axy+byz+czx+dx+ey+fz$ とすることになる。もしこれでも残差平方和が大きいならばさらに高次の項 xyz を付加して重回帰分析を行なう。以下同様にして残差平方和が一定値以下になるまで高次の項を追加する。

4.4 重回帰分析の計算量

前節までの手法により近似精度は向上されたのでここでは計算量について考える。まず重回帰分析の計算量は回帰関数の変数（＝項）の多項式時間である。最初の回帰関数即ち線形関数の項（x, y 等）の数は $n=\binom{n}{1}$ であるが、その次の高次の項（xy, yz 等）の数は $\binom{n}{2}$ となり、一般に $\binom{n}{m}$ となる。従って p 次までの項を用いた回帰関数の項の数は $\sum_{m=1}^{m=p}\binom{n}{m}$ となる。これはブール関数の変数の数 n の多項式である。従って重回帰分析の計算量はブール関数の変数の数 n の多項式時間であると言える。しかし最高次の項まで使うと項数は $\sum_{m=1}^{m=n}\binom{n}{m}=2^n$ となり、指数関数になり、計算量は指数関数時間になる。従って重回帰分析の計算量は基本的に多項式時間である。どの高次の項まで使うかは残差平方和の設定と関係している。即ちより精度良く近似しようとすれば計算量は多項式時間から指数関数時間に近付く。

5 ブール関数同定の多項式時間アルゴリズム

3.1 のアルゴリズムの原型における2番目から4番目の処理は重回帰式の係数からブール関数を同定する処理である。この処理をブール関数同定と呼ぶ。このブール関数同定のアルゴリズムの計算量であるが、このアルゴリズム中で直交関数の係数を全て求めている。その係数の数は 2^n(n は変数の数) であるから、ブール関数同定の計算量は多項式時間ではないので、ブール関数同定の多項式時間のアルゴリズムを提示しなければならない。

ここで回帰式を $p_1x_1+\ldots+p_nx_n$ とし、それの直交関数による展開式を $a_1x_1\cdot\cdot x_n+\ldots+a_{2^n}(1-x_1)\cdot\cdot(1-x_n)$ とすると、$p_1x_1+\ldots+p_nx_n=a_1x_1\cdot\cdot x_n+\ldots+a_{2^n}(1-x_1)\cdot\cdot(1-x_n)$ となる。ここで x_1 に注目する。ブール関数に近似された後で x_1 が存在するには $x_1x_2\cdot\cdot x_n, x_1x_2\cdot\cdot(1-x_n), \ldots, x_1(1-x_2)\cdot\cdot(1-x_n)$ の全ての直交関数が存在しなければならない。即ち各直交関数の係数が 0.5 以上でなければならない。即ち $a_1, a_2, \ldots, a_{2^{n-1}}$ が全て 0.5 以上でなければならない。ところで a_i は次の通りである。

$a_1=p_1+\ldots+p_n,\ a_2=p_1+\ldots+p_{n-1}, \ldots, a_{2^{n-1}}=p_1$

各係数 a_i はそれに対応する直交関数の値のみ1にし、他の直交関数の値を0にするような変数の値の組合わせを代入することによって簡単に求められる。例えば a_1 ならばこれに対応する直交関数 $x_1\cdot\cdot x_n$ の値を1にする変数の値の組合わせは $x_1=x_2=\cdot\cdot=x_n=1$ である。これを代入すると右辺は a_1 となり、左辺は $p_1+\ldots+p_n$ となり、$a_1=p_1+\ldots+p_n$ が得られる。同様に $x_1=x_2=\cdot\cdot=x_{n-1}, x_n=0$ を代入すると $a_2=p_1+\ldots+p_{n-1}$ となる。他の a_i も同様にして計算できる。従って x_1 がブール関数に近似された後で存在するには

$a_1 = p_1 + ... + p_n \geq 0.5,\ a_2 = p_1 + ... + p_{n-1} \geq 0.5, ..., a_{2^{n-1}} = p_1 \geq 0.5$
とならねばならない。ここで p_i が全て非負であると仮定すると $a_{2^{n-1}} = min\{a_i\}$ となり、$a_{2^{n-1}} \geq 0.5$ ならば $a_i(1 \leq i \leq 2^{n-1})$ が全て 0.5 以上になる。すなわち $p_1 \geq 0.5$ ならば $a_i(1 \leq i \leq 2^{n-1})$ が全て 0.5 以上になり、x_1 が近似後のブール関数に存在することになる。回帰式 $p_1x_1 + ... + p_nx_n$ は x_i に関して対称であるから以上の議論は他の x_i についても成立する。即ち $p_i \geq 0.5$ ならば x_i が近似後のブール関数に存在する。

今までの議論は最も低次の項についての議論であったが同様の議論が高次の項の存在についても可能であり、その結果一般に $x_{i_1} \cdots x_{i_k}$ が近似後のブール関数に存在する条件は $p_{i_1} + ... + p_{i_k} \geq 0.5$ となる。そして存在が確認されたブール関数を論理和で接続して最終的なブール関数を得ることになる。以上の議論は回帰式が線形関数の場合であるが同様の議論が一般の疑似線形関数でも可能である。

なお低次の項で存在が確認された変数に関してはその項より高次の項での存在の判定の必要がない。簡単な例で言えば x が存在することが分かれば xy, xz 等は存在する（$x = x \vee xy \vee xz$）ので xy, xz の存在の判定は必要ないのである。このブール関数同定アルゴリズムの例を以下に挙げる。$y = 0.65x_1 + 0.23x_2 + 0.15x_3 + 0.20x_4 + 0.02x_5$ を重回帰分析の回帰式とする。まず x_i に関しては x_1 の存在条件 p_1 のみが 0.5 以上なので x_1 が存在することになる。つぎに x_1 を除外した x_ix_j に関してはどの p_i と p_j の和も 0.5 を越えないので x_ix_j と言う項は存在しないことになる。そして次に x_1 を除外した $x_ix_jx_k$ に関しては $x_2x_3x_4$ の存在条件 $p_1 + p_2 + p_3$ が 0.5 以上なので $x_2x_3x_4$ が存在することになる。x_1, x_2, x_3, x_4 を除外すると x_5 が残るが、これ一つでは更に高次の項は作れないのでここでアルゴリズムは終了する。従って存在するのは x_1 と $x_2x_3x_4$ となり、その論理和 $x_1 \vee x_2x_3x_4$ が求めるブール関数となる。

上記のブール関数のアルゴリズムは簡単な計算より多項式時間であることが分かる。但し重回帰分析と同様高次の項まで同定しようとすると計算量が増加して行き最高次まで同定するとその計算量は指数関数時間になる。また p_i が全て非負であると言う仮定は変数反転をしているから厳しい制約にはならないと考える。この非負条件を満たさない場合の多項式時間アルゴリズムは今後の課題である。

6 重回帰分析による学習アルゴリズム

6.1 アルゴリズム

以上の議論をまとめるとアルゴリズムは以下のようになる。

1. 与えられた例で、全ての変数についてその変数が 0 の時の方が 1 の時より多くブール関数の値が 1 になっている場合にはその変数を反転する。

2. 回帰関数を線形関数から始めて残差平方和が一定値以下になるまで高次の項を追加して重回帰分析を行なう。

3. 重回帰分析の結果である回帰式（一般に疑似線形関数）の係数が全て非負ならば前章の多項式時間アルゴリズムを用いてブール関数を求める。全て非負でない時は直交関数展開してブール関数を求める。

4. 反転した変数を元に戻す。

従って重回帰分析で得られた回帰式の係数が全て非負でない時は多項式時間アルゴリズムでなくなる。

6.2 関数形について

本論文のアルゴリズムは通常の PAC 学習とは違い、ブール関数の表現形を決めていない。従って基本的には全てのブール関数に適用できるのであるが、今までの議論より分かる通り、基本的に学習するブール関数が否定を含んでない方が好ましい。否定を消す方法として変数反転を提示したがこれは勿論万能ではない。既に指摘した通り $xy \vee \overline{xy}$ 等の関数に対してはこの変数反転は有効ではない。また簡約も否定を消す。例えば $y \vee xz = y \vee x\bar{y}z$ である。このように考えてくると本論文のアルゴリズムが得意とするブール関数は簡約、変数反転によって単調多項式になる関数であると予想される。このブール関数のクラスを以降拡張単調多項式と呼ぶ。また回帰式からブール関数を求めるアルゴリズムでは回帰式の係数が非負であることが好ましい。非負であることと拡張多項式であることは関係ありそうである。最後に求めるブール関数の表現形は以上の議論より分かる通り DNF であるが、ブール関数の同定で多項式時間アルゴリズムを使うと拡張単調多項式となる。

7 実験

ここでは上記アルゴリズムを用いていくつか実験を行なう。但し基本的に線形関数のみによる重回帰分析とし、高次の項は使わない。

7.1 実験 1- 拡張単調多項式

3 変数の関数 で拡張単調多項式 $y \vee xz$ を表 5 の例から求める。例は上から下に追加して行く。

表 5 $y \vee xz$ の例

x	y	z	$y \vee xz$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

表 6 $x \vee yz \vee w$ の例

x	y	z	w	$x \vee yz \vee w$
1	0	0	1	1
1	0	0	0	1
0	1	1	1	1
0	1	1	0	1

結果は、3変数だから最小必要例数は3であるが4まで線形従属の問題で重回帰分析できなかった。例数5, 6, 7, 8で各々結果は$y, y \vee xz, y \vee xz, y$となった。従って例数6,7で正解であったが、8で正解を得られなかった。これより正解との距離は例数に対して単調減少ではないことが分かる。学習は例の与え方に依存するので例数3で可能かも知れない。また回帰式の係数は全て非負であり、ブール関数同定では多項式時間アルゴリズムを使用できた。

7.2 実験2- 非拡張単調多項式

3変数の非拡張単調多項式$xy \vee \bar{y}z$を例から求める。実験1と同じ例の与え方を行なった結果、例数3,4,5,6,7,8で各々$x, y \vee z, x, x, x \vee z, yz$を得た。これより1. やはり例数に対して正解との距離は単調減少ではない。2. 正解が得られない。これは拡張単調多項式でないから学習が難しいと思われる。3. 最小必要例数で一応重回帰分析が出来た。がわかる。正例4例を与えた時に$z \vee xy$を得た。これは正解との距離1の概念である。従って正例のみを与えた方が学習しやすいのかも知れない。なお回帰式の係数は全て非負であり、ブール関数同定では多項式時間アルゴリズムが使用できた。従って非拡張単調多項式でも回帰式の係数が全て非負となる場合もあることが分かる。

線形関数での重回帰分析では正解が得られなかったので高次の項を追加して$axy + byz + czx + dx + ey + fz$で重回帰分析すると例数6,7,8で$xy \vee \bar{y}z$を学習した。（ただし例の与え方は今までと逆にして$x = 1, y = 1, z = 1$から始めた。以降同様。）従って高次の項を追加すれば非拡張単調多項式を学習出来ることが分かる。但しブール関数同定は回帰式の係数が全て非負ではないので多項式時間アルゴリズムは使えなかった。

7.3 実験3- 細かい概念

実験2では学習しにくい非拡張単調多項式を調べたが、今度は拡張単調多項式でも学習しにくいと思われる「細かい」ブール関数を調べてみる。例題は3変数のxyzとする。予想通り8例与えても0しか得られない。即ち正解から距離1の概念までしか学習できない。そこで回帰式を高次の項を追加して$axy + byz + czx + dx + ey + fz$とすると例数3,4で$yz$を学習し、例数5,6,7,8で$xyz$を学習した。従って高次の項を追加すれば「細かい」概念を学習できることになる。但しブール関数同定は回帰式の係数が全て非負ではないので多項式時間アルゴリズムは使えなかった。

さらに4変数の$xyz \vee yzw \vee zwx$を調べる。この関数を$ax + by + cz + dw$で近似するのは結構難しいのではないかと予想される。結果は例数13, 14, 15,16で正解を学習した。この結果をどう評価するかは難しい。なお回帰式の係数は全て非負であり、ブール関数同定では多項式時間アルゴリズムが使用できた。

7.4 実験4- 良い例

例が良ければ最小必要例数で正解を学習できる例を示す。4変数の$x \vee yz \vee w$は表6の例から正解を学習できた。従って例が良ければ変数の数と同じ例数＝最小必要例数で正解が学習できる場合もあることが分かる。なお回帰式の係数は全て非負であり、ブール関数同定では多項式時間アルゴリズムが使用できた。

7.5 実験 5- 特徴的な例

4 変数の $xw \vee xyz$ の学習を考える。$xw \vee xyz = x \wedge (w \vee yz)$ である。そこで表 7 のような $x = 1$ である全例 8 個を与える。

表 7 $xw \vee xyz$ の例

x	y	z	w	$xw \vee xyz$
1	1	1	1	1
1	1	1	0	1
1	1	0	1	1
1	1	0	0	0
1	0	1	1	1
1	0	1	0	0
1	0	0	1	1
1	0	0	0	0

学習結果は $w \vee yz$ である。表 7 の正例のみの論理和をとると $xw \vee xyz$ であり正解になるのであるが、重回帰分析をして解を求めると x が落ちる。これは次のように解釈できる。表 7 のように $x = 1$ の例を与えることは、求める関数を $f(x,y,z,w)$ とすると $f(1,y,z,w)$ の例を与えることである。$f(1,y,z,w) = g(y,z,w)$ とおくと、問題は $f(0,y,z,w)$ をどう推測するかである。重回帰分析の結果は $f(0,y,z,w) = g(y,z,w)$ と推測したことになり、単純な論理和は $f(0,y,z,w) = 0$ と推測したことになる。どちらが妥当であるかは一概に判断できないであろう。なお回帰式の係数は全て非負であり、ブール関数同定では多項式時間アルゴリズムが使用できた。

8 終りに

本稿では重回帰分析によるブール関数の近似学習アルゴリズムを提示した。このアルゴリズムは回帰式の係数が全て非負の場合には多項式時間アルゴリズムになる。今後の課題は誤差と例数の関係等の理論的解析、回帰式の係数が全て非負でない場合の多項式時間アルゴリズムの発見等である。

参考文献

[Haussler 89] D. Haussler: Generalizing the PAC Model: Sample Size Boounds From Metric Dimension-based Uniform Convergence Results,*Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, pp40-45, 1989.

[Valiant 84] L.G. Valiant: A Theory of the Learnable, *Comm. ACM*, Vol.27, No.11, pp.1134-1142, 1984.

[月本 90] 月本 洋: 命題論理の幾何的モデル, 情報処理学会論文誌, Vol.31, pp.783-791, 1990.