

3ZB-1

# サポートベクタマシンのハードウェアによる一実現手法

森岡 健一<sup>†</sup> 森木 一紀<sup>‡</sup> 桑子 雅史<sup>††</sup> 持木 幸一<sup>†††</sup>

<sup>†</sup>武蔵工業大学大学院工学研究科電気工学専攻

<sup>‡</sup>武蔵工業大学工学部電気電子情報工学科

<sup>††,†††</sup>武蔵工業大学工学部コンピュータ・メディア工学科

## 1. 序論

現在、パターン認識の分野でサポートベクタマシン (SVM) [1] という認識手法が、評価サンプルに対して高い識別率を示すとして、注目されている。SVM は基本的にはパーセプトロンと同じ2クラスの線形識別器であるが、マージン最大化[1]の手法を用いることによって、高い汎化性能を示す。また、カーネルトリック[1]を用いることで、複雑な計算を行わずに非線形な識別を可能にしている。

現在、このSVMを組み込み分野へ適用しようという研究が行われている。代表的なものとしては、Kerneltron[2]や[3]などの研究が行われている。これらの研究は、SVMの処理をより高速に実行するために、SVMのアルゴリズムをハードウェアで実現する研究である。本研究は其中で、追加学習が容易な Incremental SVM learning アルゴリズム[4]を学習アルゴリズムとして用いた SVM のハードウェアを開発し、Field Programmable Gate Array(FPGA)へ実装を行う。

## 2. 原理

### 2.1. SVMの原理[1]

今、学習用サンプルを  $(\mathbf{x}_i, y_i)$  とする。このとき、 $\mathbf{x}_i$  はサンプルの特徴ベクトル、 $y_i$  はクラスラベル ( $y_i \in \{+1, -1\}$ ) である。また、判別関数を  $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$  とする。ここで、 $\mathbf{w}$  は重みベクトル、 $b$  は閾値である。SVMではこの $\mathbf{w}$ 、 $b$ をマージン最大化[1]という規則を用いて決定する。これは2次計画問題を解くことによって求め、その式は、

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^n \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to} \quad \alpha_i \geq 0$$

となる。

Title: A realization approach for hardware Support Vector Machine

<sup>†</sup>: Kenichi MORIOKA, Major of Electrical Engineering, Musashi Institute of Technology, Research Division in Engineering

<sup>‡</sup>: Kazunori MORIKI, Department of Electrical and Electronic Engineering, Faculty of Engineering, Musashi Institute of Technology

<sup>††,†††</sup>: Masashi KUWAKO, Koichi MOCHIKI, Department of Computer Science and Media Engineering, Faculty of Engineering, Musashi Institute of Technology

2次計画問題を高速に解くアルゴリズムは、[5]など様々なアルゴリズムが提案されているが、本研究では Incremental SVM learning アルゴリズム[4]を採用する。これは追加学習が容易なアルゴリズムであり、組み込み用としてハードウェアで開発した後の追加学習が簡単に行えると考えられる。このアルゴリズムをハードウェアで実行する際には、データの依存関係を考慮して並列化が可能である。並列化したアルゴリズムは以下の表1のようになる。左の番号は実行順序を表しており、番号内の複数の式は並列に実行する。

表1 Incremental SVM learning

$$1. \quad g_i = \sum_{j \in S} Q_{ij} \alpha_j + y_i b - 1 \quad (Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j)$$

2. If  $g_i > 0$  . learning stop

$$3. \text{ Else } \begin{bmatrix} \beta \\ \beta_{j1} \\ \vdots \\ \beta_{jn} \end{bmatrix} = -R \begin{bmatrix} y_i \\ Q_{ij1} \\ \vdots \\ Q_{ijn} \end{bmatrix}$$

$$4. \quad \gamma_i = \sum_{j \in S} Q_{ij} \beta_j + Q_{ii} + y_i b$$

$$\min \quad \Delta \alpha_j = \frac{C - \alpha_j}{\beta_j} \quad (\text{all } j \in S) \quad C: \text{制約パラメータ}$$

$$5. \min \quad \Delta \alpha_l = \frac{-g_l}{\gamma_l} \quad (\text{all } l \in O, E)$$

6. If  $\min \Delta \alpha_j (\text{all } j \in S) > \min \Delta \alpha_l (\text{all } l \in O, E)$   
SupportVector increment  
Else, SupportVector decrement

$$7. \quad \alpha_j^{\text{new}} = \alpha_j^{\text{old}} + \beta_j \Delta \alpha_j \quad (\text{all } j \in S)$$

$$g_l^{\text{new}} = g_l^{\text{old}} + \gamma_l \Delta \alpha_l \quad (\text{all } l \in O, E)$$

$$b^{\text{new}} = b^{\text{old}} + \beta \Delta \alpha_i$$

If SupportVector increment,

$$R^{\text{new}} \leftarrow \begin{bmatrix} & 0 \\ R^{\text{old}} & 0 \\ & \vdots \\ 0 \dots 0 & 0 \end{bmatrix} + \frac{1}{\gamma_{m+1}} \begin{bmatrix} \beta \\ \beta_{j1} \\ \vdots \\ \beta_{jn} \end{bmatrix} \cdot [\beta, \beta_{j1}, \dots, \beta_{jn}]$$

If SupportVector decrement,

$$R_{ij}^{\text{new}} \leftarrow R_{ij}^{\text{old}} - \frac{R_{ik}^{\text{old}} \cdot R_{kj}^{\text{old}}}{R_{kk}^{\text{old}}}$$

8. Go to 2.

## 3. ハードウェアアーキテクチャ

表1の学習アルゴリズムを実行するハードウ

ウェアを開発し、FPGAに実装を行う。FPGA内のブロック図を以下の図1に示す。なお、図1中のデータバスは図が煩雑になるために、バスで表現してある。FPGA内への外部からのデータ入出力は全て Interface Block が行う。

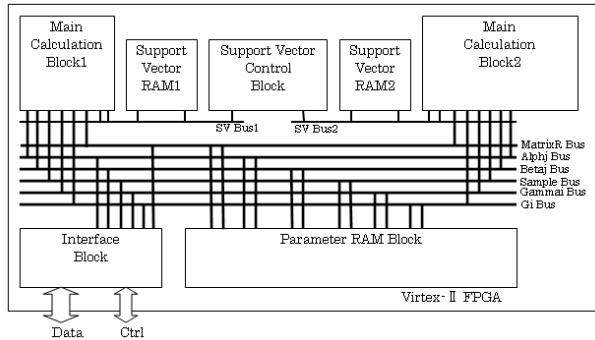


図1 FPGA内ブロック図

学習に必要なパラメータは Parameter RAM Block に保持する。学習アルゴリズムの計算は Main Calculation Block1、2 で行う。Main Calculation Block2 では除算を含む計算を行うブロックである。また、サポートベクターの情報は Support Vector RAM にあり、その更新は Support Vector Control Blockで行う。

このアーキテクチャの特徴は、同じデータが保持してあるメモリを2つ用意したことである。これは、アルゴリズムの並列化に伴って生じるメモリアクセスの競合を防ぐためである。具体的には Support Vector RAM を2つに分けることによって、Main Calculation Block1,2 がそれぞれサポートベクター情報を読み出す際の競合を防ぐ構成となっている。この競合が起きる状態は、表1の7.である。7.のとき、Main Calculation Block1 では、 $\alpha_j^{new}$  を計算するために、Support Vector RAM へのアクセスが1つある。同時に Main Calculation Block2 では、 $R^{new}$  の更新に伴う  $\beta_j$  の読み出しで Support Vector RAM へのアクセスが2つあり、合計で同時に3つのアクセスが発生する。FPGA内の専用領域を使うRAMは読み出しポートは2つまでしか作成できない。そのため、Support Vector RAM が1つでは、Main Calculation Block1、2 間でアクセスの競合が起きてしまう。そこで、競合が起きないように、Support Vector RAM を2つ用意することによって、これらの計算を高速に行える構成とした。

#### 4. 結果

開発したハードウェアは Xilinx 社製 FPGA である Virtex- (XC2V3000) 上に実装した。論

理合成した結果を以下に示す。

表2 論理合成結果

| Use Device        | XC2V3000-FF1152       |
|-------------------|-----------------------|
| Number of Slices  | 4566 out of 14336 31% |
| Maximum Frequency | 53.4 MHz              |

表2より、開発したハードウェアの回路規模は約100万ゲートとなった。また、人工的な少数の訓練サンプル(10個)を用いて学習を行った結果、学習に必要なクロックサイクル数は1266サイクルであった。このときのクロック周波数は20MHzである。よって、学習に要した時間は63[μsec]である。また、PCで同サンプルを用いて学習を行った。PCの環境は、Pentium4 2.4GHz CPU、512MB メモリである。また、学習アルゴリズムはC言語を用いて実装を行い、標準ライブラリ関数である clock() 関数にて実行時間を計測した。この結果、学習に要した時間は0.84[sec]であった。この結果から開発したハードウェアによる学習は、ソフトウェアに対して大幅な高速化を実現した。

#### 5. 結論

現在の SVM を組み込み分野へ適用するために、SVM の学習・識別アルゴリズムをハードウェアで実現した。開発したハードウェアは FPGA 上に実装し、回路規模は約100万ゲートとなった。また、人工的な訓練サンプルを学習させた結果、PCと比較して大幅な高速化を実現した。

#### 参考文献

- [1] V. Vapnik, Statistical Learning Theory, Wiley, 1998.
- [2] R. Genov and G. Cauwenberghs, "Kerneltron: Support Vector Machine in Silicon", IEEE Trans. on Neural Networks, Vol. 14, No. 5, pp. 1426-1434, 2003.
- [3] D. Anguita, A. Boni, S. Ridella, "A Digital Architecture for Support Vector Machines: Theory, Algorithm and FPGA implementation", IEEE Trans. on Neural Networks - Special Issue on Hardware Implementations, Vol. 14, No.5, pp. 993-1009, 2003.
- [4] Cauwenberghs G. & Poggio T. "Incremental and decremental support vector machine learning" Fourteenth conference on Advances in Neural Information Processing Systems, NIPS(pp.409-415)
- [5] J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimum Optimization," Advances in Kernel Methods- Support Vector Learning, Cambridge MA: MIT Press, 1998, pp 185-208.