제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

SVM을 이용한 화자인증 시스템의 하드웨어 구현

황병희[+*], 최우용[+], 문대성[+], 반성범[+], 정용화[+], 정상화[*]
[+]한국전자통신연구원 정보보호연구본부
[*]부산대학교 컴퓨터공학과
e-mail : bhhwang97@pusan.ac.kr

The Hardware Implementation of Speaker Verification System Using Support Vector Machine

Byung-Hee Hwang[+*], Woo-Yong Choi[+], Daesung Moon[+], Sung Bum Pan[+],
Yongwha Chung[+], Sang-Hwa Chung*
[+]Information Security Division, ETRI
*Department of Computer Engineering, Pusan National University

요 약

최근 목소리를 이용하여 사용자를 인증하는 화자인증(speaker verification)에 대한 관심이 증가하고 있으며, 다양한 화자 인증방법 중에서 SVM을 적용한 방법이 다른 알고리즘에 비해 우수한 성능을 나타내고 있다. 그러나 SVM을 이용한 화자인증 방법은 복잡한 계산으로 인해 휴대폰 등 휴대기기에서 실시간 처리에 어려움이 있다. 본 논문에서는 SVM을 이용한 화자인증 알고리즘을 실시간으로 처리하기 위한 하드웨어 구조를 제안하였고, VHDL을 이용하여 모델링 후 실험한 결과를 분석하였으며 전체 시스템 구성에 대하여 설명하였다.

1. 서론

정보통신 기술이 급속도로 발전하고 인터넷의 이용이 확산됨에 따라 사용자 인증에 대한 관심이 높아지고 있다. 최근까지 사용자 인증 수단으로 많이 사용되던 패스워드나 PIN(Personal Identification Number) 등은 타인에게 노출되거나 잊어버리는 등의 문제점을 가지고 있어 이를 대체하거나 보완하기 위한 방법으로 개인의 고유한 생체정보를 이용한 사용자 인증 방법에 관한 연구가 진행되고 있다. 이러한 생체인식 방법 중에서 사람의 음성을 이용하는 화자인식은 입력장치로 비교적 값이 싸고 손쉽게 구할 수 있는 마이크를 사용하며, 다른 생체인증방법에 비해서 사용자의 거부감이 적다는 장점이 있어 실생활에 활발하게 적용되고 있다.

전통적으로 많이 사용되어온 화자인증 방법으로는 DTW(Dynamic Time Warping), HMM(Hidden Markov Model), VQ(Vector Quantization), GMM(Gaussian Mixture Model), SVM(Support Vector Machine) 등이 있다[1-6].

본 논문에서는 다양한 데이터베이스를 이용하여 성능을 평가한 결과, 우수한 성능을 나타낸 SVM을 적용한 화자인증 방법[6]의 하드웨어 구현에 대하여 설명한다. 구현한 하드웨어 시스템은 사용자 인증을 위한 SVM 알고리즘의 하드웨어 구조를 설계하였고, 이를 VHDL로 모델링하여 FPGA에 탑재하여 개발한 시스템이 실시간으로 화자인증을 수행함을 확인하였다.

본 논문의 구성은 2장에서 SVM을 이용한 화자인증 알고리즘을 설명하고, 3장에서 구현된 하드웨어 구조에 대하여 살펴본다. 4장에서는 실험 결과에 대해 분석하고, 마지막으로 5장에서는 결론 및 향후 과제를 제시하였다.

2. SVM을 이용한 화자인증 시스템

SVM의 목적은 두 class를 분류하는 hyperplan을 설계하는 것으로, structural risk minimization 기법에 그 기초를 두고 있다[7]. 훈련데이터가 다음과 같이

주어졌다고 가정하자.

$$(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N) \in \Re^d \times \{\pm 1\} \quad (1)$$

여기서 $\mathbf{x}_i$ 는 input pattern이고, y_i 는 target output이다. 만약 두 class가 linearly separable하다면 식 (2)의 hyperplane에 의해서 두 class를 구분할 수 있다.

$$\mathbf{w}^T\mathbf{x} + b = 0 \quad (2)$$

이러한 hyperplane 중에서 hyperplane과 가장 가까운 데이터 포인트와의 거리를 최대로 하는 hyperplane을 optimal hyperplane이라고 하고, optimal hyperplane과 거리가 가장 가까운 데이터를 support vector라고 한다. SVM은 이러한 support vector를 이용하여 optimal hyperplane을 나타내는 방법으로 식(3)과 같은 constrained optimization 문제를 풀면 식(4)와 같은 해를 구할 수 있다[7].

$$\begin{aligned} &\min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, N \end{aligned} \quad (3)$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \quad (4)$$

여기서 α_i 는 Lagrange multiplier이다.

지금까지는 input space에서의 데이터들이 linearly separable한 경우에 대해서 살펴보았는데, input space에서의 데이터들은 linearly separable하지 않은 경우에는 input space에서는 문제를 풀 수가 없고, 더 높은 차원의 공간(feature space)으로 변환해서 문제를 해결해야 한다. 이때 사용하는 함수가 kernel function이다. Kernel function를 이용하면 식(5)와 같은 optimal hyperplane을 얻을 수 있다.

$$\sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) = 0 \quad (5)$$

Kernel Function에는 여러 가지가 있으나, SVM에서 많이 쓰이는 kernel function에는 다음과 같다.

- Polynomial learning machine

$$k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T\mathbf{x}_i + 1)^p \quad (6)$$

- RBF (Radial Basis Function) network

$$k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right) \quad (7)$$

- Two-layer perceptron

$$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\beta_0 \mathbf{x}^T\mathbf{x}_i + \beta_1) \quad (8)$$

SVM을 이용한 화자 인증 시스템을 설명하면 다음과 같다. 먼저 인증을 위해 많은 화자 data들을 vector($\mathbf{x}_i$)로 입력 받는다. 여기서 입력된 vector의 수를 N이라 하고 train될 사람의 ID는 k라 하자. k의 vector이면 y_i 는 1, 그렇지 않으면 -1을 입력한다. 이 값은 desired data이다. 입력 받은 vector($\mathbf{x}_i$)들은 새로운 N*N matrix를 계산하는데 쓰여진다. Matrix를 생성하기 위해서는 Kernel Function을 이용한 계산이 필요하다. 여기서 쓰여진 Kernel Function은 Polynomial Learning Machine이다. 생성된 matrix에 Gauss Jordan Algorithm[8]을 적용하면 최종적으로 계산된 desired data의 값을 구할 수 있는데 이 값이 Lagrange multiplier이다. Lagrange multiplier가 0보다 크면 그 input vector는 Support Vector(SV)가 되고 여기서 구해진 SV들로 SV table을 구성한다. ID인 k와 그 SV들을 table에 입력하는 것이다. 인증을 위해서는 k의 입력 vector와 앞서 구해진 table을 통해 SV를 얻어온 후 Kernel Function을 이용한 계산한 결과값을 가지고 인증을 하게 된다.

3. 하드웨어 시스템 구조

그림 1에서 나타낸 것과 같이 제안한 하드웨어는 크게 Train VMM(Vector Matrix-Multiplication)[9] Controller, Test VMM Controller, Gauss Jordan Controller, Support Vector Table Controller의 네 가지 논리적 단위로 구성되어 있다. Train VMM Controller는 등록된 vector data (표1 참조)의 주소를 Kernel Function Module에 입력으로 준다. Kernel Function Module은 입력된 주소정보로 vector data를 차례로 읽어온 후 Float Calculator를 통해 계산을 수행한다. 결과값이 반환되면 Train VMM Controller는 matrix를 구성한다. Gauss Jordan Controller는 Gauss Jordan Algorithm을 하드웨어로 구현한 모듈로서, Big Finder, Swap, Big row, Matrix Calculator로 구성되어 있다. 각각의 Module을 순서대로 반복하여 수행하게 되면 Lagrange multiplier (표2 참조)를 구할 수 있다. Support Vector Table Controller는 구해진 Lagrange Multiplier로 SV를 알고 이를 table (표3, 표4 참조)로 구성하는 역할을 담당한다. Test VMM Controller에서는 구해진 table과 test vector를 사용하여 인증을 위한 계산이 수행되도록 관리하는 역할을 담당한다.

3.1. n차원의 vector계산을 위한 control

실험에 사용될 vector의 차원은 가변적이다. Main Controller에서는 이를 알고 계산이 가능하도록 해야 한다. 이를 위해서 실행하기 전에 SRAM의 0000h ~ 0001h에 저장되어 있는 환경변수를 읽어온다. 본 논문에서 구현된 하드웨어 시스템은 시작 시에 이 환경변수를 읽어와 vector의 총 수, train될 사람의 vector의 수 그리고 vector의 차원을 먼저 알게 된다. Train VMM Controller나 Test VMM Controller에서는 얻은 정보를 통해 vector의 차원과 총 개수 등이 가변적이라 하여도 계산을 정확히 수행할 수 있도록 한다.

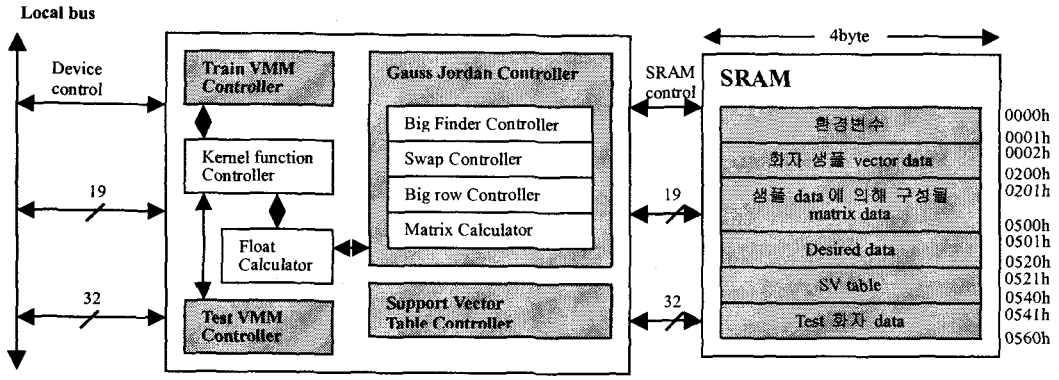

그림 1. 전체 블록 다이어그램

3.2. Kernel Function Controller

본 논문에서 사용된 kernel function은 Polynomial Learning Machine이다. Polynomial은 2로 구현되었다. Kernel function은 Train VMM Module과 Test VMM Module에서 새로운 matrix의 구성과 test vector의 인증을 위한 계산시에 사용된다. 주 계산은 vector의 내적이다. N 차원의 vector 내적 계산을 위해서는 부동소수점의 사칙연산을 수행할 수 있어야 한다.

3.3. Lagrange Multiplier

Support Vector를 구하기 위해 식(4)에서 α_i 인 Lagrange Multiplier를 계산해야 한다. Gauss Jordan Controller가 이 역할을 담당한다. Train 과정을 마치게 되면 Gauss Jordan Module이 동작하게 한다. SRAM의 0002h ~ 0200h에 저장된 화자 sample data가 Train VMM의 동작을 통해 0201h ~ 0500h에 새로운 matrix data로 생성되어 쓰여진다. 생성된 matrix를 이용하여 Big Finder를 통해 최대값을 가지는 data를 찾은 후 SWAP module에서 최대값의 row, column에 대응하는 row의 vector들을 서로 바꾸는 역할을 수행한다. Big Row Controller는 최대값의 row에 대한 계산을 수행하고 Matrix Calculator에서는 나머지 row에 대한 계산을 수행한다. 이 과정을 N번 거치게 되면 Gauss Jordan Module에 대한 동작은 완료되고 SRAM의 0501h ~ 0520h에서 Lagrange Multiplier를 구할 수 있다.

3.4. Support Vector Table의 구성

Gauss Jordan Module이 동작을 마치면 matrix data뿐만 아니라 desired data의 계산도 모두 끝나게 된다. 이 값이 각 vector의 Lagrange Multiplier이다. SV Table Controller에서는 SRAM의 0501h ~ 0520h에 저장되어 있던 Lagrange Multiplier의 값을 읽어와 32번째 bit가 0인 data에 대응하는 sample vector를 Support Vector라 둔다. Support Vector가 되는 sample data의 시작주소와 그 data의 Lagrange Multiplier로 먼저 <표 3>와 같은 SV table1을 구성한다. SV Table Controller는 다시 train ID, SV table1의 시작주소 그리고 SV의 개수를 가지는 <표 4>와 같은 SV table2를 구성하여 SRAM의 0521h ~ 0540h 사이에 저장한다.

3.5. Test

Test의 동작은 Test VMM Controller에 의해 수행된다. Test VMM Controller는 <표4>와 같은 SV table 2에서 ID에 대응하는 SV의 개수와 SV table 1의 시작주소를 읽어와 SV table 1에 접근하여 SV의 주소를 알아낸다. 알아낸 SV 주소와 test vector 주소로 Kernel Function을 동작시키고 식(5)와 같은 인증을 위한 계산을 수행한다.

4. 실험 결과

SVM을 이용한 화자 인증 시스템 구현을 위하여 PCI I/F를 이용하여 호스트와 통신하며 ARM9, Xilinx FPGA와 SRAM 모듈로 구성된 시스템을 구축하였다.

실험은 화자 sample vector들을 가지고 Support Vector를 찾아내고 table을 제대로 구성하는지를 검증하였으며, test vector를 입력시켜 인증결과가 올바르게 출력되는지를 확인하였다. 실험 vector는 표1과 같이 3차원 vector 5개이다. A, B의 desired는 1, 나머지는 -1이다.

<표1. Train vector data>

Vector	(1)	(2)	(3)
A	0.010987	0.050896	-0.063688
B	0.010074	-0.006244	-0.05714
C	0.023489	0.302097	-0.220871
D	0.0732	0.297585	-0.209109
E	-0.071338	0.279824	-0.137406

<표2. Support Vector와 Lagrange multiplier>

Support Vector	Lagrange Multiplier
A	2572.574951
C	1226.763184
D	708.179932

소프트웨어로 구현한 결과와 비교한 결과, 제대로 된 table을 구성했음을 확인할 수 있다.

<표3. SV table1>

주소	SV 주소	Lagrange Multiplier
0521h	0002h	2572.574951
0523h	000Bh	1226.763184
0524h	000Eh	708.179932

<표4. SV table2>

ID	Table1시작 주소	SV 개수
1	0521h	3

다음은 test결과로 A, B를 Test Vector로 입력했을 경우 결과값이 0보다 크므로 인증됨을 확인할 수가 있다.

<표5. test 결과>

Test Vetor	결과
A	567.902049
B	620.962022

시간 측정은 소프트웨어로 구현한 결과와 VHDL을 이용하여 하드웨어 모델링 후에 시뮬레이션 한 결과를 비교하였다. 소프트웨어의 시간 측정은 Compile 후 나오는 어셈블러를 분석하여 추정하였다. 여기서 소프트웨어는 최상으로 동작한다고 가정하여 각 instruction당 1 클럭 걸린다고 하고 PCI를 통해 동작하므로 클럭의 주기는 30ns로 둔다. 어셈블러를 분석한 결과는 표6과 같다.

<표6. 어셈블러 라인수>

Part	라인수
Train VMM Part	374
Matrix Part	305
Others	3091

Train VMM Part에서는 Matrix를 생성하기 위한 vector의 계산으로 인해 15번의 loop를 반복해야 하고 Matrix에서는 각 row에 대해 25번의 계산을 해야 한다. Vector의 개수가 5개 이므로 row의 개수도 5개이다. 그러므로 Matrix Part는 125번의 loop를 돈다고 볼 수 있다. 시뮬레이션의 시간을 측정한 것과 소프트웨어의 시간 예측은 다음 표와 같다.

<표7. 시간 측정 및 예측 결과>

	시 간
하드웨어 시뮬레이션 결과	0.528ms
소프트웨어 성능 예측	((374*15)+(305*125)+3091)*30ns = 1.405ms

표7에서 나타난 바와 같이 본 논문에서 구현한 하드웨어 시스템은 소프트웨어보다 빠른 성능을 보임을 알 수 있다. 그뿐 아니라 SVM을 적용한 화자 인증 방법은 곱셈 연산이 아주 많다. ARM920T의 경우 곱셈을 하기 위해서는 보통 6~7 클럭이 걸린다. 그러나 본 시스템은 5 클럭으로 설계 하였다. 이와 같이 부동소수점의 사칙연산 수행 사이클 비교를 통해서도 본 하드웨어 시스템이 소프트웨어보다 빠르게 동작한다고 추정할 수 있다.

5. 결론 및 향후과제

본 논문에서는 SVM을 이용한 화자인증 알고리즘의 하드웨어 구현에 대하여 설명하였고 구현한 하드웨어가 정상적으로 동작하는지 확인하였다. 시뮬레이션 결과를 통해서는 초당 약 220명을 처리할 수 있음을 확인했다. 또한, 제안한 시스템은 화자 data 뿐만 아니라 인증이 필요한 다른 신체적 특징, 즉 지문이나 홍채 등의 data인 경우라 하더라도 특징 정보가 vector로 표현될 수 있다면 사용이 가능하다. 현재, FPGA에 SVM알고리즘을 탑재하기 위해 시뮬레이션 작업이 계속 진행 중이다. 앞으로는 본 하드웨어의 속도 향상을 위해 VMM의 design에 있어 병렬적 처리와 FPGA의 게이트수를 줄여 집적도면에서 향상을 보일 수 있도록 하는 연구가 진행될 것이다.

참고문헌

[1] Joseph P. Campbell, "Speaker recognition: a tutorial," *Proc. of the IEEE*, vol. 85, no. 9, pp. 1437-1462, Sep. 1997.

[2] Qi Li, Biing-Hwang Juang, Chin-Hui Lee, Qiru Zhou, Frank K. Soong, "Recent advancements in automatic speaker authentication," *IEEE Robotics and Automation Magazine*, pp. 24-34, Mar. 1999.

[3] Jialong He, Li Liu, Gunther Palm, "A New Codebook Training Algorithm for VQ-based Speaker Recognition," *Proc. of the ICASSP*, vol. 2, pp. 1091-1094, 1997.

[4] C. Martin del Alamo, F. J. Caminero Gil, C. de la Torre Munilla, L. Hernandez Gomez, "Discriminative training of GMM for speaker identification," *Proc. of the ICASSP*, vol. 1, pp. 89-92, 1996.

[5] Simon Haykin, *Neural Networks*, Prentice Hall, 1999.

[6] Woo-Yong Choi, *et. al.*, "Support Vector Machines for Robust Speaker Verification" *Proc. of the AICSST*, pp. 262-267, 2002.

[7] Bernhard Scholkopf, Christopher J. C. Burges, Alexander J. Smola, *Advances in Kernel Methods*, The MIT Press, 1999.

[8] George Nakos, David Joyner, *Linear Algebra*, Brooks/Cole, 1998

[9] Roman Genov, Gert Cauwenberghs "Charge-Mode Parallel Architecture for Vector-Matrix Multiplication", *Proc. of the IEEE*, vol. 48, NO. 10, Oct. 2001.