Thema

지능성 발현을 위한 CSVM 프로세서 개발

위재우 심사관
(특허청 전자상거래심사팀)
이종호 교수
(인하대 정보통신공학부)

1. 서 론

본 연구에서는 통합 기능을 가지는 고속의 Support Vector Machine (SVM) 프로세서를 개발하고 이를 여러 가지 문제에 적용하여 프로세서의 성능을 검증한다. SVM은 통계적 학습 이론에 기반하여 최적 분류를 함으로서 뛰어난 일반화 성능을 보이는 학습 방법이다. Local Minima에 강인하고 사용자가 초기에 설정해 주어야 할 파라미터가 적은 장점을 가지고 있어 Bioinformatics, 통신 등의 많은 분야에서 널리 사용되고 있다. SVM을 실시간 처리 등이 요구되는 문제에 적용하기 위해서는 통합형, 고속의 SVM 전용 프로세서가 절실히 요구되지만 기존의 SVM 프로세서는 SVM에 필요한 모든 연산들을 포함하지 못 하였다.

최근에 SVM을 하드웨어화하여 실시간 처리가 요구되는 분야에 적용하려는 시도는 크게 아날로그 구현과 디지털 구현 방법으로 진행되고 있다. 아날로그 SVM 하드웨어인 Kerneltron[1]은 내부적으로 아날로그 방식으로 설계하고 외부와는 디지털 신호로 변환하여 전달한다. 이 하드웨어는 빠른 커널 연산을 하여 물체 검출 등의 실시간 적용 분야에 초점을 맞추고 있지만 칩 안에서 학습까지 이루어지지 않아 재현 동작만 온라인으로 칩 상에서 이루어진다. 디지털 SVM 하드웨어인 디지털 SVM(DSVM)[2, 3]은 하드웨어에 적합한 학습 알고리즘을 제안하고 학습까지 이루어지는 디지털 SVM 하드웨어를 구현하였다. 하지만 이 디지털 하드웨어는 커널 연산 부분은 칩 상에서 동작하지 않음으로 인하여 시간 소모가 많은 커널 연산 과정에 의하여 처리 시간 상 병목 현상이 발생하게 되며 또한 칩만의 독자적 연산 수행(Stand-alone Operation)이 불가능하다는 단점이 지적된다.

본 연구에서 제안하는 SVM 프로세서는 SVM이 필요로 하는 커널 연산, 학습 연산, 재현 연산 회로를 모두 포함한다. 하드웨어 구현에 부적합한 Quadratic Programming (QP) 학습 방법 대신에 Kernel Adatron (KA)과 Kernel Perceptron (KP) 학습 알고리즘을 사용한다. 적용 문제에 따라서 프로세서 사용자는 최고의 성능을 위하여 학습 방법을 선택 사용 가능하다. 병렬로 설계된 기본 블록의 동시 수행은 빠른 속도를 가능하게 한

다.

양자화 오차가 발생하는 고정소수점 연산 실험을 하여 결과를 부동소수점 연산 결과와 비교하고 하드웨어 설계 사양을 결정하였다. 실험은 백혈병 환자 데이터의 분류를 수행하였고 비선형 통신 채널 등화 문제에도 적용하였다. KA를 사용하여 채널 등화 문제에 적용하여 QP 방법에 의한 결과와 비교하였다. 소프트웨어를 사용한 SVM 처리 시간이 샘플의 수에 따라 2차적으로 증가하는 반면, 하드웨어 설계 결과 처리 시간이 선형적으로 증가하여 많은 샘플을 가지는 문제에 대하여 상당히 처리 속도가 증가하는 것을 알 수 있었다. 하드웨어 성능인 처리 시간과 회로 면적에 있어서 기존의 프로세서와 비교하여 더 우수한 성능을 보였다.

2. Support Vector Machine

SVM은 기존의 신경회로망에서 이용된 경험적 위험을 최소화하는 원리보다는 구조적 위험을 최소화하는 근사적 방법이고 통계적 학습 이론에 기반하여 최적 분류를 함으로서 뛰어난 일반화성능을 보여 준다[4]. SVM은 커널 함수에 의하여 정의된 특징 공간에서 초평면을 경계로 학습 데이터를 분리한다. 학습 데이터 중에서 초평면과 가장 근접한 데이터들의 거리의 합인 마진이 최대가 되는 다차원 평면을 찾는다. 통계 학습 이론에 의하면 초평면의 일반화 성능은 속해 있는 공간의 차원이 아닌 마진에 의하여 결정되므로 고차원에서도 분류 능력이 뛰어나다.

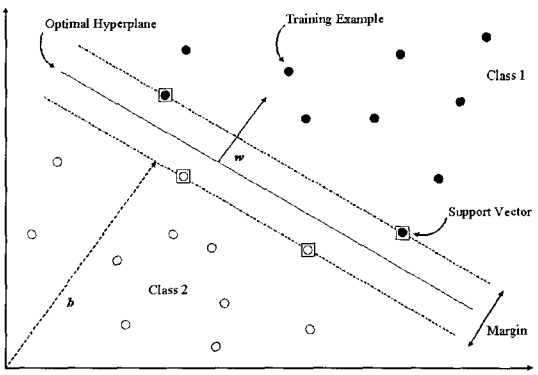

그림 1. 선형 최대 마진 분류기.

학습 데이터가 $(x_1, y_1), \cdots, (x_n, y_n)$이고 입력패턴 $x_i \in \Re^d$ (d: 입력 공간의 차원)에 대하여 두 개의 클래스 $y_i = \{+1, -1\}$로 분류되는 문제를 생각해 보면, 그림 1과 같이 결정 함수인 $f(x) = \text{sign}((w \cdot x)+b)$의 정규(Normal) 벡터 w와 바이어스 b는 마진을 최대화하도록 결정된다.

SVM은 데이터를 비선형 커널 함수를 이용하여 다른 내적 공간, 즉 특징 공간으로 사상하여 비선형적인 초평면을 만들어서 비선형 분류 문제를 해결한다. 비선형 커널 함수에 의하여 SVM은 식 (1)과 같은 비선형 함수를 학습하게 된다.

$$f(x) = sign[\sum_{i=1}^{n} \alpha_i y_i K(x \cdot x_i) + b] \qquad (1)$$

식 (1)에서 파라미터인 α는 선형적으로 제한적인 QP 문제에 의하여 결정된다. 식 (3)의 제한적 조건하에서 식 (2)를 최대화하는 α를 구하게 된다.

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad (2)$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \ (i = 1, 2, \ldots, N) \qquad (3)$$

s개의 Support Vector들이 존재한다면 b값은 식 (4)에 의하여 구할 수 있다.

$$b = \frac{1}{s} \sum_{j=1}^{s} \left(y_j - \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j) \right) \qquad (4)$$

위와 같은 방법으로 SVM의 파라미터를 구하는 QP 방법은 계산량이 많고 구현하기 어렵다. 본 연구에서는 QP 방법이 아닌 구현하기 쉬운 KA와 KP 학습 알고리즘을 사용한다.

그림 2는 SVM의 구조를 나타낸다. 입력 x와 Support Vector x_i는 커널 함수에 의하여 비선형으로

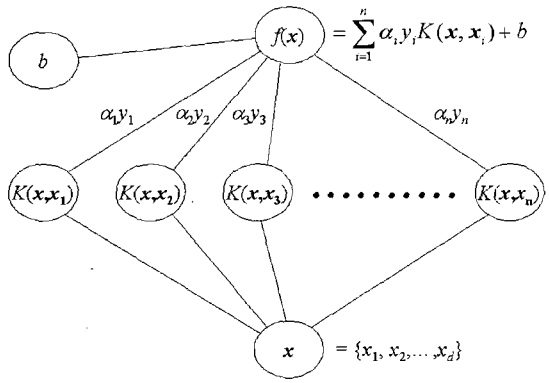

그림 2. SVM의 기본 구조.

사상된다. 최종적으로 계산된 f(x) 값의 부호에 따라서 입력 x의 클래스를 결정하게 된다.

2.1 Kernel Adatron

KA 알고리즘은 통계적인 학습 방법의 하나인 Adatron 알고리즘에 커널을 사용함으로서 비선형 Feature 공간에서 마진을 최대화하게 한 것이다. Adatron 알고리즘은 이론적으로 최적해로의 수렴이 알고리즘의 반복에 지수 함수적으로 빠르게 일어난다고 알려져 있다[5]. 결국 KA 알고리즘은 SVM의 특징공간에 Adatron을 도입하는 것으로 용이한 구현성 뿐만 아니라 커널에 의한 비선형 특징공간에서의 단순한 동작 특성도 동시에 얻을 수 있어, 기존 QP 학습 알고리즘이 가지는 계산과 구현상의 제약들을 효과적으로 해결할 수 있다.

표 1. KA 알고리즘.

{초기화} 모든 i에 대해서 $\alpha_i = 0$, 학습율 η을 세팅
{학습 루프} Repeat
for $i = 1, \ldots, n$

$$z_i = \sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j), \quad \Delta\alpha_i = \eta(1 - y_i z_i)$$

if $\alpha_i + \Delta\alpha_i > 0$ then $\alpha_i \leftarrow \alpha_i + \Delta\alpha_i$ else $\alpha_i \leftarrow 0$
end for
{종료 조건} until (최대학습회수에 도달하거나

$$r = \frac{1}{2}\left[\min_{\{i|y_i=+1\}}(z_i) - \max_{\{i|y_i=-1\}}(z_i)\right] \approx 1$$

2.2 Kernel Perceptron

Kernel Perceptron (KP) 알고리즘은 퍼셉트론의 w 영역으로부터 SVM의 α 영역으로 바꾼 것이다 [4]. KP 알고리즘은 Rosenblatt의 퍼셉트론 알고리즘과 마찬가지로 수렴성이 보장된다. KP 알고리즘을 정리하여 표 2에 나타내었다.

표 2. KP 알고리즘.

{초기화} 모든 i에 대해서 $\alpha_i = 0$, b = 0으로 세팅
{학습 루프} Repeat
for $i = 1, \ldots, n$

$$o_i = \sum_{j=1}^{n} \alpha_j q_{ij} + b, \quad q_{ij} = y_i y_j K(x_i, x_j)$$

if $o_i \le 0$ then update $\alpha_i \leftarrow \alpha_i + 1$, b = b + y_i
end for
{종료 조건} until (최대학습회수에 도달하거나 모든 i 에 대하여 $o_i > 0$)

2.3 복잡도 계산

SVM은 커널 계산, 학습, 그리고 재현 단계 동작을 수행하게 된다. $K(x_i, x_j) = exp(-1/2\sigma^2 \| x_i - x_j \|^2)$의 커널 함수를 사용하여 각 단계의 계산 복잡도를 표 3에서 나타내었다. 표 3으로부터 커널 단계와 학습 단계에 필요한 계산량이 더 많다는 것을 알 수 있다.

d = 2이고 커널 연산에 대하여 샘플의 수와 그에 필요한 곱셈 연산의 수와의 관계를 그림 3에 나타내었다. 곱셈 연산은 다른 연산에 비하여 가장 복잡도가 상대적으로 높기 때문에 이를 사용하여 비교하였다. 샘플의 수가 증가할수록 커널 계산을 위한 연산량, 즉 곱셈의 수가 학습단계를 위한 수보다 더 빨리 증가하는 것을 알 수 있다. 따라서 더 큰 샘플의 문제에 대하여 속도 증가 효과를 최대화하기 위하여

표 3. 각 단계에 필요한 연산량.

Number of Operations	Kernel Computation	Learning (KA)	Recall
Addition or Substitution	$(2d+1)n^2$	$n(n+1)$	$n-1$
Multiplication	$2dn^2$	$2n(n+1)$	$2n$
exp(•)	n^2	N/A	N/A
Comparison	N/A	n	N/A

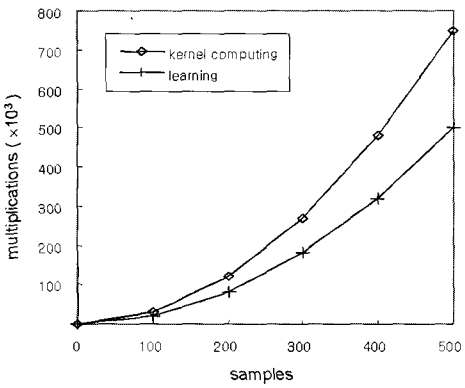

그림 3. 샘플의 수에 따라 필요한 곱셈 연산의 수.

커널 계산 단계에 필요한 회로를 CSVM 프로세서에 포함시키기로 결정하였다. CSVM은 한 칩에 모든 단계의 회로를 수행하는 최초의 SVM 프로세서이다.

3. CSVM 구조

3.1 주요 블록과 신호

CSVM 프로세서의 구조는 그림 4와 같다. 본 하드웨어는 SVM의 마진 상에 위치하는 Support Vector을 기본단위로 하여 Support Vector Element (SVE) 모듈을 병렬 처리하여 연산속도를 증가시켰다. SVE의 개수는 학습샘플의 개수이며 학습결과 0이 아닌 값을 갖는 SVE가 Support Vector가 된다. 학습샘플의 개수가 많을 경우는 칩을 확장하여 사용가능하게 설계되었다. 칩의 확장은 각 CSVM의 공유버스를 연결하고 출력선과 확장 입력선을 연결한다. 학습 단계를 위해서는 그림에서 가장 아래에 위치하는 하나의 CSVM의 Learning Element (LE)만 동작하게 된다.

모든 SVE 간은 공유버스로 연결을 시켜서 커널 연산이 동시에 이루어지게 하였다. KA와 KP 학습 알고리즘의 유사한 점을 이용하여 칩에 KA, KP 두 가지 학습방법을 함께 내장하였고 SVE를 이용하여 커널 계산, 학습과 재현 동작을 모두 함으로써 적은 면적에 효과적으로 설계하였다. LE는 학습 단계의 파라미터 갱신을 위한 계산 시에만 사용된다.

CSVM의 입력과 출력 신호는 그림 5와 같다. 데이터의 시작을 칩이 인식하도록 칩으로 data_in 신호를 최초로 넣을 때 Start신호를 1로 해 준다. 샘플이 상당히 많은 데이터를 분류할 때 m개의 CSVM 칩을 함께 사용하기 위하여 i-1번째 CSVM의 data_out 출력 신호를 i번째 CSVM의 data_ext 핀을 통해서 입력을 받고 두 개의 CSVM의 공유 데이터 버스 입출력핀인 data_to_from_bus를 서로 연결해 준다.

Mode 입력 신호는 칩의 동작을 결정한다.

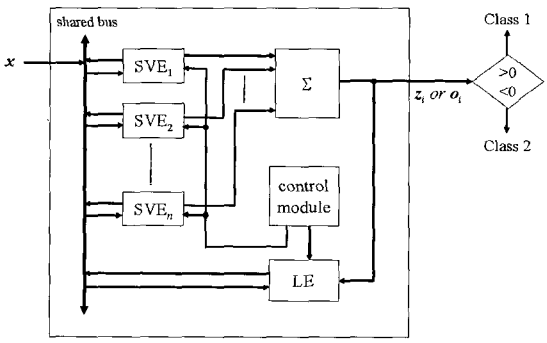

그림 4. CSVM 구조.

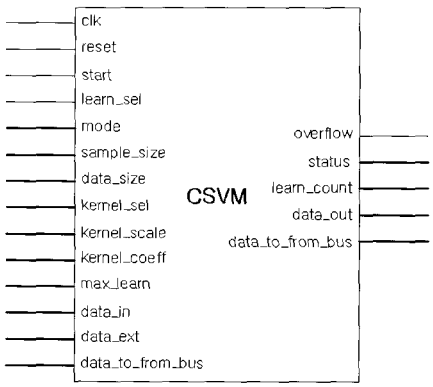

그림 5. CSVM의 입·출력 신호.

mode=0에서는 학습 동작만 이루어지고 mode=3에서는 학습이 종료된 후 재현 동작도 연속으로 이루어진다. 재현 동작만을 할 경우, 칩 내에 이미 저장되어 있는 파라미터를 이용할 때는 mode=1을 설정하면 되고 외부에서 파라미터를 입력 받아 사용하고자 할 때는 mode=2로 설정한다.

출력 신호 Status 신호는 현재 칩의 상태를 사용자에게 알려 준다. status=0은 대기상태, status=1은 학습 동작 상태를 나타내며, status=2는 재현 동작 상태, status=3은 동작 종료 상태를 알려주게 된다.

칩의 동작은 로딩, 커널 계산, 학습, 재현 단계로 나누어진다.

(1) 로딩 단계: 외부로부터 x, y를 로드하는 단계이다. 한 샘플에 해당하는 x, y는 공유 버스를 통하여 순차적으로 각 SVE의 내부 메모리로 저장된다. 이 동작은 칩의 입력 핀의 수가 한정되어 있으므로 병렬적인 수행이 불가능하고 순차적으로 이루어지므로 하드웨어 구조에 관계없이 샘플의 수에 따라서 일정한 로딩 시간이 소요된다.

(2) 커널 계산 단계 : SVE의 커널 계산부를 동시에 (Concurrent) 수행하도록 하여 커널 계산에 걸리는 시간을 상당히 감소시켰다. 하나의 SVE_i에서 x_i를 공유버스로 보내면 모든 나머지 SVE_j $(j \neq i)$들이 이를 동시에 받아 자신의 x_j와 커널 함수 $K(x_i, x_j)$를 계산하고 이를 내부 메모리에 저장하게 된다. 이 동작은 병렬적으로 수행됨에 따라 순차적인 커널 계산 방법보다 속도가 이론적으로 n배만큼 빨라지게 된다.

(3) 학습 단계: KA 또는 KP 학습 알고리즘을 이용하여 마진을 최대화하는 α와 b를 찾는 과정이다. 이 과정은 SVE와 LE 모두를 사용되게 된다. 이전 동작인 커널 계산 단계에서 계산한 $K(x_i, x_j)$를 메모리에서 읽어서 각 SVE는 KA학습인 경우, $z_{ij} = \alpha_j y_j K(x_i, x_j)$의 연산을 하게 되고, KP학습인 경우는 $oij = \alpha_j y_i y_j K(x_i, x_j)$ 연산을 한다. 그 후에 모든 SVE는 모든 SVE로부터 계산된 z_{ij} 값이나 $(o_{ij} + b)$ 값을 더하여 z_i 또는 o_i 값을 얻게 된다. 그 후 학습 알고리즘의 갱신 조건에 해당하는 연산을 학습부에서 하게 되고 LE에서 계산된 새로운 α와 b는 공유버스를 통하여 해당 SVE의 메모리로 보내져서 종료조건이 만족될 때까지 위에서 기술한 학습 과정을 반복하게 된다. 이 과정에서도 z_{ij} 또는 o_{ij}를 얻기 위하여 병렬적인 연산이 이루어지므로 그만큼 학습 속도가 향상되게 된다.

(4) 재현 단계: 학습에서 얻은 파라미터 α와 b, 그리고 α가 0이 아닌 SVE에 저장된 x값을 이용하여 입력된 테스트 데이터 x_t의 클래스를 결정하게 된다. 테스트 샘플에 대한 마진 값 m_t는 식 (5)에 의하여 계산된다.

$$m_t = \sum_{j=1}^{n} \alpha_j y_j K(x_t, x) + b \qquad (5)$$

마진 값이 양수이면 1계열로, 음수이면 2계열로 판정한다. 이 단계에서는 커널 연산이나 학습 연산에서 사용한 회로를 그대로 재사용하므로 회로 면적을 그만큼 절약하게 된다. 재현 단계에서도 SVE의 병렬 동작으로 순차적 동작보다 이론적으로 n배 빠른 연산을 하게 된다.

3.2 각 블록의 구조

3.2.1 Support Vector Element

SVE는 그림 6과 같이 공유버스 인터페이스부와 메모리부, 식 (6)의 연산을 하는 부분으로 이루어진다. SVE의 출력 신호인 z_{ij}와 o_{ij}는 식 (6)에 의하여 계산된다.

$$z_{ij} = \alpha_j y_j K(x_i, xj)$$
$$o_{ij} = \alpha_j y_i y_j K(x_i, x_j) \qquad (6)$$

z_{ij}는 KA학습 또는 재현 동작에 사용되고 o_{ij}는 KP 학습 동작에서만 사용되므로 KA학습 또는 재현 동작에서는 y_i 곱셈 연산은 이루어지지 않고 Bypass된

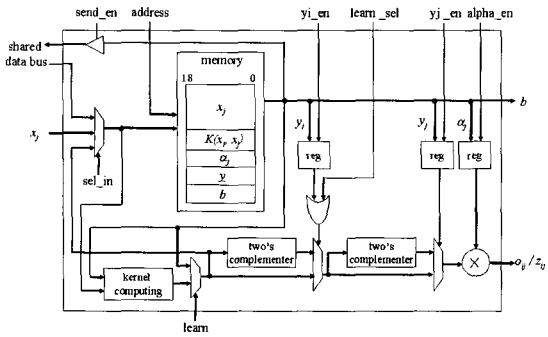

그림 6. Support Vector Element.

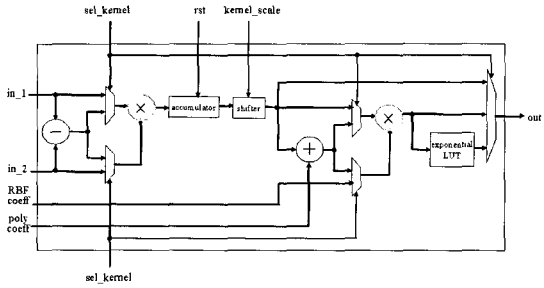

그림 7. 커널 계산부.

다. y_i값은 1 또는 -1 값이기 때문에 곱셈기 대신에 보수 변환기(Two's Complementer)를 사용하여 면적을 절약하였다. 재현 단계에서는 커널 계산부(Kernel Computing Element)에서 신호를 받아서 y_i를 곱하게 되고 학습 단계에서는 메모리에서 이미 계산되어 저장된 커널 값을 받아서 y_i 값을 곱하게 된다.

메모리에 저장되어있는 x_j와 y는 로딩 단계에서 저장되고 커널 연산 단계에서 $K(x_i, x_j)$가 계산되어 메모리의 해당부분에 저장된다. α와 b는 매회 학습마다 갱신되어서 메모리에 저장된 후 각 해당 레지스터에 저장된다. 하나의 메모리 출력 포트로부터 여러 파라미터를 레지스터로 보내야 하기 때문에 각각의 레지스터들은 해당 파라미터가 메모리로부터 도착할 때만 해당 yi_en, yj_en, alpha_en 신호를 1로 하여 저장한다. 원하지 않는 신호가 도착했을 때 저장하지 않고, 이전 값을 유지하기 위하여 해당 yi_en, yj_en, alpha_en 신호를 0으로 설정한다.

공유버스 인터페이스부의 입력부분에서 입력은 해당 동작에 따라서 공유버스 값, 외부 입력 값, 메모리에 저장될 커널 연산 값 중에 선택된다. 인터페이스부의 출력부분은 3상태 게이트를 이용하여 공유버스에 두 개 이상의 데이터가 올려지는 것을 막아주게 된다. 데이터를 보내고자 하는 SVE는 send_en=1로 하여 신호를 보내게 되며 한꺼번에 두 개의 SVE에서 신호를 보낼 수 없도록 제어부에서 각 SVE의 send_en을 통제한다.

3.2.2 커널 계산부

커널 계산부는 커널 계산 단계와 재현 단계에서 사용되고 학습 단계에서는 커널 계산 단계에서 계산되어 메모리에 저장된 커널값을 사용한다.

커널 함수는 선형, 다항 함수, RBF 커널을 그림 7과 같이 내장하였다. RBF 커널 함수 $\exp(-1/2\sigma^2 \| x_i - x_j \|^2)$에서 $1/2\sigma^2$ 항의 연산에 필요한 나눗셈 연산은 회로 구현 시 많은 면적과 연산 시간이 소모된다. 따라서 $1/2\sigma^2$ 항을 하나의 외부입력 파라미터 c로 받아들여 더 경제적인 설계를 하였다.

커널 함수 모듈은 면적을 최소화하기 위하여 곱셈기 2개만으로 3종류의 커널 함수를 설계하였다. 그림 7에서 누산기(Accumulator) 앞에 위치하는 첫 번째 곱셈기에서는 선형, 다항 함수의 $x \cdot x_i$ 연산을 하고 RBF 함수의 $(x-x_i) \cdot (x-x_i)$ 연산을 한다. LUT 앞에 위치하고 있는 두 번째 곱셈기에서는 다항 함수의 $(x \cdot x_i) \cdot (x \cdot x_i)$ 연산을 하고 RBF함수의 $c \cdot \| x_i - x_j \|^2$ 연산을 한다. RBF 연산에서 두 번째 곱셈기의 출력값은 지수함수 LUT로 통과하게 된다. LUT의 입력비트의 크기에 비례하여 메모리 공간을 차지하므로 실험을 통하여 입력비트의 크기를 결정하였다.

고차원의 데이터의 커널 계산 시 커널 값이 너무 커질 수 있으므로 커널 스케일링 방법을 사용하여 커널 계산 값의 범위를 줄여서 데이터 비트의 정수

부분을 효과적으로 절약시켰다.

3.2.3 학습부

학습부는 커널 값을 사용하여 모든 SVE에서 계산되어서 더해진 o_i 또는 z_i 을 받고 갱신되어야 할 SVE_i의 메모리에 저장되어 있는 파라미터인 a^{old}, b^{old}, y_i 값을 공유버스를 통하여 받는다. 공유버스로부터 온 값들을 load 신호에 따라 a^{old}, b^{old}, y_i 레지스터에 차례대로 저장한다. Kernel Adatron Learning Element (KALE)와 Kernel Perceptron Learning Element (KPLE)에서 계산되어 갱신된 a^{new}, b^{new} 값은 send_en 신호가 활성화되면 공유버스를 통하여 SVE_i의 메모리에 저장된다. 학습 종료 조건이 만족하면 학습 단계는 끝나게 되고 최종 파라미터 a와 b가 모든 SVE의 메모리에 저장되게 된다.

4. 실험결과

우리의 하드웨어의 성능을 입증하기 위하여 백혈병 질병 진단 문제, 비선형 채널 등화 문제에 적용하여 연산시간과 성능을 평가한다.

4.1 Leukemia Data 실험결과

백혈병 데이터[6]는 DNA 마이크로어레이칩으로부터 얻은 유전자 발현 양상 데이터이다. 이 데이터는 72명의 환자에 대한 각 7129개의 유전자로 이루어져 있다. 72명의 환자 샘플 중에 38명의 데이터는 학습을 위해서 사용되었고 나머지 34명의 데이터는 테스트용으로 사용되었다. 급성 림프구성 백혈병(ALL, Acute Lymphoblastic Leukemia)과 급성 골수성 백혈병(AML, Acute Myeloid Leukemia)의 두 가지 클래스로 분류된다. 전처리 과정 [7]을 통해서 얻은 3571개의 유전자를 실험에 사용하였다.

KP 알고리즘과 RBF커널 함수를 사용한 부동소수점 알고리즘 실험을 통해서 단지 1개의 테스트 패턴을 오분류하였다. 표 4는 고정소수점 알고리즘을 사용하여 다양한 데이터 폭과 커널 스케일율에 따라 얻은 테스트오차의 수를 나타내고 있다. 데이터의 소수 부분은 12비트로 고정하였다. 데이터 폭이 14

표 4. 다양한 데이터 폭과 커널 스케일율에 대한 백혈병 데이터의 테스트 오차 개수.

데이터 폭	커널 스케일율				
	1	4	16	64	256
14	14	15	17	13	1
16	20	18	17	1	1
18	20	17	1	1	1
20	16	1	1	1	1

비트로 감소할지라도 256의 비율로 커널 스케일을 한다면 테스트 오차는 1로 감소한다. 데이터의 차원이 커진다면 커널 스케일율을 크게 정해주어야 좋은 결과를 얻을 수 있다.

4.2 비선형 채널 등화 실험결과

첫 번째 실험 모델은 비선형 채널 $\hat{x}(n) = \tilde{x}(n) - 0.9\,\tilde{x}^3(n)$, $\tilde{x}(n) = u(n) + 0.5\,u(n-1)$이다. 부가되는 백색 잡음의 분산 $\sigma_e^2 = 0.2$, M = 2, RBF 커널의 파라미터 $\sigma^2 = 0.5$이다. $D = 0$일 경우를 모델 1, $D = 2$일 때를 모델 2라고 명명하도록 한다. 결과는 10번 실험 후에 평균값을 취하고 500개의 샘플로 학습을 하고 2400개의 샘플로 테스트를 한다.

Anguita *et al.* [2]의 실험 결과와 비교하여, KA의

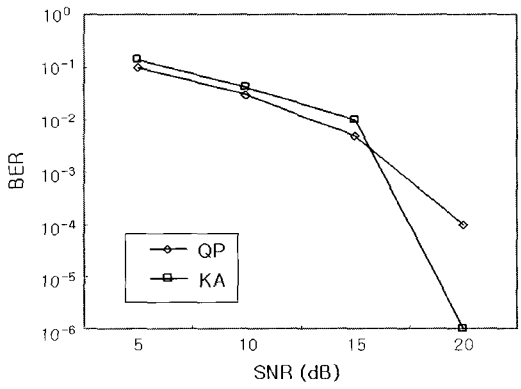

그림 8. KA 학습 알고리즘과 QP 방법을 사용한 실험 결과의 평균 BER.

결과가 우월하다는 것을 알 수 있었다. KA 학습을 사용한 CSVM은 모델 1에 대하여 17.75 %와 모델 2에 대하여 3.95 %의 오차율을 얻는다. 모델 2에 대한 결과는 Anguita *et al.* [2]의 논문에서 실험한 4.2 %을 능가하였고 모델 1의 결과는 유사했다.

두 번째 실험은 RBF 커널의 KA학습을 사용할 경우와 QP 방법을 사용할 경우에 대하여 SNR에 따른 비트 에러율(Bit Error Rate, BER)을 비교하였다. 채널 모델은 $\hat{x}(n) = \tilde{x}(n) + 0.2\,\tilde{x}^2(n)$, $\tilde{x}(n) = 0.3482\,u(n) + 0.8704\,u(n-1) + 0.3482\,u(n-2)$을 사용하였고 파라미터는 $M=3$, $D=1$, $\sigma^2=1$로 설정하였다. 학습 샘플수는 500개를 사용하였고 테스트 샘플은 2000개를 사용하여 10번 실험하였다. 그림 8과 같이 5-10 dB 범위에서는 KA 알고리즘과 QP 방법을 사용하여 학습한 Sebald *et al.* [8]의 논문의 결과와 유사했다. 15 dB 이상의 범위에서는 KA의 성능이 QP보다 더 우월했다. RBF 커널을 사용한 학습은 Sebald *et al.* [8]의 논문에서 사용한 다항 커널을 사용한 경우보다 더 좋은 결과를 보이는 것을 알 수 있었다.

그림 9와 10에서는 부동소수점과 고정소수점을 사용한 결정 평면을 보이고 있다. 이 실험에서는 Anguita *et al.* [2]의 논문에서 사용한 32개의 학습샘플과 2900개의 테스트샘플의 모델 2에 대하여 수행되었다. 그림 10에서 고정소수점 연산모델의 결정 평면이 요동치는 경향을 알 수 있고 이로 인해 경계선에 가까운 테스트 샘플들은 오분류되어지고 있다. 고정소수점 연산모델의 데이터폭이 적어질수록, 결정 평면은 더 요동치고 결과들은 불안정해진다.

오차와 데이터폭과의 관계가 그림 11에 나타냈다. 데이터의 소수자리 비트폭이 12비트 미만에서는 비트폭에 따라 성능의 차이가 컸지만, 12비트 이상일 때에는 성능의 차이가 거의 발생하지 않음을 알 수 있었다. 따라서 등화기 문제에서는 데이터의 소수자리 비트폭을 12비트로 설계하였다.

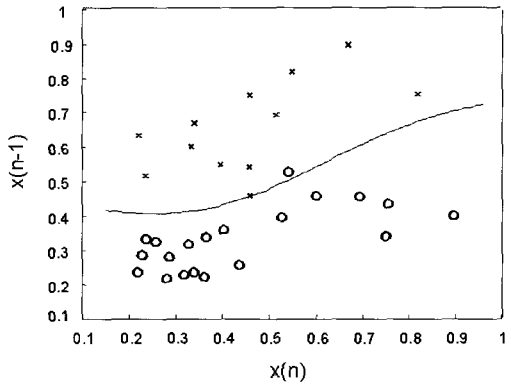

그림 9. 부동소수점 연산모델을 사용한 결정 평면.

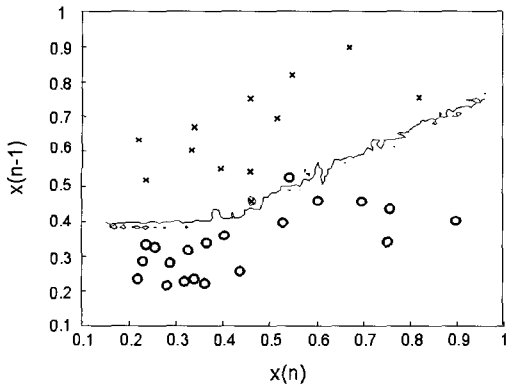

그림 10. 고정소수점 연산모델을 사용한 결정 평면.

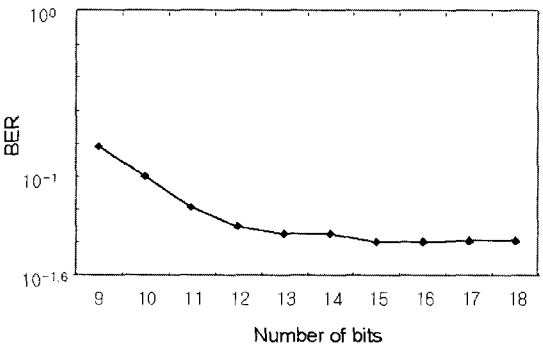

그림 11. 데이터의 소수자리 비트폭에 따른 비트오차율.

5. CSVM 하드웨어 구현 및 성능

고정소수점 실험 결과에서 얻은 하드웨어 설계 명세를 사용하여 실제 하드웨어를 설계하고 합성,

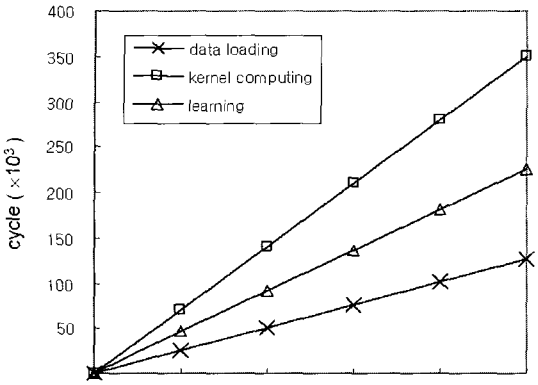

그림 12. 다른 샘플수에 따른 처리 시간.

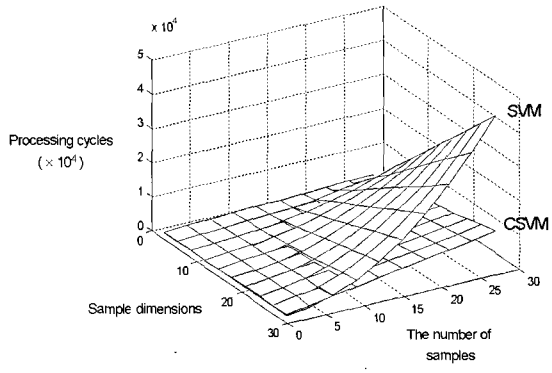

그림 13. 하드웨어를 사용하지 않은 SVM과 CSVM 프로세서의 연산 사이클 곡면.

타이밍 시뮬레이션을 수행하였다. 하드웨어 구현 결과를 기존의 하드웨어와 수행시간과 회로면적의 성능을 비교하였고 이를 분석하였다.

최종적으로 실제 FPGA에 CSVM을 다운로드하고 PCI 인터페이스를 통해 호스트 컴퓨터에서 CSVM의 성능을 검증하였다.

CSVM은 Verilog HDL로 설계되었으며 Xilinx Virtex2 FPGA XC2V6000 [9]에 구현하였다. XC2V6000 FPGA는 6백만 게이트들을 구현 가능하고 144개의 18비트 전용 곱셈기와 144개의 18,000 비트 독립적인 메모리 블록(BRAM)을 사용할 수 있다.

그림 12는 프로세서를 이용하였을 때, 샘플의 수에 따른 각 단계 동작의 소요 시간이다. 그림 3의 소프트웨어 모델에서는 샘플의 수에 따라서 연산량이 지수적으로 증가하는 반면, 프로세서를 이용한다면 선형적으로 증가하는 것을 알 수 있다. 이 그림으로부터 커널 동작 등을 병렬 연산을 이용하는 CSVM을 사용할 때 큰 샘플을 가진 문제에 얼마나 효과적으로 사용할 수 있는지를 보여준다.

그림 13에서는 샘플 차원과 개수에 따른 연산 사이클의 변화 곡면을 나타냈다. 상단의 곡면은 하드웨어를 사용하지 않은 SVM 알고리즘의 사이클 증가 곡면이고 하단의 곡면은 CSVM 프로세서를 사용하였을 때의 곡면이다. 이 그림으로부터 샘플의 수가 더 커진다면 두 곡면은 점점 더 상당한 차이를 보일 것이라는 것을 예상할 수 있다.

표 5에서는 DSVM과 회로 소자의 수를 비교하였다. 커널 연산 회로가 없는 DSVM보다 다소 소자의 수가 증가하는 것을 알 수 있다. 반면에 샘플수가 32개일 때 학습 단계 시간을 비교해보면, DSVM은 140,000 사이클이 소요된 반면에 CSVM은 122,250 사이클이 걸려서 연산 속도가 증가한 것을 알 수 있다. 또한 CSVM은 칩 위에서 커널 연산 동작까지 병렬로 이루어지므로 Off-chip 상에서 커널 연산하는 DSVM에 비교하여 전체 동작 시간을 더 감소시킨다.

표 5. 회로 소자 수 비교.

프로세서	Registers	Counters	Mux	Tristates	Decoders	Add/Sub	Muliplier	Comps.	Shifters
DSVM	213	1	210	20	1	165	32	34	0
CSVM	282	0	193	33	0	124	64	92	44

6. 결 론

본 연구에서는 공유 데이터 버스를 이용한 병렬 연산과 온칩 커널 연산을 사용하는 효율적인 CSVM 프로세서 구조를 제안하였다. CSVM 프로세서에 채용한 SVM 학습 알고리즘인 Kernel Adatron과 Kernel Perceptron 방법의 실험 결과를 질병진단 문제, 비선형 등화 문제에 적용하고 기존에 많이 사용하는 QP방법의 결과와 비교하여 더 빠르고 구현이 용이함을 보였다. 소프트웨어를 사용한 SVM 처리 시간이 샘플의 수에 따라 2차적으로 증가하는 반면, 하드웨어 설계 결과 처리 시간이 선형적으로 증가하여 많은 샘플을 가지는 문제에 대하여 상당히 처리 속도가 증가하는 것을 알 수 있었다. 하드웨어 성능인 처리 시간과 회로 면적에 있어서 기존의 프로세서와 비교하여 더 우수한 성능을 보였다.

본 연구에서 제안한 CSVM 프로세서는 SVM에 필요한 모든 동작을 포함하는 최초의 SVM 프로세서이다. 병렬 구조와 공유 버스를 이용한 Concurrent 동작으로 빠른 SVM 연산이 가능하고 특히 고차원의 데이터의 분류 문제에 적합하다. 많은 계산을 요구하여 지금까지 호스트컴퓨터에서 Off-line으로 수행하던 SVM의 커널 연산을 하드웨어 위에서 공유 데이터 버스를 이용하여 빠르게 처리하였다. 커널 연산을 포함한 SVM에서 필요한 모든 연산이 CSVM 하드웨어 위에서 한 번에 빠르게 이루어지므로 온라인, 독립적인 수행(Stand-alone)이 요구되는 적용분야에 사용 가능하다. 커널 스케일링 방법은 많은 데이터의 특징을 요구하는 바이오인포매틱스 분야의 패턴분류 문제에 효율적이다.

CSVM의 기본 모듈인 SVE가 많은 면적을 소모하는 메모리와 곱셈기를 포함하여 많은 샘플 데이터를 가진 대상에 적용한다면 면적이 너무 커지므로 비용이 오르는 단점이 있다. 또한 SVE의 개수가 많아질수록 병렬성이 증대되어 속도가 더 증대되므로 한정된 하드웨어 공간에 더 많은 SVE를 포함시켜야 한다. 많은 종류의 실시간 적용 분야에 CSVM을 적용하고 병렬성을 극대화하기 위하여 SVE의 면적을 최소화하는 연구가 필요하다.

본 연구에서는 SVM의 커널 파라미터 등을 실험적으로 정하였는데, 사용자에게 실험의 편리성을 제공하기 위해서는 최적의 파라미터 값을 찾는 방법의 도입이 필요하다. 진화 알고리즘 등의 최적화 방법을 사용하여 가장 좋은 성능을 가지는 파라미터를 찾는다면 사용자의 불편함을 덜어 줄 수 있을 것이라고 생각한다.

참고 문헌

[1] Genov, R. and Cauwenberghs, G., "Kerneltron: Support Vector "Machine" in Silicon", IEEE Transactions on Neural Networks, Vol. 14, No. 5, pp. 1426-1434, 2003.

[2] D. Anguita, A. Boni and S. Ridella, "A Digital Architecture for Support Vector Machines: Theory, Algorithm and FPGA Implementation", IEEE Transactions on Neural Networks, Vol. 14, No. 5, pp. 993-1009, 2003.

[3] D. Anguita, A. Boni and S. Ridella, "Digital Kernel Perceptron", Electronics Letters, Vol. 38, No. 10, pp. 445-446, 2002.

[4] Vapnik V. N., Statistical Learning Theory, John Wiley and Sons, New York, 1998.

[5] T.T. Friess, N. Cristianini and C. Campbell, "The Kernel-adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines", Proceedings of the 15th International Conference on Machine Learning, Wisconsin, USA, pp. 188-196, 1998.

[6] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring", Science, Vol. 286, pp. 531-537, 1999.

[7] S. Dudoit, J. Fridlyand and T. P. Speed, "Comparison of Discrimination Methods for the Classification of Tumors using Gene Expression Data", Journal of the American Statistical Association, Vol. 97, No. 457, pp. 77-87, 2002.

[8] D. J. Sebald, and J. A. Buchlew, "Support Vector Machine Techniques for Nonlinear Equalization," IEEE Transactions on Signal Processing, Vol. 48, No.

11, pp. 3217-3226, 2000.
[9] Xilinx Inc., Virtex-II Data Sheet,
http://www.xilinx.com, 2001.

저|자|약|력

성 명 : 위재우

◈ 학 력
· 1997년
인하대 전기공학과 공학사
· 1999년
동 대학원 전기공학과 공학석사
· 2005년
동 대학원 전기공학과 공학박사

◈ 경 력
· 2005년
인하대 수퍼지능기술연구소 전임연구원
· 2005년 - 현재
특허청 전자상거래심사팀 심사관

성 명 : 이종호

◈ 학 력
· 1976년
서울대 전기공학과 공학사
· 1978년
동 대학원 전기공학과 공학석사
· 1986년
미국 아이오와주립대 전기 및 컴퓨터 공학과 공학박사

◈ 경 력
· 1979년 - 1982년
해군사관학교 전임강사
· 1980년 - 1982년
국방과학연구소 위촉연구원
· 1986년 - 1989년
미국 노틀담대학교 조교수
· 1997년 - 1998년
인하대 집적회로설계센터소장
· 1989년 - 현 재
인하대 정보통신공학부 교수
· 2000년 - 현 재
수퍼지능기술연구소 소장