情報保護學會論文誌
第 21 卷 第 6 號, 2011. 12

# 비디오 감시 시스템을 위한 멀티코어 프로세서 기반의 병렬 SVM*

김 희 곤,[1†] 이 성 주,[1] 정 용 화,[1‡] 박 대 희,[1] 이 한 성[2]
[1]고려대학교, [2]ETRI

# Multicore Processor based Parallel SVM for Video Surveillance System*

Heegon Kim,[1†] Sungju Lee,[1] Youngwha Chung,[1‡] Daihee Park,[1] Hansung Lee[2]
[1]Korea University, [2]ETRI

요 약

최근 지능형 비디오 감시 시스템은 영상 분석 및 인식기술 등의 보다 진화된 기술 개발을 요구하고 있다. 특히, 비디오 영상에서 객체를 식별하기 위하여 Support Vector Machine(SVM)과 같은 기계학습 알고리즘이 이용된다. 그러나 SVM은 대용량의 데이터를 학습시키기 위하여 많은 계산량이 필요하기 때문에 수행시간을 효율적으로 감소시키기 위하여 병렬처리 기법을 적용할 필요가 있다. 본 논문에서는, 최근 사용이 증가하고 있는 멀티코어 프로세서를 활용한 SVM 학습의 병렬처리 방법을 제안한다. 4-코어 프로세서를 이용한 실험 결과, 제안 방법은 SVM 학습의 순차처리 방법과 비교하여 2.5배 정도 수행시간이 감소됨을 확인하였다.

ABSTRACT

Recent intelligent video surveillance system asks for development of more advanced technology for analysis and recognition of video data. Especially, machine learning algorithm such as Support Vector Machine (SVM) is used in order to recognize objects in video. Because SVM training demands massive amount of computation, parallel processing technique is necessary to reduce the execution time effectively. In this paper, we propose a parallel processing method of SVM training with a multi-core processor. The results of parallel SVM on a 4-core processor show that our proposed method can reduce the execution time of the sequential training by a factor of 2.5.

**Keywords**: Video Surveillance Applications, SVM, Parallel Processing, Multi-core

## I. 서 론

현대 사회에서는 보안과 범죄 예방 등의 목적으로 다양한 분야에서 비디오 감시 시스템을 이용하고 있다. 특히 대형화 된 비디오 감시 시스템에서는 많은 인력과 비용을 필요로 한다. 사람이 항상 모니터링을 해야 하는 비디오 감시 시스템의 문제점을 보완하기 위해 감시 영역의 객체를 탐지하고 상황을 분석하는 지능형 감시 시스템이 제안되고 있다[1]. 특히, 비디오 감시 시스템에서 범인과 같은 특정 인물의 얼굴, 표정, 행동패턴 등을 인식하거나 분류하여 분석하는 분야의 연구가 활발히 진행되고 있고, 특정 객체의 패턴을 분류하기 위해서 기계학습 알고리즘을 이용한 많

접수일(2011년 10월 25일), 게재확정일(2011년 12월 2일)
* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (C1090-1111-0010)
† 주저자, khg86@korea.ac.kr
‡ 교신저자, ychungy@korea.ac.kr

은 기법들이 연구되어져 왔다[2].

SVM[3,4]은 얼굴인식 응용에서 뛰어난 인식성능을 제공하는 방법으로 알려져 있으며, 비디오 감시 시스템용으로 많은 연구가 발표되고 있다[5,6]. 특히, SVM은 분류 전의 학습 과정에서 많은 계산량이 필요하기 때문에 다중 프로세서(multi-processor)를 이용한 병렬처리로 수행시간을 단축할 필요가 있다.

이러한 병렬 SVM에 대한 기존 연구로는 크게 다중 클래스와 이진 클래스 SVM의 병렬처리로 구분할 수 있다. 통상 다중 클래스 SVM은 다수의 SVM이 각각 독립적으로 학습을 할 수 있으므로 자연스럽게 병렬처리 될 수 있으며, 분산메모리 시스템/클러스터나 GPU에서 병렬처리 결과가 다수 발표되고 있다[7-10]. 반면, 이진 클래스 SVM은 학습 과정에서의 데이터 종속성으로 인하여 데이터 통신 및 동기화 오버헤드가 상대적으로 큰 분산메모리 시스템이나 클러스터에서 병렬처리가 쉽지 않다는 문제가 있다. 이를 해결하기 위하여 대량의 학습 데이터를 분할하고 분할된 학습 데이터 간에는 데이터 종속성이 없는 것처럼 병렬처리를 하는 방법 등이 제안되었으나[11-20], 정확도면에서 차이가 있다는 또 다른 문제가 발생한다[21].

본 논문에서는 최근 사용이 증가하고 있는 멀티코어 프로세서를 이용하여 학습 데이터간의 데이터 종속성을 준수하면서 학습 시간을 감소시키는 병렬처리 방법을 제안한다. 즉, 이진 클래스 SVM 학습의 계산 특성을 먼저 분석하고, 데이터 종속성을 준수하면서 가장 많은 시간을 필요로 하는 부분을 공유메모리 기반의 병렬 프로그래밍 언어인 Pthread[22]로 병렬처리 함으로써 정확도 및 수행시간 감소를 동시에 보장할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 SVM 및 기존의 병렬 SVM 기법과 멀티코어 프로세서에 대해서 설명하고, 3장에서는 멀티코어를 이용하여 이진 클래스 SVM 학습을 병렬처리 하는 방법을 제안한다. 4장에서는 실험을 통하여 제안 방법의 성능을 확인한다. 마지막으로 5장에서는 결론으로 논문을 마친다.

## II. 배 경

### 2.1 SVM

최근 이진 분류기로 많이 활용되는 SVM은 구조적 위험 최소화 개념에 기반한 최적의 선형 결정 평면을 찾음으로써 두 개의 클래스를 분류하는 방법이다[3]. 이때 결정 평면은 학습 원소들의 가중화된 조합이며, 이러한 학습 원소들을 support vector라 부르고 두 클래스간의 경계면을 나타낸다.

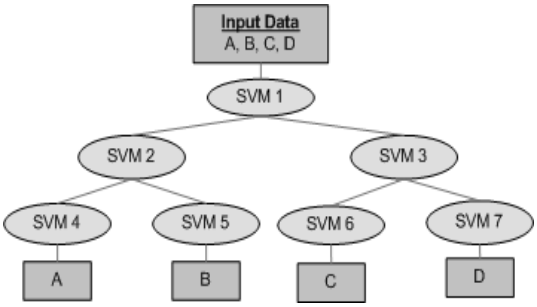
예를 들어, 선형 분리 데이터를 가정할 때 최대 여백 분류의 목표는 support vector들의 거리가 최대화되는 초월면(hyperplane)에 의해 두 개의 클래스를 분리하는 것이다. 이러한 초월면은 최적 경계 초월면(optimal separating hyperplane)이라고 불리며, 2차 프로그래밍(Quadratic Progamming) 문제의 해를 구함으로써 support vector를 구할 수 있다.

또한 선형 분리가 불가능한 데이터인 경우에는, 입력 벡터를 선형 초월면이 발견되는 고차원의 특징 공간으로 비선형 매핑할 수 있다. 이때 목표 함수와 결정 함수가 데이터 벡터의 내적으로 표현됨에 따라, 계산적으로 복잡한 매핑을 명시적으로 계산할 필요가 없어진다. 즉, Mercer 조건을 만족하는 커널 함수는 데이터 벡터 대신에 사용되는 매핑 함수를 다시 치환할 수 있다. 본 논문에서는 2차 프로그래밍의 해로 널리 사용되는 Sequential Minimal Optimization(SMO) 방식의 LIBSVM[4]을 가정하고, 커널 함수로 Radial Basis Function(RBF)을 가정한다.

### 2.2 병렬 SVM

SVM은 분류 전의 학습 과정에서 많은 계산량이 필요하기 때문에 SVM 학습 시간을 단축하기 위해서 많은 연구들이 진행 되어왔다. 예를 들어, [그림 1]과 같이 복수개의 SVM으로 구성된 다중 클래스 학습은 각 SVM 학습 간 데이터 종속성이 없어 손쉽게 병렬처리할 수 있으며, 분산메모리 시스템/클러스터나 GPU에서 병렬처리 결과가 다수 발표되고 있다[7-10]. [그림 1]을 보면 SVM 1이 완료된 후에 SVM 2와 SVM 3은 서로 데이터 종속성이 없기 때문에 두 개의 SVM을 병렬처리할 수 있다. 마찬가지로 SVM 2이 완료된 후에 SVM 4와 SVM 5는 병렬처리 할 수 있고, SVM 3이 완료된 후에 SVM 6과 SVM 7을 병렬처리 할 수 있다. 이처럼 다중 클래스 SVM의 경우 병렬처리를 쉽게 할 수 있음을 알 수 있다.

반면, 하나의 SVM으로 구성된 이진 클래스 학습은 복잡한 데이터 종속성으로 인하여 쉽게 병렬처리 할 수 없는 어려움이 있다[14]. 이진 클래스 SVM은

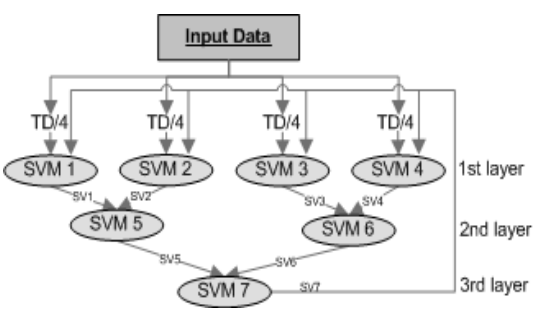



〔그림 1〕 다중 클래스 SVM

하나의 학습부분을 병렬처리 해야 하므로 학습하는 데이터의 종속성을 판단하고 병렬처리를 해야 하므로 보다 세밀한 병렬성의 확보가 필요하다. 특히 데이터 통신 및 동기화 오버헤드가 상대적으로 큰 분산메모리 시스템이나 클러스터에서는 병렬처리가 쉽지 않다는 문제가 있다. 이를 해결하기 위하여 대량의 학습 데이터를 분할하고 분할된 학습 데이터 간에는 데이터 종속성이 없는 것처럼 병렬처리를 하는 방법 등이 제안되었으나〔11-20〕, 정확도면에서 차이가 있다는 또 다른 문제가 발생한다〔21〕.

예를 들어, 〔14〕 등에서 제안된 decomposition 방법은 데이터 종속성을 피하기 위하여 전체 학습 데이터를 구분하여 복수개의 SVM을 병렬로 학습하고 support vector가 될 가능성이 없는 학습 데이터를 제외하는 과정을 반복하여 최종적인 support vector를 구한다(〔그림 2〕 참조). 또한, interior-point QP solver를 기반으로 한 Psvm〔17〕은 데이터 종속성을 완화하기 위하여 불완전한 Cholesky factorization으로 커널 행렬을 근사화한다.

그러나 이러한 이진 클래스 병렬화 연구들은 SVM의 순차 학습에 존재하는 데이터 종속성을 회피 또는 완화하기 위하여 순차 처리와 다른 계산 과정을 거침으로써 그 결과가 순차 처리와 달라질 수 있다는 문제가 있다. 예를 들어, 기존의 병렬 처리 결과 중 학습의 최종 결과물인 support vector의 수를 순차 처리와 비교한 모든 논문〔11,12,13,15〕에서 적게는 1개에서 많게는 100개 이상까지 차이가 있음을 확인하였다. 병렬 처리의 목적이 순차 처리와 동일한 결과를 빠른 시간 내에 얻는 것이라고 가정하면, SVM 순차 학습에 존재하는 데이터 종속성을 준수하는 범위에서 병렬처리로 수행시간을 단축할 필요가 있다.

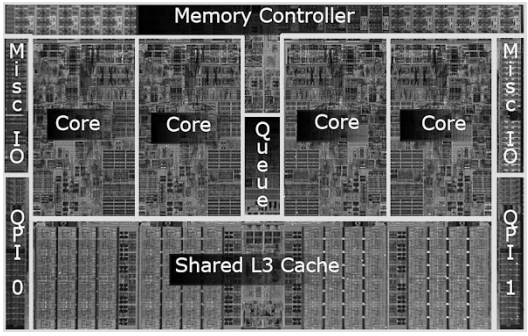



〔그림 2〕 Cascade SVM〔14〕

## 2.3 멀티코어 프로세서

1970년대 초에 최초의 범용 프로세서가 등장한 이후 프로세서의 성능 개선을 위하여 주파수를 높이는 방식을 지속적으로 취해왔으나, 3GHz 이상의 주파수에서는 전력소모량이나 발열량 등이 크게 증가한다는 문제가 발생하였다〔23〕. 이러한 문제를 극복하기 위하여 주파수를 높이는 대신 하나의 칩 안에 코어 수를 증가시키는 방식으로 프로세서의 성능을 개선하는 방식이 제안되었다. 즉, 2005년에 인텔에서 최초의 2-코어 프로세서가 발표되었으며, 이후 하나의 칩 안에 내장되는 코어의 수는 지속적으로 증가하고 있다. (예를 들어, 2012년에는 48-코어의 서버용 프로세서가 인텔에서 발표될 예정) 특히, 서버나 PC 뿐만 아니라 최근에는 스마트폰 등 임베디드 시스템으로까지 범용 멀티코어 프로세서의 사용범위가 증가하고 있는 실정이다. 이러한 추세에 비추어 기존의 싱글코어 프로세서에서 동작하던 응용 프로그램을 멀티코어 프로세서에서 동작하도록 병렬화가 필요한 시점이다.

일반적으로 멀티코어 프로세서는 기존의 싱글코어 프로세서와 동일한 처리 능력을 갖는 코어를 복수개 내장한 "범용" 멀티코어와 그래픽 등 특수 응용에 적합한 코어를 복수개 내장한 "특수목적용" 멀티코어로 구분할 수 있다. 예를 들어, 최근 사용이 증가하고 있는 Graphics Processing Unit(GPU)는 그래픽 등의 특수 응용에 맞도록 설계된 멀티코어 프로세서이다〔9〕. 하지만 GPU는 Single Instruction Multiple Data(SIMD) 방식으로 동작하는 특성으로 인하여 제한적인 응용에 적용될 수 있으며, Compute Unified Device Architecture(CUDA) 프로그래밍은 통상의 순차 프로그래밍과 확연히 다른 구조를 가지고 있다. 반면, Multiple Instruction Multiple Data(MIMD) 방식으로 동작하는 범용 멀티코어 프로세서는 병렬성이 존재하는 임의의 응용을 처리할 수 있으며 프로그래밍 방식도 순차 프로그래밍과




〔그림 3〕 Intel i7 4-core processor〔26〕

유사한 구조를 가지므로, 그 적용 범위가 매우 넓다고 할 수 있다. 본 논문에서는 Intel i7 범용 멀티코어를 이용한 병렬처리를 고려하고, 그 내부 구조는 〔그림 3〕과 같다.

## III. 본 론

먼저 이진 클래스 SVM 학습의 계산 특성을 분석하고, 암달의 법칙에 의한 성능 한계치를 구한 후, 공유메모리 기반의 Pthread를 이용하여 SVM 학습을 병렬처리한다.

### 3.1 이진 클래스 SVM 학습의 계산 특성 분석

본 논문에서 고려하는 LIBSVM에서는 Q Matrix를 만드는데 커널 함수를 사용하고, 이 커널 함수의 단일 수행시간은 작지만 학습을 위한 반복 수행 횟수가 상당히 많다는 특징이 있다. 따라서 LIBSVM에서는 학습을 위한 수행시간의 대부분을 반복적인 Q Matrix 생성에 사용하게 된다. 각각의 Q Matrix를 만드는 부분은 행렬 연산으로 이루어져 있고, 개별 Q Matrix 생성내의 행렬 연산은 데이터 종속성이 없는 것으로 확인하였다. 그러나 각각의 Q Matrix 생성 간에는 데이터 종속성이 있고, 한 번의 반복문이 수행되어야 다음 반복문을 수행 할 수 있는 〔그림 4〕와 같은 구조를 가지고 있다.

LIBSVM 학습의 계산 특성을 분석하기 위하여, PC 환경(Intel i7-720QM 4-core)에서 순차처리로 실험데이터 adult〔24〕의 수행시간을 측정한 결과, 총 수행시간이 15.03초이고, Q Matrix를 생성하는데 사용된 누적 수행시간은 13.11초이었다. 즉, 반복적으로 Q Matrix를 생성하는 것이 총 수행시간의 약

〔표 1〕 SVM 학습에 필요한 계산량

| | 전체 수행시간 | 반복문 수행시간 | Q Matrix 생성시간 |
|---|---|---|---|
| 시간 (sec) | 15.03 | 14.72 | 13.11 |
| 백분율 (%) | 100 | 97.9 | 87.2 |

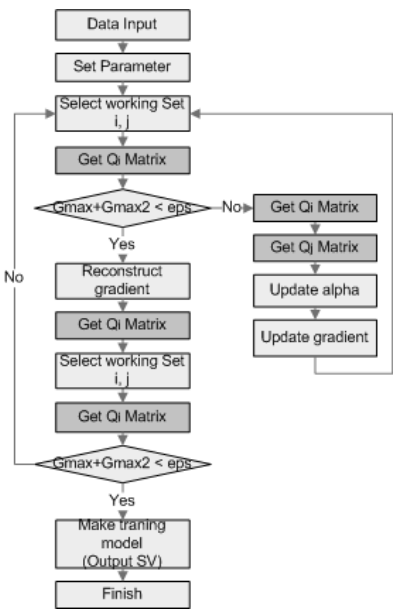
87%를 차지하는 것을 알 수 있다. 또한, Q Matrix 생성을 4-코어에서 병렬처리 할 경우 얻을 수 있는 최대 speedup은 암달의 법칙에 의하여 계산할 수 있다. 즉, LIBSVM의 학습을 4개의 코어에서 병렬처리한다면 식 (1)에서 보는 것처럼 이론적으로 최대 2.89배의 성능향상을 기대할 수 있다.

$$Speedup = \frac{Time_s(1-core)}{Time_p(4-core)}$$

$$= \frac{Sequential\ Part\ Time}{Sequential\ Part\ Time + \frac{\parallel\ Part\ Time}{4}}$$

$$= \frac{15.03\text{sec}}{1.92\text{sec} + (\frac{13.11}{4} sec)}$$

$$\fallingdotseq 2.89 \quad (1)$$




〔그림 4〕 LIBSVM 학습 순서도

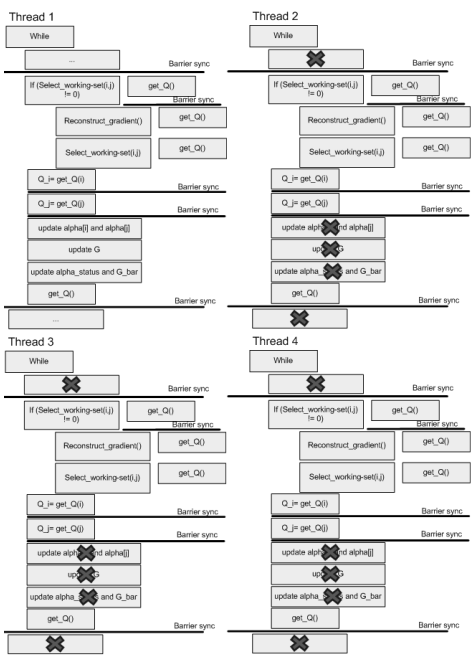
### 3.2 Pthread를 이용한 이진 클래스 SVM 학습의 병렬처리

LIBSVM의 학습은 입력 데이터에 대하여 반복적인 연산을 하다가 미리 정한 임계값보다 작아지게 되면 학습이 종료된다. 상황에 따라서 약간의 차이가 있지만, 대부분의 경우 한 번의 반복문 내에서 Q Matrix를 생성하는 Get_Q() 함수가 3번 호출된다. 앞서 계산한바와 같이 가장 많은 시간이 소요되는 Q Matrix 생성만을 병렬처리 하더라도 4-코어에서 충분한 speedup을 얻을 수 있으므로, 본 논문에서는 Q Matrix 생성 부분 중 Get_Q() 함수를 Pthread로 병렬화 하였다.

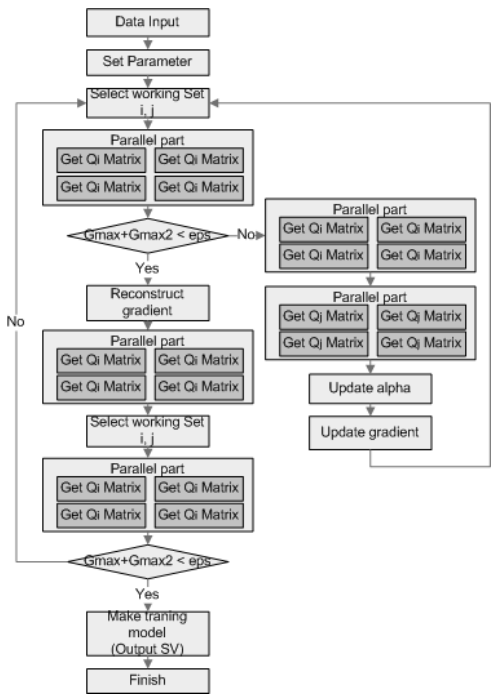
앞서 말한 것과 같이 Get_Q() 함수는 단일 수행시간은 짧으나 무수히 많은 반복으로 인하여 Get_Q()의 전체누적 수행시간은 SVM 학습 수행시간의 대부분을 차지하게 된다. 이 부분의 병렬처리를 위해 쓰레드 생성을 반복문 내부에서 하게 된다면 무수히 많은 쓰레드가 생성이 되는 결과를 가져오게 된다. 즉, 이렇게 많은 쓰레드가 생성된다면 쓰레드들의 생성을 위한 수행시간이 추가로 소요되므로 이러한 경우 병렬처리를 했음에도 불구하고 수행시간이 오히려 늘어날 수 있다.

본 논문에서는 [그림 5]와 같이 반복문 외부에서 쓰레드를 생성하는 방법을 선택하였다. 생성된 쓰레드들은 내부의 Get_Q() 함수만 병렬처리를 하게 되고 나머지 수행은 순차처리를 하게 된다. Get_Q() 함수 외의 부분들은 데이터들의 종속성이 있기 때문에 병렬처리를 하면 쓰레드간의 통신을 위한 오버헤드가 증가하여 수행시간의 증가를 일으킬 수 있으므로 병렬처리를 하지 않았다.

[그림 6]은 본 논문에서 제안한 병렬 LIBSVM의 학습 순서도를 나타내고 있다. 전체적인 흐름은 순차처리와 거의 유사하지만, Get_Q() 함수 부분을 병렬처리 하고 있다. Get_Q() 함수에서 Q Matrix 생성을 위해서 몇 가지 인자가 필요하다. 이러한 데이터 종속성을 만족시키기 위해 Get_Q() 함수 호출 전에 배리어(barrier) 동기를 사용한 후, 병렬처리를 위한 정보들을 모든 쓰레드들과 공유한다. 이렇게 얻은 정보로 Q Matrix를 생성하고, Q Matrix생성이 완료되면 이후 연산을 위한 데이터 종속성을 만족시키기 위해 다시 한 번 배리어 동기를 사용한다. 이렇게 반복문을 수행하고 종료조건을 만족시키면 SVM의 학습은 끝나게 된다.




[그림 5] 병렬 SVM 학습에서의 thread 할당(4개 쓰레드를 이용한 경우)




[그림 6] 병렬 LIBSVM 학습 순서도(4개 쓰레드를 이용한 경우)

## IV. 실험 결과

본 논문에서는 Intel core i7 720QM 1.6GHz 4코어 프로세서[26]를 이용하여 병렬처리를 하였다. 실험 데이터는 이진 클래스 SVM 학습에 많이 쓰이는 adult, web 데이터[24]와 멀티미디어에서의 실험을 위해 이미지 데이터에서 특징점을 추출한 scene 데이터[25]를 사용하였다. 특히, scene 데이터는 다중 클래스 학습의 경우이지만, 본 논문에서 제안한 이진 클래스 SVM 병렬처리 방법이 다중 클래스 학습에 적용될 수 있는지를 확인하기 위하여 추가로 실험하였다. adult, web, scene 데이터의 특징은 [표 2]와 같다.

순차처리 SVM과 제안한 병렬처리 SVM의 성능을 비교하면 [표 3] 및 [그림 7]과 같다. 순차처리는 기존에 공개되어 있는 LIBSVM을 사용하였고, 병렬처리는 LIBSVM을 제안한 방법으로 수정한 병렬 LIBSVM을 사용하였다. 또한, 쓰레드 개수에 따른 추이를 확인하기 위해 쓰레드 개수를 변화시켜서 실험을 하였다.

실험결과 4개의 코어를 이용할 경우 병렬처리가 순차처리보다 최대 약 2.5배 정도의 수행시간이 감소하였음을 확인 할 수 있다. 특히, scene 데이터의 경우 본 논문에서 대상으로 한 이진 클래스가 아닌 점과 특징점 중에 값을 가지고 있는 데이터가 많은 점 때문에 쓰레드 개수가 늘어남에 따라 성능향상이 예상치 만큼 이루어지지 않았다. 단지 데이터 개수와 특징점 개수가 아닌 학습데이터의 특징에 따라서 학습시간 차이가

[표 2] 학습 데이터 특징

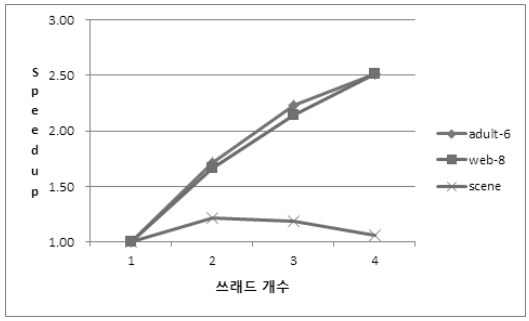
(단위: 개)

| 학습데이터 | 클래스 개수 | 데이터 개수 | 특징점 개수 |
|---|---|---|---|
| adult-6 | 2 | 11,220 | 123 |
| web-8 | 2 | 49,749 | 300 |
| scene | 6 | 1,211 | 294 |

[표 3] 이진 클래스 SVM 학습 시간

(단위: 초)

| 학습 데이터 | 순차 LIB SVM | 병렬 LIBSVM | | | |
|---|---|---|---|---|---|
| | | 쓰레드 개수 | | | |
| | | 1 | 2 | 3 | 4 |
| adult-6 | 14.98 | 15.10 | 8.81 | 6.78 | 6.00 |
| web-8 | 35.89 | 36.26 | 21.80 | 16.89 | 14.43 |
| Scene | 2.72 | 2.73 | 2.25 | 2.30 | 2.58 |




[그림 7] 쓰레드 개수에 따른 병렬 SVM의 speedup

날 수 있으며, 다중 클래스 학습의 경우 독립적인 개별 SVM 학습을 병렬로 처리하는 것이 더 효율적일 수 있음을 알 수 있다. 또한, 쓰레드 개수가 4개인 점을 감안하면 최대 2.5배의 성능향상은 아주 좋은 결과라고 볼 수 없지만, 이것은 앞에서 계산한 암달 법칙의 성능 상한치에 근접함을 확인 할 수 있었다.

또한, [표 4]의 테스트 데이터로 support vector의 수와 정확도를 비교한 결과, [표 5]와 같이 각각 순차처리와 병렬처리를 할 때 support vector의 수는 adult-6에서 4340개, web-8에서 2957개, scene에서 1112개로 동일함을 확인하였다. 학습데이터의 정확도를 판단하기 위해 추가적인 데이터로 분류를 시도한 결과, adult-6에서의 분류 정확도가 순차처리와 병렬처리에서 각각 84.18%로 동일하고, web-8에서도

[표 4] 테스트 데이터 특징

(단위: 개)

| 학습데이터 | 클래스 개수 | 데이터 개수 | 특징점 개수 |
|---|---|---|---|
| adult-6 | 2 | 21,341 | 123 |
| web-8 | 2 | 14,951 | 300 |
| scene | 6 | 1,196 | 294 |

[표 5] 순차처리 LIBSVM과 병렬처리 LIBSVM의 support vector 수와 분류 정확도 비교

| 학습 데이터 | 처리방법 | Support vector 개수 | 정확도 (%) |
|---|---|---|---|
| adult-6 | 순차LIBSVM | 4340 | 84.18 |
| | 병렬LIBSVM | 4340 | 84.18 |
| web-8 | 순차LIBSVM | 2957 | 97.45 |
| | 병렬LIBSVM | 2957 | 97.45 |
| scene | 순차LIBSVM | 1112 | 70.07 |
| | 병렬LIBSVM | 1112 | 70.07 |

각각 97.45%, scene에서도 두 가지 처리방법에서 각각 70.07%로 동일한 것을 확인하였다.

## V. 결 론

본 논문에서는 이진 클래스 SVM의 학습 시간을 단축시키기 위해 멀티코어 프로세서를 이용한 병렬처리 방법을 제안하였다. 먼저 이진 클래스 SVM 학습의 계산 특성을 분석한 후, 데이터 종속성을 만족하면서 가장 많은 시간을 필요로 하는 부분을 효과적으로 병렬처리 하였다. 4-코어 프로세서에서의 실험을 통하여 제안 방법이 순차처리 방법에 비하여 최대 약 2.5배 빠른 학습이 가능하고, 이러한 성능 향상은 암달의 법칙에 의하여 계산된 성능 상한치에 근접함을 확인할 수 있었다. 또한, 제안된 방법은 침입 탐지 등 비정상 상황 탐지를 위한 단일 클래스 SVM 학습에 확대 적용될 수 있어, 다양한 정보보호 응용에 활용될 수 있을 것으로 기대된다.

## 참고문헌


[1] M. Valera and S. Velastin, "Intelligent Distributed Surveillance Systems: a Review," IEE Proceedings on Vision, Image and Signal Processing, vol. 152, no. 2, pp. 192-204, Apr. 2005.

[2] A. Tolba, A. El-Baz, and A. El-Harby, "Face Recognition: a Literature Review," Intl. J. of Signal Processing, vol. 2, no. 2, pp. 88-103, Feb. 2006.

[3] V. Vapnik, The Nature of Statistical Learning Theory, 2nd Ed., Springer, Nov. 1999.

[4] C. Chang and C. Lin, LIBSVM: a library for Support Vector Machines, http://www.csie.ntu.edu.tw/ ~cjlin/libsvm.

[5] 강봉수, 정호석, 이한성, 임영희, 정용화, 박대희, "혼합 계층형 SVM을 이용한 실시간 요주의 인물 식별 시스템," 보안공학연구논문지, 7(5), pp. 479-494, 2010년 10월.

[6] 이종욱, 강봉수, 이한성, 정용화, 박대희, "감시 시스템에서 SVDD와 SRC를 이용한 범죄 용의자 얼굴 식별," 한국정보과학회 논문지, 17(2), pp. 135-139, 2011년 2월.

[7] C. Zhang, P. Li, A. Rajendran, and Y. Deng, "Parallel Multicategory Support Vector Machines for Classifying Microarray Data," Proc. of IMSCCS, pp. 110-115, Jun. 2006.

[8] J. Munoz-Mari, A. Plaza, J. Gualtieri, and G. Camps-Valls, "Parallel Implementations of SVM for Earth Observation," Parallel Programming, Models and Applications in Grid and P2P Systems, pp. 292-312, Jun. 2009.

[9] S. Herrero-Lopez, J. Williams, and A. Sanchez, "Parallel Multiclass Classification using SVMs on GPUs," Proc. of GPGPU, pp. 2-11, Mar. 2010.

[10] 이은지, 이성주, 강봉수, 정용화, 박대희, 민병기, "비디오 감시 시스템을 위한 GPU 기반의 계층형 다중클래스 SVM," 신호처리합동학술대회, pp. 40-43, 2010년 10월.

[11] R. Collobert, S. Bengio, and Y. Bengio, "A Parallel Mixture of SVMs for Very Large Scale Problems," Neural Computation, vol. 14, no. 5, pp. 1105-1114, May. 2002.

[12] G. Zanghirati and L. Zanni, "A Parallel Solver for Large Quadratic Programs in Training Support Vector Machines," Parallel Computing, vol. 29, no. 4, pp. 535-551, Apr. 2003.

[13] J. Dong, A. Krzyzk, and C. Suen, "A Fast Parallel Optimization for Training Support Vector Machine," Proc. of MLDM, pp. 96-105, Jul. 2003.

[14] H. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik, "Parallel Support Vector Machines: The Cascade SVM," Proc. of NIPS, pp. 521-528, Dec. 2004.

[15] L. Cao, S. Keerthi, C. Ong, P. Uvaraj, X. Fu, and H. Lee, "Developing Parallel Sequential Minimal Optimization for Fast Training Support Vector Machine," Neurocomputing, vol. 70, no. 1-3, pp. 93-104, Dec. 2006.

[16] J. Fan, Y. Gao, and H. Luo, "Hierarchical

Classification for Automatic Image Annotation," ACM SIGIR, pp. 111-118, Jul. 2007.

[17] E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui, "PSVM: Parallelizing Support Vector Machines on Distributed Computes," Proc. of Advances in Neural Information Processing Systems, pp.257-264, Dec. 2007

[18] D. Bickson, E. Yom-Tov, and D. Dolev, "A Gaussian Belief Propagation Solver for Large Scale Support Vector Machines," Proc. of European Conference on Complex Systems, pp. 1640-1648, Sep. 2008.

[19] T. Hazan, A. Man, and A. Shashua, "A Parallel Decomposition Solver for SVM: Distributed Dual Ascend using Fenchel Duality," Proc. of CVPR, pp. 1-8, Jun. 2008.

[20] Y. Lu, V. Roychowdhury, and L. Vandenberghe, "Distributed Parallel Support Vector Machines in Strongly Connected Networks," IEEE Tr. on Neural Networks, vol. 19, no. 7, pp. 1167-1178, Jul. 2008.

[21] N. Alham, "Parallelizing Support Vector Machines for Scalable Image Annotation," Ph.D. Thesis, Brunel University School of Engineering, Apr. 2011.

[22] B. Barney, "POSIX Threads Programming," http://www.llnl.gov/computting/tutorials/pthreads, 2006.

[23] D. Patterson and J. Hennessy, Computer Organization and Design, 4th Ed., Morgan Kaufmann, Nov. 2008

[24] B. Scholkopf, C. Burges, and A. Smola, Advances in Kernel Methods - Support Vector Learning, The MIT Press, Dec. 1998.

[25] M. Boutell, J. Luo, X. Shen, and C. Brown, "Learning Multi-label Scene Classification." Pattern Recognition, vol. 37, no. 9, pp. 1757-1771, Sep. 2004.

[26] Intel core i7 Processor, http://www.intel.com/pressroom/archive/releases/2008/20081117comp_sm.htm

## 〈著者紹介〉

김 희 곤 (Heegon Kim) 학생회원
2011년 2월: 고려대학교 전산학과 학사
2011년 3월~현재: 고려대학교 전산학과 석사과정
〈관심분야〉 멀티코어, 에너지 효율성

이 성 주 (Sungju Lee) 학생회원
2006년 2월: 고려대학교 전산학과 학사
2006년 3월~2008년 2월: 고려대학교 전산학과 석사
2008년 3월~현재: 고려대학교 전산학과 박사과정
〈관심분야〉 멀티코어, 에너지 효율성, 정보보호

정 용 화 (Yongwha Chung) 종신회원
1984년: 한양대학교 전자통신공학과 학사
1986년: 한양대학교 전자통신공학과 석사
1997년: 미국 Univ. of Southern California 전기공학과(컴퓨터공학 전공) 박사
1986년~2003년: 한국전자통신연구원 생체인식기술연구팀장
2003년~현재: 고려대학교 컴퓨터정보학과 교수
〈관심분야〉 생체인식, 정보보호, 생체정보 보호

박대희 (Daihee Park) 정회원
1982년: 고려대학교 수학과 학사
1984년: 고려대학교 수학과 석사
1989년: 플로리다 주립대학교 전산학과 석사
1992년: 플로리다 주립대학교 전산학과 박사
1993년 ~ 현재: 고려대학교 컴퓨터정보학과 교수
〈관심분야〉 지능 데이터베이스, 데이터마이닝, 인공지능, 퍼지이론

이 한 성 (Hansung Lee) 정회원
1996년: 고려대학교 전산학과 학사
1996년 ~ 1999년: (주)대우엔지니어링
2002년: 고려대학교 전산학과 석사
2008년: 고려대학교 전산학과 박사
2006년 ~ 2007년: 고려대학교 컴퓨터정보학과 초빙전임강사
2008년 ~ 2009년: 고려대학교 BK21 연구교수
2009년 ~ 현재: 한국전자통신연구원
〈관심분야〉 멀티미디어 마이닝, 휴먼인식, 네트워크 마이닝, 기계학습, 지능 데이터베이스