Self-Organizing Map을 이용한 유닉스 시스템 사용자의 명령어 패턴 분석

김인영, 장병탁
서울대학교 컴퓨터공학과
E-mail: {iykim, btzhang}@scai.snu.ac.kr

Analysis of Command Patterns of Unix Users with a Self-Organizing Map

In-Young Kim and Byoung-Tak Zhang
Department of Computer Engineering
Seoul National University
E-mail: {iykim, btzhang}@scai.snu.ac.kr

요약

기존에 알려진 침입 패턴에만 대응할 수 있는 수동적인 침입 탐지 시스템과 달리 사용자의 비정상적 행위를 감지하고 이를 기반으로 침입 여부를 탐지할 수 있는 시스템의 개발이 필요하다. 이러한 시스템의 개발을 위해서 사용자의 명령어 패턴을 분석하고 비정상적인 패턴에 대해서 탐지하도록 훈련시킬 수 있다. 본 논문에서는 Self-Organizing Map 신경망을 이용하여 유닉스 시스템에서 사용자의 명령어 패턴을 클러스터링하고 비정상적인 명령어 패턴을 구별해내는 방법을 제시하고자 한다.

1. 서론

인터넷, 인트라넷 등과 같은 정보 기술의 활용으로 거의 모든 시스템이 개방 환경에 노출되어 있다. 이로 인한 이익도 많은 반면 시스템 침입에 의한 정보 유출과 파괴 같은 역기능도 무시할 수 없는 정도가 되었다. 이에 따라 시스템 침입 탐지 및 예방 시스템의 개발이 요구되었고 국내외에서 이미 침입 탐지 시스템의 개발이 이루어진 상황이다.

그러나 기존의 보안 관련 시스템들에는 보통 능동적인 데이터 습득 및 이를 통한 자동화된 학습 과정이 존재하지 않는다. 대신 흔히 사용되는 방식은 현재 사용되는 해킹의 패턴을 관리자가 분석한 후 이러한 절차 내지는 패턴을 직접 if-then 규칙으로 시스템 안에 심어주고 이러한 규칙에 적용되는 사용자의 접속은 불법 침입에 해당하는 것이라고 규정하는 것이다. 그러므로 이 경우 알려진 침입 패턴에 대해서는 탐지할 수 있지만 새로운 침입 패턴에 대해서는 적절히 대처할 수 없다는 단점이 있다.

이러한 단점을 보완하기 위해서 각각의 유형에 따라 침입을 탐지하는 시스템보다 더 일반적인 탐지 시스템 개발이 요구된다. 이는 사용자의 비정상적인 명령어 패턴 탐지를 바탕으로 침입 탐지 시스템 개발을 함으로써 만족시킬 수 있다.

이러한 시스템의 개발을 위한 기본 단계는 각 사용자가 쓰는 명령어의 패턴을 학습하고 비정상적인 패턴을 탐지하는 것이다. 이에 따라 본 논문에서는 SOM (Self-Organizing Map)을 이용하여 사용자 명령어 패턴을 클러스터링하고 임의의 명령어 패턴에 대해서 분류하는 것을 실험하려고 한다. 사용자 명령어 패턴을 클러스터링 하는 과정은 학습하는 과정으로 볼 수 있고 임의의 명령어 패턴을 분류하는 과정은 학습 후 예측이라고 생각할 수 있다. 즉, 임의의 명령어 패턴에 대해서 시스템 침입 시도 여부에 관한 예측을 하는 과정이라고 볼 수 있다.

SOM을 이용한 패턴 클러스터링을 위한 데이터로는 사용자가 남긴 명령어 로그와 접속한 시각이 있다. 데이터 수집을 위해 일정 기간동안 사용자의 명령어와 로그인한 시간을 history file로 남기게 되며 이것을 이용하여 SOM의 입력 데이터를 생성해낸다. 생성된 데이터들을 클러스터링한 후 임의의 패턴을 입력으로 주어 어떤 클래스에 속하는 지 분류해내는 작업을 한다. 만약 전혀 새로운 패턴이라면 기존의 어떤 클래스에도 속하지 않을 것이며 이는 곧 비정상적인 명령어 사용을 의미한다. 그러므로 이러한 경우, 극단적으로는 침입으로 간주할 수도 있다. 여기서 실험하고자 하는 것은 비정상적인 명령어 패턴을 탐지하는 것으로 국한하며 그 이상의 침입 여부 판단은 다음 단계에서의 할 일로 남겨두고자 한다.

2. 문제 정의 및 데이터 수집

본 실험에서는 유닉스 시스템에서 사용자의 명령어 패턴을 클러스터링한 후 임의의 입력 패턴을 분류하고 비정상적인 패턴 탐지를 목적으로 한다. 사용자의 비정상적인 명령어 패턴을 탐지하기 위해서는 평상시의 사용자 명령어 패턴을 학습시켜야 하며 충분히 훈련을 한 후 평상시의 패턴과 다른 명령어 패턴이 들어오면 이를 비정상적인 행위라고 판단할 수 있는 것이다[1].

실험에 사용할 데이터 수집은 서울대학교 컴퓨터공학과 학부생들이 사용하는 컴퓨터와 인공지능 연구실에 있는 컴퓨터를 대상으로 하였다. 데이터의 수집 기간은 2주일 정도이고 사용자 20여명에 대해서 데이터 수집을 하였다.

데이터 수집은 사용자가 로그인 할 때 로그인 시각과 사용하는 명령어를 history file에 계속적으로 남기는 방법을 사용하였다. 로그인 시각을 남기는 이유는 각 사용자의 일별 명령어 사용 횟수를 알아내고 이를 이용하여 SOM의 입력 데이터들을 생성해내기 위해서이다.

History file로부터 입력 데이터를 생성해내기 위해서는 몇 단계의 처리 과정을 거치게 된다. 우선 실험에 참가한 사용자들 전체에 대해서 명령어 사용 횟수를 알아낸다. 그 후에 가장 많이 사용한 명령어부터 일정 개수의 명령어를 선택하고 각 사용자에 대해서 일별 해당 명령어 사용 횟수를 이용하여 입력 벡터를 만든다.

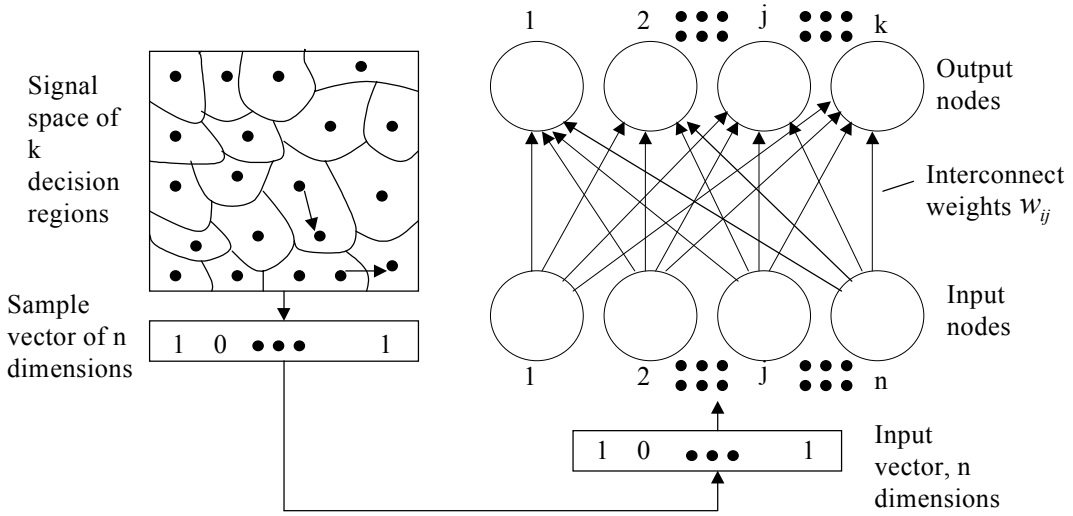
이 실험에서는 50개의 명령어들에 대한 일별 사용 횟수를 이용하여 50차원의 입력 벡터를 생성하였으며 실험에 사용한 명령어 집합은 [표1]과 같다.

ls	vi	cd	rm	exit	telnet	make	g++	ps	mv
cp	t	vim	man	lpq	mkdir	more	gcc	clear	grep
diff	hanterm	gdb	top	du	whoami	su	cat	chmod	kill
nohup	quota	mail	echo	logout	finger	openwin	which	a.out	showmoney
lpr	setenv	psnup	dir	where	who	tar	javac	find	date

[표 1] 실험에 사용한 50개의 명령어

3. SOM (Self-Organizing Map)

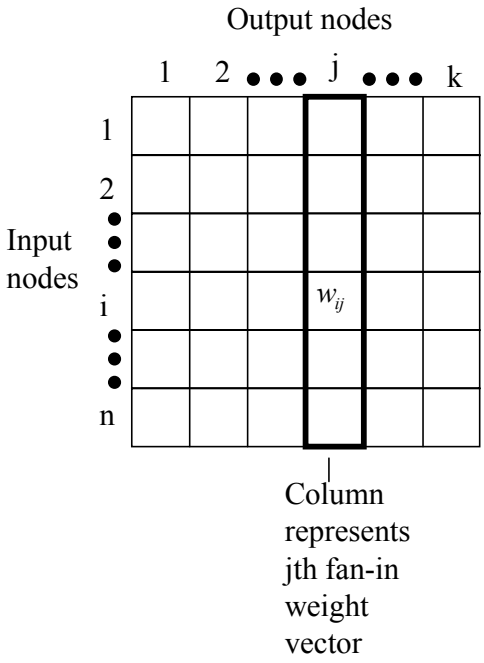
SOM은 무감독 학습 방법의 일종으로 스스로 n 차원의 입력 데이터들을 군집화하여 (클러스터링하여) 2차원으로 사상시켜준다. SOM은 신경망을 이용하여 구현한 것으로 그 내부는 아래의 그림과 같다[2, 3].



[그림 1] SOM의 구조

[그림1]은 2-layer 신경망으로 n 차원의 입력 데이터를 표현하는 n 개의 입력 노드들과 k 개의 분류 영역(decision region)을 표현하기 위한 k 개의 출력 노드로 구성되어 있다. 모든 입력 노드들은 모든 출력 노드들과 연결되어 있고 연결 가중치(weight)를 가진다.

[그림2]는 입력 노드 i 와 출력 노드 j 를 연결하는 weight w_{ij} 들의 행렬을 보여준다. 행렬에서 i 번째 행은 입력 노드 i 로부터 각 출력 노드로 나가는 연결 가중치를 나타내며 j 번째 열은 각 입력 노드로부터 출력 노드 j 로 들어오는 연결 가중치를 나타낸다. 여기서 j 번째 열로 나타내어지는 벡터는 j 번째 Fan-in weight vector라고 하며 입력 벡터와의 거리 (유클리드 거리) 계산에 쓰인다.



[그림 2] 연결 가중치 행렬

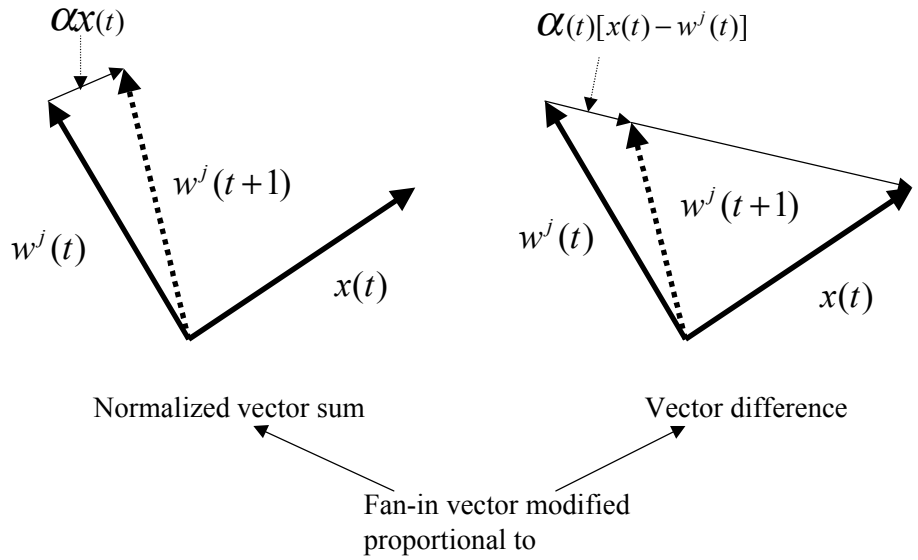
초기 상태에서는 연결 가중치들을 임의로 할당한다. 임의의 연결 가중치를 할당한 후 입력 벡터와의 유사성을 측정한다. 유사성 측정은 여러 가지 방법으로 할 수 있는 데 유클리드 거리를 이용하는 것도 하나의 방법이다. 입력 벡터와 k 개의 Fan-in weight vector 사이의 유클리드 거리를 구하여 입력 벡터와 가장 유사한 (유클리드 거리가 가장 작은) j 번째 Fan-in weight vector를 찾으면 그 입력 벡터에 대해서 j 번째 출력 노드가 승자가 된다. 이렇게 승자를 선택하면 승자의 Fan-in weight vector는 갱신되는 데 식으로 나타내면 아래와 같다.

$$w^j(t+1) = w^j(t) + \alpha(t)[x(t) - w^j(t)]$$

$$\alpha(t) = 0.1(1 - t/10^4)$$

이 식의 의미는 j 번째 출력 노드가 승자가 되었으면 그 노드의 연결 가중치 벡터는 입력 벡터 쪽으로 약간 이동한다. Fan-in weight vector를 입력 데이터 벡터와 비슷하게 만들어 가는 것이다.

[그림 3]은 연결 가중치 벡터 갱신 과정을 보여주는 데 연결 가중치 벡터가 입력 벡터 쪽으로 움직여 가는 것을 알 수 있다. 그림의 왼쪽은 Normalized Vector Sum을 이용하여 연결 가중치 벡터를 갱신하는 방법이고 그림의 오른쪽은 Vector Difference를 이용하여 갱신하는 방법이다. 두 가지 방법이 서로 다르지만 연결 가중치 벡터 갱신에 미치는 영향은 같다. 그림을 보면 왼쪽의 경우와 오른쪽의 경우 모두 연결 가중치 벡터가 입력 벡터의 방향으로 이동하는 것을 알 수 있다.



[그림 3] Fan-in vector 갱신

연결 가중치 벡터를 갱신하는 경우에 승자가 된 노드 하나의 Fan-in weight vector만 갱신하지 않고 그 노드의 주위에 있는 노드들의 (이웃 노드) 연결 가중치도 조금씩 갱신하는 방법도 사용 가능하다.

이렇게 각 입력에 대해서 승자가 정해지고 그에 따라 Fan-in weight vector가 입력 벡터 쪽으로 움직여 간다. 그 움직임은 초기에는 산만하나 점차 안정되어 간다. 훈련이 끝난 후 각각의 Fan-in weight vector는 각 분류 영역(클래스)의 centroid에 근사한다.

충분히 훈련이 된 SOM은 임의의 입력을 분류하여 특정 클래스에 할당할 수 있다. 훈련 단계에서 사용된 데이터와 비슷한 데이터가 입력으로 들어오면 맵상에서 가장 유사한 노드가 승자가 되고 해당 노드(클래스)로 분류되며 전혀 새로운 데이터가 입력으로 들어오면 맵에서 비슷한 클래스가 없으므로 새로운 노드가 할당되어 새 클래스를 만들게 될 것이다.

4. 실험 및 결과

SOM_PAK-3.1[4]을 이용하여 실험을 하였고 실험은 두 단계로 진행된다. 첫 번째는 훈련 단계이며 생성된 데이터의 대부분을 사용하여 클러스터링을 하는 단계이다. 이 과정을 통해 각 사용자들의 명령어 사용 패턴이 어떻게 분포되어 있는지 알 수 있다. 비슷한 명령어 패턴을 보이는 사용자들은 뭉쳐 있을 것이고 그렇지 않은 사용자의 경우는 다른 데이터들과 떨어져 분포할 것이다. 실험의 두 번째 단계는 훈련된 SOM을 이용하여 분류 작업을 하는 것이다. 이 때 테스트에 사용되는 데이터는 훈련에 사용되지 않은 데이터를 이용한다. 테스트 과정에 사용하는 데이터도 두 가지가 있는데 첫 번째는 훈련 과정에 참여한 사용자의 데이터 중 훈련에는 사용하지 않은 일부의 데이터이고 두 번째는 훈련 과정에 전혀 참여하지

않은 사용자의 데이터이다. 또는 임의로 만들어낸 데이터를 사용할 수도 있다.

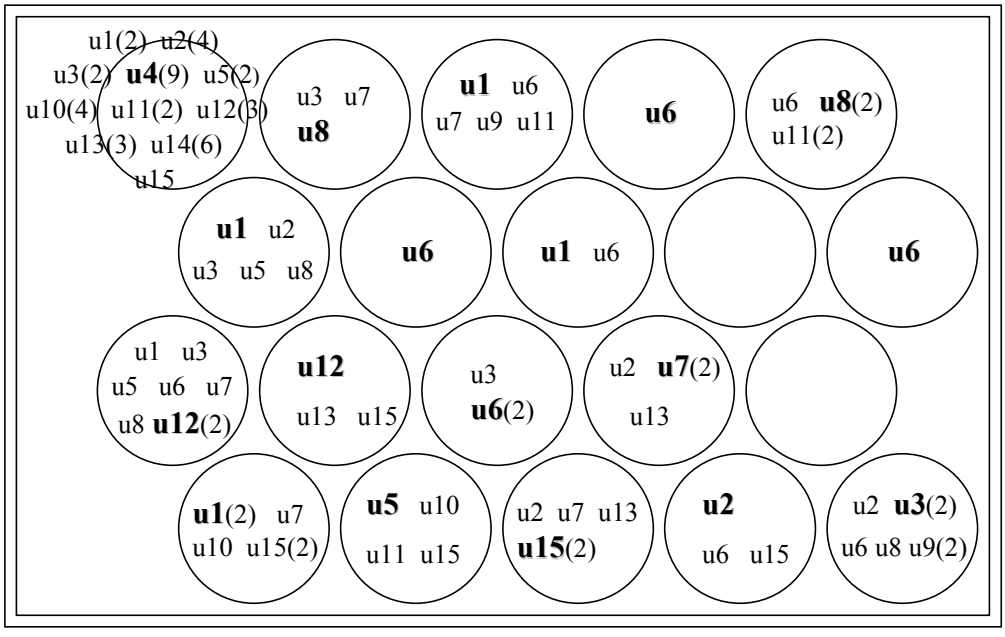
이 실험에서는 훈련을 위해 13명의 사용자 데이터 전부와 사용자 2명의 일부분의 데이터(약 70%)를 사용하여 클러스터링을 하였으며 테스트를 위해서는 훈련 과정에 사용하지 않은 데이터 (사용자 2명의 30% 데이터)와 훈련 과정에 참여하지 않은 사용자들의 데이터를 이용하였다.

가. 실험에 사용한 파라미터는 아래의 [표2]와 같다.

파라미터	값
dimension	5x4
topology type	hexa
neighborhood type	gaussian
training length of first part(ordering phase)	1000
training rate	0.05
radius	10
training length of second part(convergence phase)	20000
training rate	0.02
radius	4

[표 2] 파라미터 설정

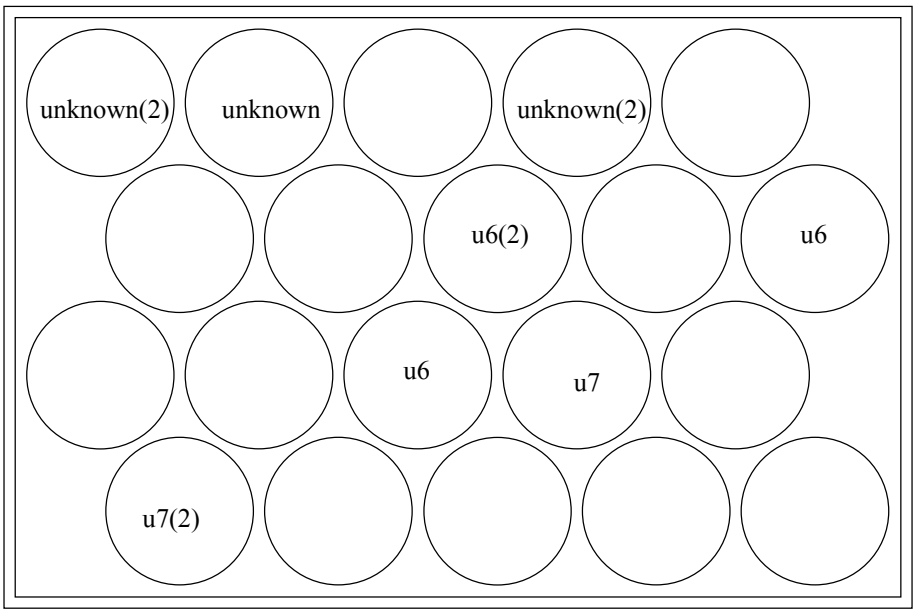
나. 실험의 첫 번째 단계로 데이터를 클러스터링한 결과이다.



[그림 4] 클러스터링 결과 : 5x4 SOM

[그림4]는 데이터들을 클러스터링 한 결과를 보여준다. 2차원 상의 5x4 맵을 나타내며 20개의 각 클래스는 x 좌표 0부터 4까지, y 좌표 0부터 3까지의 범위로 나타내어진다. 클러스터링에 사용된 데이터들은 사용자 1번부터 사용자 15번까지 15명의 데이터이며 그림에서 u1부터 u15까지 기호로 나타내었다. 각 클래스에 굵은 글자로 표시된 것은 각 클래스의 centroid에 해당하고 괄호 안에 표시된 숫자는 중복되어 나타나는 횟수를 나타낸다. 각각의 데이터는 20개 클래스 중 하나에 속하게 되며 유사한 데이터들은 같은 클래스에 속하거나 인접한 클래스에 속해 있다.

다. 실험의 두 번째 단계로 입력 데이터에 대해 분류 작업을 한 결과이다.



[그림 5] 테스트 결과: 분류 작업

[그림5]는 [그림4]의 맵을 이용하여 입력을 분류해 본 결과이다. u6과 u7의 데이터 중 클러스터링에는 사용하지 않은 데이터를 이용하여 분류해 보았다. [그림4]와 비교해서 보면 u6과 u7은 제대로 분류되었음을 알 수 있다.

그림에서 'unknown' 은 클러스터링에 참여하지 않은 사용자의 명령어 패턴을 입력으로 주었을 때의 분류 결과이다. 그림을 보고 정확히 알 수는 없으나 unknown 데이터가 흩어져 나타나지 않고 중복되게 인접한 클래스에 속한다는 것은 unknown 데이터도 어떤 패턴을 가지며 그 패턴이 유사하기 때문에 인접한 클래스에 속하는 것이라고 볼 수 있다. 그림에서는 나타나지 않지만 전혀 새로운 패턴이 입력으로 들어올 경우 quantization error의 값을 보고 그것이 기존의 클래스와는 다른 패턴임을 발견할 수 있을 것이다.

5. 결론

SOM을 이용하여 사용자의 명령어 패턴을 클러스터링한 결과로 생성된 map은 명령어 패턴을 분류하는 데 사용할 수 있음을 실험을 통해 확인하였다. 즉, 사용자가 평상시와 다름없이 명령어를 사용하면 SOM은 이를 비슷한 패턴들의 클래스로 분류하고 그렇지 않은 경우에는 비정상적인 명령어 패턴임을 탐지하게 된다. 이러한 사실은 SOM을 이용하여 기존의 침입 탐지 시스템과는 다른 새로운 탐지 시스템을 구현할 수 있다는 것을 의미한다.

그러나 사용자가 비정상적인 패턴을 사용하는 것이 곧 침입 시도를 의미하지는 않는다. 그러므로 사용자의 평소와 다른 비정상적 패턴을 침입 시도라고 판단하기 위해서는 여러 가지를 고려하여야 하며 앞으로의 연구는 비정상적인 패턴을 탐지하였을 때 그것이 침입 시도인지 아니면 정상 사용자의 평소와는 다른 패턴인지를 구별하는 데 중점을 두어야 할 것이다[5].

감사의 글

본 연구는 학술진흥재단 자유공모과제(1999-001-E01025)에 의해 지원되었음.

참고문헌

[1] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion Detection with Neural Networks. In *NIPS-98*, pp. 943-949, 1998.

[2] Michael Chester. *Neural Networks: A Tutorial,* Prentice Hall. pp. 42-49, 1993.

[3] Simon Haykin. *Neural Networks: A Comprehensive Foundation,* Prentice Hall. pp. 443-483, 1999.

[4] Teuvo Kohonen, Jussi Hynninen, Jari Kangas and Jorma Laaksonen. *SOM_PAK The Self-Organizing Map Program Package, 1995.*

[5] Aurobindo Sundaram. *ACM Crossroads, April 1996.*