Self-Organizing Map을 이용한 유닉스 시스템 사용자의 비정상 행위 탐지

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

Anomaly Detection of Command Patterns of Unix Users with a Self-Organizing Map

In-Young Kim and Byoung-Tak Zhang
Seoul National University

요 약

인터넷, 인트라넷 등 컴퓨터 네트워크의 급속한 발전에 따라 컴퓨터 시스템의 보안이 중요한 문제로 떠올랐다. 이에 따라 시스템 침입 탐지를 위한 여러 연구가 진행되어 왔으나 지금까지의 연구는 대부분 기존의 침입 패턴을 탐지하는 데 중점을 두어 왔다. 그 결과 침입 탐지 시스템 내부에 입력되어 있지 않은 새로운 패턴으로 침입을 시도하는 경우에는 이를 탐지해내지 못한다. 본 논문에서는 사용자의 비정상 행위 탐지를 이용한 침입 탐지 시스템을 연구하고 Self-Organizing Map 신경망을 이용하여 유닉스 운영체제 사용자의 비정상 행위를 탐지하는 시스템을 구현하고자 한다.

1. 개 요

컴퓨터 시스템들간에 상호 연결성이 강화됨에 따라 인터넷, 인트라넷 등과 같은 정보 기술의 활용으로 거의 모든 시스템이 개방 환경에 노출되어 있다. 이로 인한 이익도 많은 반면 네트웍을 통한 시스템 불법 침입이 늘어나고 있다. 지금까지 침입 탐지를 위한 대부분의 방법은 If-then 규칙이나 인위적으로 획득된 접근 정보에 대한 단순한 분석에 의존하여 왔다. 그러므로 이 경우 알려진 침입 패턴으로 접근할 때는 침입을 탐지할 수 있지만 새로운 침입 패턴으로 접근할 경우에는 침입 여부조차 모를 수가 있다.

이러한 단점을 보완하기 위해서 최근 일부에서는 기계 학습 기법을 응용한 데이터 마이닝 기법을 사용하여 침입 패턴을 예측 및 탐지하려는 시도가 있어 왔다. 데이터 마이닝은 주어진 데이터로부터 알려지지 않은 유용한 지식 정보를 획득하는 일련의 과정을 일컫는다. 일반적인 마이닝 기법에서는 필요한 데이터가 미리 주어진다는 가정 하에서 출발하지만 이 가정은 전산망의 불법 침입 탐지와 같이 패턴이 복잡하고 다양하며, 매우 유동적인 환경에는 적용하기가 어렵다.

본 연구에서는 신경망을 이용한 기계 학습 기법을 통해 시스템 사용자의 비정상적인 행위를 탐지하고자 한다. 사용자의 비정상적인 행위라는 것은 평소의 명령어 사용 패턴과는 다른 비정상적인 명령어 패턴을 보이는 것을 말한다. 본 논문에서의 연구는 비정상 행위 탐지를 이용한 침입 탐지 시스템 개발을 위한

중요한 과정이 될 것이다.

이러한 시스템을 개발하기 위한 기본 단계는 각 사용자가 쓰는 명령어 패턴을 학습한 후 입력으로 주어진 명령어 패턴에 대해서 비정상 여부를 판단하는 것이다. 본 연구에서는 SOM (Self-Organizing Map) 신경망을 이용하여 사용자의 명령어 패턴을 클러스터링하고 임의의 명령어 패턴에 대해서 분류하는 과정을 거쳐 비정상 패턴을 탐지하도록 한다.
사용자 명령어 패턴을 클러스터링하는 과정은 학습 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 과정으로 볼 수 있다.

SOM 신경망을 이용한 명령어 패턴 클러스터링을 위한 데이터로는 사용자가 남긴 명령어 로그를 이용한다. 데이터의 수집을 위해 일정 기간동안 사용자의 명령어 로그를 남기며 이것을 이용하여 SOM의 입력 데이터를 생성해낸다. 생성된 데이터들을 클러스터링한 후 임의의 패턴을 입력으로 주어 어떤 클래스에 속하는 지 분류해내는 작업을 한다. 만약 전혀 새로운 패턴이 입력으로 들어올 경우 기존의 어떤 클래스와도 유사하지 않으며 이는 곧 비정상적인 명령어 사용을 의미한다. 그러므로 이 경우 시스템은 사용자의 비정상적인 행위가 있었음을 알 수 있다.

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 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 인공지능 연구실에 있는 컴퓨터를 대상으로 하였고 사용자 11명에 대해 데이터를 수집하였다. 이를 위해서 유닉스 시스템에서의 history file에 사용자의 로그인 시각과 사용하는 명령어를 남기도록 하였다. 로그인 시각을 남기는 이유는 사용자의 일별 명령어 사용 횟수를 알아내어 이를 데이터로 사용하기 위해서이다.

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

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

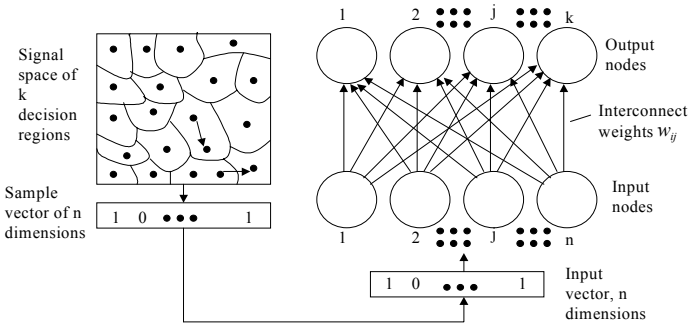
ls	exit	g++	history	cp	nslookup	finger	gzip	nohup	cat
cd	more	man	telnet	ps	ghostview	.hall	ping	clear	po
vi	grep	gcc	client	t	hanterm	rmdir	vcal	parse	s
rm	make	gdb	mkdir	su	gnuplot	sendrcvmsg	vsom	chmod	df
mv	mail	lpq	visual	sl	whoami	randinit	xfig	a.out	ftp

[표 1] 실험에 사용한 50개의 명령어

명령어 일별 사용 횟수는 nonlinear scaling을 이용하여 0부터 1까지 0.1의 간격으로 11개의 구간으로 나누었다. 예를 들면 해당 명령어를 한 번도 사용하지 않은 것은 0으로 하고 한 번 내지 두 번을 사용한 것은 0.1로 하였으며 가장 마지막 구간인 1은 해당 명령어를 200번 이상 사용한 경우에 해당한다. 일별 명령어 사용 횟수를 직접 데이터로 사용하지 않고 노말라이즈(normalize)하는 것은 실험 결과에 중요한 영향을 미친다. 특히 사용 횟수가 적은 부분에서 더 민감하다고 할 수 있다.

3. 학습 방법

SOM(Self-Organizing Map)은 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].

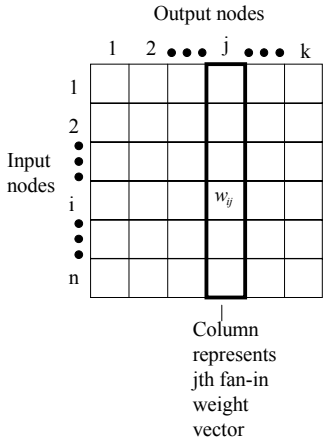


[그림 1] SOM의 구조

[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

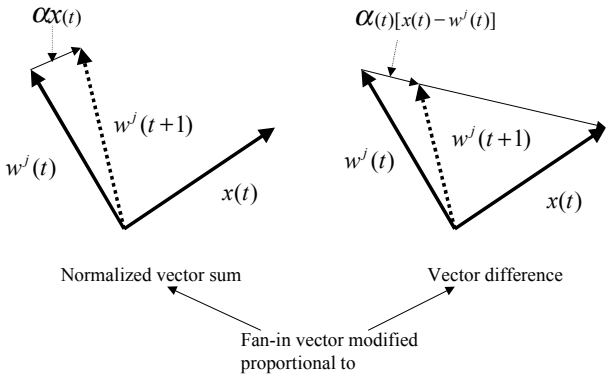
초기 상태에서는 연결 가중치들을 임의로 할당한다. 임의의 연결 가중치를 할당한 후 입력 벡터와의 유사성을 측정한다. 유사성 측정은 여러 가지 방법으로 할 수 있는 데 유클리드 거리를 이용하는 것도 하나의 방법이다. 입력 벡터와 k 개의 Fan-in weight vector 사이의 유클리드 거리를 구하여 입력 벡터와 가장 유사한 (유클리드 거리가 가장 작은) j 번째 Fan-in weight vector를 찾으면 그 입력 벡터에 대해서 j 번째 출력 노드가 승자가 된다. 이렇게 승자를 선택하면 승자의 Fan-in weight vector는 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 왼쪽과 오른쪽의 경우 모두 연결 가중치 벡터가 입력 벡터 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 입력 패턴에 대해 분류 작업을 수행할 수 있다. 임의의 입력 데이터에 대해서 그 데이터와 가장 유사한 값을 가지는 노드(클래스)로 분류를 하게 된다. 이 때 훈련 단계에서 사용된 데이터와 비슷한 데이터가 입력으로 들어왔다면 Quantization error[5] $\|x - m_c\|$ (입력 데이터 x와 노드 c의 대표값 사이의 거리) 가 비교적 작은 값을 가지게 될 것이고 기존의 데이터들과는 다른 새로운 패턴의 데이터가 입력으로 들어오면 Quantization error값이 굉장히 큰 값을 가지게 될 것이다. 따라서 Quantization error값이 미리 정해 놓은 Threshold 이상이 되는 데이터에 대해서는 비정상적인 데이터로 분류해낼 수 있다.

4. 실험 및 결과

SOM_PAK-3.1[4]을 이용하여 실험을 하였고 실험은 크게 훈련 단계와 예측 단계로 진행된다. 훈련 단계에서는 데이터를 학습하여 클러스터링하며 예측 단계에서는 입력으로 들어오는 데이터들을 분류하고 비정상 패턴을 탐지하게 된다.

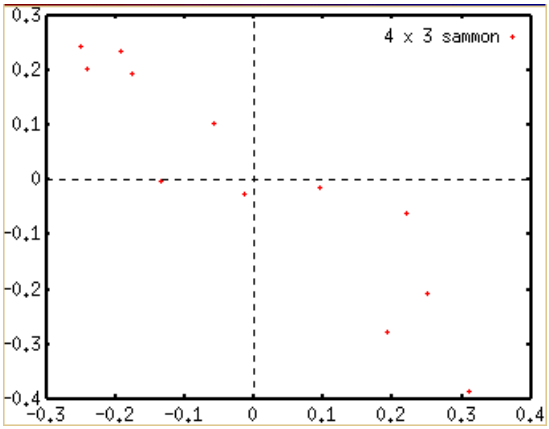
실험에 사용할 데이터는 두 가지로 나누어 사용한다. 하나는 트레인 데이터로 학습 과정(훈련)에 사용할 데이터이며 수집된 데이터의 70% 정도를 사용한다. 또 다른 하나는 테스트 데이터이며 나머지 30%정도의 데이터를 사용한다.

11명의 사용자에 대한 클러스터링을 위해 4×3개의 노드(클래스)를 가지는 맵을 이용하였고 실험에 사용한 파라미터는 아래와 같다[표 2].

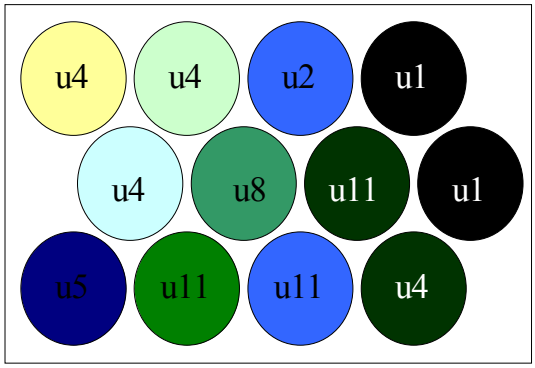
파라미터	값
dimension	4x3
topology type	hexa
neighborhood type	bubble
ordering phase :	
training length	1000
training rate	0.05
radius	8
convergence phase :	
training length	10000
training rate	0.02
radius	3

[표 2] 파라미터 설정

실험의 첫 번째 과정인 훈련 단계는 트레인 데이터를 사용하여 클러스터링을 하는 단계이다. 이 과정을 통해 사용자들의 명령어 패턴이 어떻게 분포되어 있는지 알 수 있다. 비슷한 명령어 패턴을 보이는 사용자들은 같은 노드(클래스)에 속하게 될 것이고 그렇지 않은 사용자의 경우는 다른 데이터들과 떨어져 다른 노드에 속하거나 같은 노드에 속하더라도 Quantization error가 큰 값을 가질 것이다. 훈련 과정으로 얻은 SOM맵은 아래와 같이 두 가지로 표현할 수 있으며 전체 데이터들의 분포를 각 노드 대표값(클러스터의 centroid)으로 2차원 상에 나타낸 것이다. 하나는 대표값 사이의 실제 거리(유클리드 거리)를 이용하여 2차원에 표현한 것이고[그림 4] 다른 하나는 대표값들의 유사한 정도를 명암으로 나타내었다 [그림 5].



[그림 4] 12개의 클러스터 대표값들의 분포



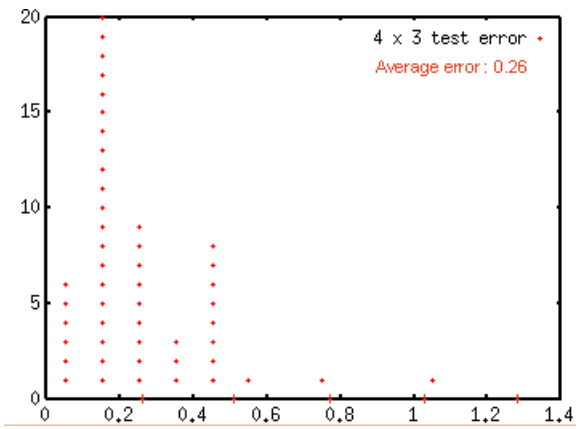
[그림 5] 12개 클러스터 대표값 유사성 분포

그림에는 나타나지 않았으나 [그림 4]에서 왼쪽 가장 위에 위치한 값과 오른쪽 가장 아래에 위치한 값은 [그림 5]에서 각각 가장 색깔이 진한 노드와 색깔이 가장 옅은 노드에 해당한다. [그림 4]에서 거리가 멀리 떨어져 있는 데이터들은 서로 패턴이 다른 데이터들이며 거리가 멀지 않은 데이터들은 그 패턴이 유사한 것이라고 볼 수 있다. 마찬가지로 [그림 5]에서도 색깔이 진한 것들은 서로 비슷한 패턴이고 명암 차이가 많이 나는 노드들은 서로 다른 패턴이다.

예측 단계에 들어가기 전에 훈련을 통해 얻은 SOM맵이 데이터(Train Data Set)에 대해 Over-Fitting 하지 않았는지 확인해보는 과정을 거친다. 이것은 Quantization error의 평균값을 비교해 보는 것으로 확인할 수 있다. 훈련된 SOM맵에 트레인 데이터를 적용했을 때의 평균 에러와 테스트 데이터를 적용했을 때의 평균 에러가 비슷하다면 트레인 데이터에 Over-Fitting 한 것이 아니라는 것을 알 수 있다.

실험의 두 번째 단계는 예측 단계로 훈련된 SOM을 이용하여 분류 작업을 하고 비정상적인 패턴을 탐지하는 과정이다. 이 때에 사용하는 데이터는 임의로 만들어 낸 Random data와 앞의 단계에서 사용하지 않은 새 데이터이다. 각 입력 데이터들은 훈련된 SOM맵상에서 가장 유사한 노드(클래스)로 분류가 되며 Quantization error가 Threshold θ 이상일 경우 비정상 패턴 (Anomaly)으로 탐지된다.

Threshold 값을 정하기 위해서 아래와 같은 그래프를 이용한다[그림 6]. 그래프는 테스트 데이터를 훈련된 SOM맵에 적용하였을 때 Quantization error의 분포를 보여준다. 그림에서 볼 수 있듯이 대부분의 데이터들이 가지는 에러 값은 평균 에러 δ의 두 배를 넘지 않는다. 그러므로 Threshold $\theta \approx 2\delta$로 설정할 수 있다.



[그림 6] 테스트 데이터를 적용한 에러 분포

훈련된 SOM맵에 임의의 데이터들을 입력으로 주고 실제로 예측을 한 결과 Random data는 Quantization error가 평균 에러 δ의 5배 이상인 값들로 나와 쉽게 비정상적인 패턴임을 탐지할 수 있었다. 또한 맵을 훈련할 때 사용하지 않은 데이터들에 대해서도 기존의 데이터 패턴과 유사할 경우 Quantization error가 Threshold θ를 넘지 않는 범위로 나와 정상적인 패턴임을 알 수 있었다.

5. 결론

SOM 신경망을 이용하여 데이터들을 학습하고 새로운 데이터들에 대해 비정상적 명령어 사용 패턴 여부를 분류하는 실험을 해 보았다. 실험 결과 평소의 사용 패턴과 다른 패턴이 입력으로 주어질 경우 비정상 패턴임을 쉽게 탐지할 수 있었다.

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[6].

비정상적 행위 탐지(사용자의 비정상적인 명령어 사용)를 함에 있어 몇 가지 고려할 점들이 있다. 이번 실험에서는 각 사용자마다의 명령어 사용 패턴에 대한 클러스터링을 한 것이 아니고 전체 데이터에 대해서 그 패턴의 유사성을 바탕으로 클러스터링한 것이다. 따라서 Quantization error가 작게 나왔더라도 그 사용자의 기존 패턴과는 다를 수가 있다. 즉, 다른 사용자의 패턴을 모방한 것으로도 볼 수 있다.

또한 비정상 패턴 여부를 결정하는 Threshold θ 를 정하는 것도 매우 신중히 해야 할 과정이다. 만약 θ값이 크다면 사용자가 조금씩 비정상적인 명령어를 사용함으로써 SOM맵은 비정상적인 패턴을 학습하여 결과적으로 그 패턴이 비정상이 아니라고 판단하게 될 것이다.

위와 같은 문제점으로 인해 비정상적 행위 탐지만으로 침입 탐지 시스템을 개발하는 것은 어려운 일이지만 기존의 탐지 시스템을 보강하는 역할은 충분히 할 수 있을 것이다.

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] Teuvo Kohonen. *Self-Organizing Maps: Second Edition,* Springer. pp. 48-51, 1997

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