ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC & KĨ THUẬT MÁY TÍNH

THỰC TẬP TỐT NGHIỆP

Nghiên Cứu Giải Thuật Gom Cụm Và Ứng Dụng Vào Hệ Thống Phát Hiện Xâm Nhập

Hội đồng: Mạng và Hệ Thống Máy Tính

Nhóm thực hiện:

Phan Sơn Tự 51204436
Nguyễn Thế Anh 51200082
Nguyễn Cẩm Diệu 51200493

Giáo viên hướng dẫn:

TS. Nguyễn Đức Thái
KS. Nguyễn Nhật Nam

TP HỒ CHÍ MINH, Ngày 30 tháng 5 năm 2016

Lời cam kết

Nhóm chúng em gồm ba thành viên Nguyễn Cẩm Diệu - 51200493, Phan Sơn Tự - 51204436 và Nguyễn Thế Anh - 51200082 là sinh viên khoa Khoa Học và Kĩ Thuật Máy Tính, Đại học Bách Khoa TP.HCM. Nhóm chúng em xin cam kết báo cáo thực tập tốt nghiệp với đề tài "Nghiên cứu giải thuật gom cụm và ứng dụng vào hệ thống phát hiện xâm nhập" là công trình nghiên cứu độc lập, tự tìm hiểu của nhóm, không sao chép bất kì công trình nghiên cứu nào.

Đề tài được thực hiện cho mục đích tìm hiểu và nghiên cứu ở bậc đại học.

Tất cả những tài liệu tham khao được ghi trong báo cáo đều được trích dẫn rõ ràng từ một số nguồn đáng tin cậy và từ một số bài báo khoa học.

Tất cả số liệu trong bài báo cáo đều được nhóm thực hiện một cách trung thực, không gian dối, không sao chép từ bất kì nguồn nào.

Các công cụ hỗ trợ cho việc thực hiện đo đạt số liệu đều là mã nguồn mở và tập dữ liệu nhóm thực hiện được chủ của tập dữ liệu chia sẻ rộng rãi trên mạng.

Hình ảnh trong bài báo cáo đều được trích dẫn nguồn gốc rõ ràng.

Lời cảm ơn

Lời đầu tiên, nhóm em xin được phép gửi lời cảm ơn chân thành đến TS Nguyễn Đức Thái đã tận tình giúp đỡ nhóm em trong thời gian thực hiện đề tài. Nhờ những hướng dẫn tận tình của thầy cùng với những sự góp ý, chỉ bảo, đưa ra những thiếu sót, khuyết điểm, ưu điểm đã giúp nhóm chúng em ngày càng hoàn thiện mình hơn và có thể hoàn thành đề tài một cách tốt đẹp.

Đồng thời, nhóm chúng em cũng xin gửi lời cảm ơn đến giáo viên Nguyễn Nhật Nam đã hỗ trợ nhiệt tình cho nhóm chúng em về mặt kiến thức, kĩ thuật và cung cấp cho nhóm chúng em các tài liệu có ích liên quan đến đề tài mà chúng em đang thực hiện, cùng với đó là định hướng kế hoạch thực hiện đề tài để nhóm chúng em hoàn thành trong thời gian tốt nhất.

Cuối cùng, nhóm em xin gửi lời cảm ơn và chúc quý thầy cô của khoa Khoa Học và Kĩ Thuật Máy Tính sức khỏe dồi dào để tiếp tục sứ mệnh nâng bước những sinh viên như chúng em tiếp tục con đường trau dồi kiến thức phục vụ đất nước và xã hội.

Thành phố Hồ Chí Minh, Ngày 30 tháng 5 năm 2016
Nguyễn Cẩm Diệu
Phan Sơn Tự
Nguyễn Thế Anh

Lời giới thiệu

Ngày nay sự phát triển của lĩnh vực công nghệ thông tin len lỏi vào khắp các lĩnh vực và có nhiều đóng góp quan trong trong đời sống hiện đại. Có khi nào chúng ta có thể tưởng tượng rằng máy tính có thể hiểu được ngôn ngữ tự nhiên của con người hay nó có giúp chúng ta phân loại ra được đâu là thóc và đâu là gạo không? Học máy (Machine learning) cho phép chúng ta thực hiện được điều tưởng chừng như không thể đó.

Máy học là ngành học cung cấp cho máy tính khả năng học hỏi mà không cần được lập trình một cách rõ ràng. Những thuật toán phân cụm (clustering) là một trong những kĩ thuật quan trọng trong lĩnh vực học máy, nó giúp chúng ta phân loại dữ liệu đầu vào bằng máy tính để áp dụng vào các lĩnh vực trong đời sống hằng ngày.

Trong đề tài thực tập này trình bày một số thuật toán phân cụm phổ biến, áp dụng các kĩ thuật gom cụm vào việc triển khai hệ thống phát hiện xâm nhập OSSEC HIDS. Ba đóng góp quan trọng của nhóm trong đề tài: Đầu tiên là đưa ra là trình bày ba thuật toán gom cụm phổ biến; thứ hai là các phương pháp thu giảm số chiều của ma trận thuộc tính; thứ ba là quy trình áp dụng học máy vào hệ thống phát hiện xâm nhập OSSEC HIDS.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh mục chữ viết tắt

PCA	Principal Component Analysis
SVD	Singular Value Decomposition
EM	Expectation Maximization
SOM	Self Organizing Maps
SVM	Support Vector Machine
BMU	Best Matching Unit
IDS	Intrusion Detection Systems
IPS	Intrusion Prevention Systems
NIDS	Network-based Intrusion Detection Systems
HIDS	Host-based Intrusion Detection Systems
OSSEC	Open Source HIDS SECurity
ADFA	Australian Defence Force Academy
ACL	Access Control List
BSD	Berkeley Software Distribution
HTTP	Hyper-Text Transfer Protocol

Chương 1

Giới thiệu đề tài

1.1 Tính cấp thiết của đề tài

Machine learning là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào thư mục tương ứng. Thế nhưng machine learning đã đi xa hơn những khái niệm ban đầu rất nhiều.

Chúng em chọn đề tài này vì nó khá mới mẻ ở Việt Nam và nhận thấy những lợi ích mà học máy đem lại khi giải quyết những vấn đề thực tiễn trong cuộc sống, ứng dụng trong nhiều lĩnh vực của đời sống xã hội. Bên cạnh đó, cơ hội việc làm về lĩnh vực học máy hay trí tuệ nhân tạo ở Việt Nam hiện nay là khá lớn, chúng em muốn nghiên cứu đề tài không chỉ tìm hiểu thêm cái mới mà còn để định hướng cho tương lai khi ra trường.

Hiện nay đã có một số lĩnh vực áp dụng thành công kĩ thuật học máy để giải quyết công việc như: Bệnh án điện tử, dự đoán thị trường chứng khoán, dự báo thời tiết, ... Ngoài ra, việc áp dụng học máy trong bảo mật thông tin đang là đề tài mới ở Việt Nam mà nếu áp dụng thành công học máy thì sẽ giúp các doanh nghiệp Việt Nam dự báo được các cuộc tấn công an ninh mạng từ đó có giải pháp bảo vệ tài sản thông tin tốt hơn.

1.2 Mục tiêu của đề tài

Trong lĩnh vực Học máy có các phương pháp học sau: Học có giám sát (supervised learning), Học không có giám sát (unsupervised learning), Học bán giám sát (semi-supervised learning), Học tăng cường (reinforcement learning). Đề tài của nhóm chúng em sẽ tập trung vào kĩ thuật học không giám

sát cụ thể là tìm hiểu các kĩ thuật gom cụm trong học máy từ đó áp dụng vào một ví dụ cụ thể đó là phân cụm một sự kiện xảy ra trong máy tính có phải là một cuộc tấn công an ninh mạng hay không. Để thực hiện được điều đó nhóm chúng em cần vạch ra những mục tiêu cụ thể để thực hiện đề tài:

- Tìm hiểu các kĩ thuật gom cụm phổ biến
- Tìm hiểu giải thuật thu giảm thuộc tính của ma trận trước khi đưa học máy vào để phân cụm
- Tìm hiểu các công cụ áp dụng học máy và vận dụng vào một tập dữ liệu mẫu
- Khảo sát và đánh giá độ hiệu quả của từng giải thuật học máy.
- Tìm hiểu framework mã nguồn mở về bảo mật thông tin OSSEC HIDS.

Nhóm chúng em đã tìm hiểu được ba giải thuật gom cụm là Kmeans, EM và SOM, sử dụng giải thuật PCA, SVD để thu giảm số chiều của ma trận thuộc tính, áp dụng các kiến thức đó vào việc phân cụm cuộc tấn công an ninh mạng, đánh giá kết quả và sử dụng một vài chức năng của OSSEC HIDS.

Trong giới hạn kiến thức tìm hiểu, nhóm dừng lại ở việc tìm hiểu giải thuật, sử dụng công cụ có sẵn để áp dụng các giải thuật gom cụm vào tập dữ liệu. Nhóm chưa áp dụng các kĩ thuật học máy vào OSSEC HIDS để xây dựng luật mà chỉ dừng lại ở việc tìm hiểu một số chức năng có sẵn của công cụ. Đồng thời kết quả của việc phân cụm của tập dữ liệu chưa đạt được hiệu quả như mong muốn.

1.3 Phương pháp thực hiện đề tài

Mục tiêu đề ra ở phần trên gồm 2 nội dung chính: nghiên cứu về học máy và tìm hiểu về công cụ phát hiện xâm nhập.

Với nội dung đầu, nhóm tiếp cận theo hướng tìm các tài liệu liên quan, nghiên cứu sau về lý thuyết từng giải thuật, hiểu rõ bản chất, và đánh giá cơ bản. Sau đó tìm tập dữ liệu thực tế, sử dụng các công cụ để áp dụng những giải thuật đã tìm hiểu vào tập dữ liệu, tiến hành đánh giá, và cải tiến giải thuật nếu cần.

Với nội dung sau, nhóm tập trung vào tìm hiểu sâu các module có trong hệ thống. Hiểu rõ hệ thống từ đó có thể sửa đổi theo nhu cầu đề tài.

1.4 Bố cục

Bố cục báo cáo được trình bày theo hướng để người đọc có thể dễ dàng tiếp cận và nắm bắt vấn đề. Nên trong phần tiếp theo sẽ trình bày những công trình liên quan và rút ra những kết luận về công trình này. Sau đó sẽ là nền tảng về lý thuyết để người đọc hiểu được phương pháp tiếp cận, hiện thực, ý tưởng của nhóm đưa ra và các kết quả đạt được. Phần tiếp theo là thiết kế, giải pháp, ý tưởng dựa vào những kiến thức nền tảng trên. Cuối cùng nhóm sẽ kết luận lại những gì đã đạt được, và đề ra hướng phát triển kế tiếp.

Chương 2

Những công trình liên quan

Ngày nay, vấn đề bảo mật dữ liệu đang rất cần thiết, số lượng các cuộc xâm nhập tấn công ngày càng gia tăng, cách thức tấn công cũng hết sức đa dạng và các cuộc tấn công với cách thức chưa xuất hiện bao giờ ngày càng nhiều. Lấy ý tưởng từ hệ thống phát hiện xâm nhập, từ rất sớm, các nhà khoa học đã mong muốn xây dựng một hệ thống kết hợp với học máy để có thể nhận biết các xâm nhập bất thường mà không cần định nghĩa hành vi xâm nhập một cách cụ thể.

Một trong những công trình gần đây có thể kể đến là "Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks" – Gideon Creech. Tổng quan về công trình này, đây là một bài luận văn, xây dựng hệ thống HIDS dựa trên việc học từ tập dữ liệu về systemcall có tên là ADFA. Trong công trình này, sử dụng hai hướng tiếp cận:

- Sử dụng giải thuật học máy One-Class SVM với độ chính xác cao, 15% với tỉ lệ báo sai và 80% với tỉ lệ báo đúng. Nhưng, vấn đề ở đây là giải thuật mất khá nhiều thời gian dành cho việc chạy giải thuật và gần như không thực tế trong công nghiệp.

- Sử dụng giải thuật học máy K-mean, vì giải thuật này tương đối đơn giản nên việc học là rất nhanh, cho ra kết quả trong thời gian ngắn. Nhưng, đánh đổi ở đây là độ chính xác, độ chính xác cho ra lại quá thấp và thiếu chính xác, điều này cũng dẫn đến việc không thể để hiện thực trong công nghiệp. Tỉ lệ báo đúng là 60% và tỉ lệ báo sai là 20%.

Nhìn nhận hướng tiếp cận này đã được giải quyết tốt bởi một chuyên gia – Creech – nhưng vẫn chưa thể áp dụng được vào công nghiệp mà chỉ mang tính chất nguyên cứu, học thuật nên nhóm quyết định thay đổi hướng tiếp cận với bài toán ban đầu.

Ở đây, bài toán của chúng ta là xây dựng một hệ thống phát hiện xâm

nhập mà không cần định nghĩa các hành vi xâm nhập một cách rõ ràng bằng cách dựa vào cơ sở học máy.

Hướng tiếp cận cổ điển gồm các bước:

1. Thu thập dữ liệu mẫu.
2. Sử dụng giải thuật gom cụm (unsupervised) để phân loại (không nhãn) đối với dữ liệu mẫu.
3. Dùng chuyên gia để gán nhãn.
4. Dùng giải thuật phân lớp (supervised) để phân loại (có nhãn) đối với dữ liệu đã được chuyên gia xử lý.
5. Kiểm tra, đánh giá, tối ưu và áp dụng mô hình cuối cùng vào hệ thống HIDS.

Hướng tiếp cận của nhóm:

1. Thu thập dữ liệu mẫu.
2. Sử dụng giải thuật gom cụm (unsupervised) để phân loại (không nhãn) đối với dữ liệu mẫu.
3. Dùng chuyên gia gán nhãn.
4. Đánh giá, tối ưu và tạo ra mô hình học máy.

Trong mô hình cổ điển, ta thấy các giải thuật gom cụm chủ yếu chỉ để giúp chuyên gia trong việc giảm tải khối lượng phải đánh nhãn, việc tạo ra mô hình học máy vẫn là nằm ở giải thuật phân lớp. Còn trong mô hình nhóm đề xuất, nhóm đã đưa giải thuật gom cụm thành giải thuật chính và sử dụng vào việc tạo ra mô hình học máy.

Nhóm cũng nhận thấy rằng, với hướng tiệp cận này chi phí phát triển cho toàn bộ quá trình được rút ngắn đi rất nhiều. Đồng thời, việc sử dụng cùng một giải thuật gom cụm ở cả quá trình tiền xử lý và quá trình học cũng giúp hạn chế thay đổi tính chất cụm và có thể nâng cao độ chính xác ở một mức độ nào đó.

Vì hướng tiếp cận này tương đối mới nên nhóm tập trung vào việc tìm ra một "đường học" có thể cân bằng được giữa tốc độ và độ chính xác.

Với tất cả những sự nhìn nhận trên, nhóm quyết định xây dựng một hướng tiếp cận mới và cố gắng phát huy những ưu điểm đang có của hướng tiếp cận này và hạn chế mắc phải những nhược điểm mà hướng tiếp cận cổ điển đã mắc trước đó.

Chương 3

Nền tảng lý thuyết

3.1 Kiến thức nền tảng

Trong phần này chúng ta sẽ đi vào việc ôn tập, nhắc lại một số định nghĩa, kiến thức của các môn nền tảng về Xác suất thống kê, Giải tích và Đại số tuyến tính. Đó là các kiến thức tối quan trọng được sử dụng rất nhiều trong việc xây dựng cũng như tối ưu thuật toán trong Học máy.

Đồng thời, trong phần này cũng đề cập đến một số kiến thức mới, các kiến thức này được sử dụng trong quá trình giải thích các thuật toán. Để hiểu rõ được từng bước đi của thuật toán ta cần nắm vững và vận dụng nhuần nhuyễn các kiến thức này.

3.1.1 Đại số tuyến tính

- Ma trận A cấp $m \times n$ là một mảng hình chữ nhật gồm m hàng và n cột với các phần tử là số hoặc các đối tượng toán học được biểu diễn như sau:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} = (a_{ij}),\ \forall\, i = \overline{1,m},\ j = \overline{1,n}$$

  Trong đó:
  $a_{ij} \in R$: là phần tử thuộc dòng i và cột j của ma trận A.
  m: số dòng của ma trận A.
  n: số cột của ma trận A.
  Ký hiệu $M_{m \times n}$ là tập hợp các ma trận $m \times n$.

- Vector là một ma trận A_{m1} có một cột và m dòng.

$$A = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix}$$

- Phép cộng và phép trừ hai ma trận. Để cộng và trừ hai ma trận ta thực hiện việc cộng và trừ lần lượt cho từng phần tử tương ứng nhau của hai ma trận. Lưu ý, hai ma trận cần có cùng chiều, nghĩa là số hàng và số cột của hai ma trận là như nhau.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} a+w & b+x \\ c+y & d+z \end{bmatrix}$$

- Phép nhân vô hướng là phép nhân giữa một ma trận và một số, ta thực hiện phép nhân vô hướng bằng cách nhân số đó cho tất cả các phần tử trong ma trận.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times x = \begin{bmatrix} a \times x & b \times x \\ c \times x & d \times x \end{bmatrix}$$

- Nhân hai ma trận: Cho $A \in M_{m \times k}$ và $B \in M_{k \times n}$. Gọi $A_1, A_2, \ldots, A_m$ là m dòng của A; $B^{(1)}, B^{(2)}, \ldots, B^{(n)}$ là n cột của B.
  Ta viết:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix}; \quad B = \begin{bmatrix} B^{(1)} & \ldots & B^{(n)} \end{bmatrix}$$

  Với

$$A_i = \begin{bmatrix} a_{i1} & a_{i2} & \ldots & a_{ik} \end{bmatrix}; \quad B^{(j)} = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{kj} \end{bmatrix}$$

  Khi đó $C = A \times B$ gọi là ma trận tích của A với B và phần tử của C_{ij} được xác định như sau: $c_{ij} = A_i \times B^{(j)} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj}$
  Một số tính chất của phép nhân hai ma trận:

  - Không có tính giao hoán: $A \times B \neq B \times A$
  - Tính kết hợp: $(A \times B) \times C = A \times (B \times C)$

- Ma trận đảo của A được ký hiệu là A^{-1} với tích của hai ma trận là một ma trận đơn vị.

- Ma trận chuyển vị của ma trận A là ma trận có được từ A bằng cách viết các hàng của ma trận A theo thứ tự thành cột, ký hiệu là A^T.

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

- Ma trận đối xứng: Gọi a_{ij} là phần tử của ma trận đối xứng A, thì $\forall\, i, j :\ a_{ij} = a_{ji}$
  Tính chất liên quan: Gọi x là một vector n chiều $(n \times 1)$ và $A = x.x^T$ thì A là một ma trận đối xứng $(n \times n)$

- Ma trận bán định dương (positive semi-definite): Ma trận M $(n \times n)$ được định nghĩa là ma trận bán định dương khi vào chỉ khi với vetor V bất kỳ có n chiều ta luôn có: $V^T MV > 0$

3.1.2 Xác suất thống kê

- Xác suất có điều kiện: Cho hai biến cố A và B. Ta gọi xác suất của biến cố A khi biến cố B đã xảy ra là xác suất của A với điều kiện B, ký hiệu là P(A|B).

- Công thức xác suất đầy đủ: Cho $A_1, A_2, \ldots, A_m$ là nhóm đầy đủ các biến cố, với mọi biến cố F ta có:

$$P(F) = P(A_1).P(F|A_1) + P(A_2).P(F|A_2) + \cdots + P(A_n).P(F|A_n)$$

- Công thức Bayes: Cho $A_1, A_2, \ldots, A_m$ là nhóm đầy đủ các biến cố, với mỗi $k(k = \overline{1,n})$, ta có:

$$P(A_k|F) = \frac{P(A_k).P(F|A_k)}{P(F)} = \frac{P(A_k).P(F|A_k)}{\sum_{i=1}^{n} P(A_i).P(F|A_i)}$$

- Kỳ vọng: Cho X là đại lượng ngẫu nhiên rời rạc có các bảng phân phối xác suất.

X	X_1	X_2	...	X_n	...
P	P_1	P_2	...	P_n	...

Bảng 3.1: Bảng phân phối xác suất

Khi đó ta gọi kỳ vọng của X là số:

$$E(X) = x_1p_1 + x_2p_2 + \cdots + x_np_n + \cdots = \sum_{n=1}^{\infty} x_np_n$$

Nếu X là đại lượng ngẫu nhiên liên tục có hàm mật độ f(x) thì kỳ vọng của X là:

$$E(X) = \int_{-\infty}^{+\infty} xf(x)dx$$

- Phương sai: Cho X là một đại lượng ngẫu nhiên có kỳ vọng E(X). Khi đó ta ký hiệu phương sai của X là D(X):

$$D(X) = E[(X - E(X))^2]$$

- Phân phối nhị thức: Đại lượng ngẫu nhiên rời rạc $X = 0, 1, 2, \ldots, n$ gọi là có phân phối nhị thức nếu tồn tại số $p \in (0, 1)$ sao cho:

$$p_k = P(X = k) = C_n^k p^k q^{n-k},\ q = 1 - p,\ k = \overline{0, n}$$

  Ta ký hiệu: $X \sim B(n, p)$

- Phân phối chuẩn: Đại lượng ngẫu nhiên X gọi là có phân phối chuẩn nếu hàm mật độ của X có dạng:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}},\ \sigma > 0$$

  Ta ký hiệu: $X \sim \mathcal{N}(a, \sigma^2)$

- Phân phối chuẩn tắc: Đại lượng : $X \sim \mathcal{N}(0, 1)$ gọi là có phân phối chuẩn tắc. Nếu X có phân phối chuẩn tắc thì hàm mật độ của X là:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

- Phân phối Gaussian: Phân phối chuẩn với n chiều còn được gọi với tên khác là phân phối Gaussian, nó được tổng quát hóa từ phân phối chuẩn của số thực lên cho vector nhiều chiều. Phân phối này được tham số hóa bằng hai đại lượng: một vector trung bình $\mu \in R^n$ và một ma trận tương quan (Covariance matrix) $\Sigma \in R^{n \times n}$ với $\Sigma \geq 0$ là một ma trận đối xứng (symmetric) và bán định dương (positive semi-definite). Ký hiệu: $\mathcal{N}(\mu, \Sigma)$.
  Ta định nghĩa, một vector x có quy luật phân phối chuẩn $\mathcal{N}(\mu, \Sigma)$ khi và chỉ khi hàm mật độ của X được biểu diễn dưới dạng:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$$

  Từ đó ta suy ra kỳ vọng của vector X chính là μ:

$$E[X] = \int_x xp(x; \mu, \Sigma)dx = \mu$$

Tổng quát hóa Covariance của một biến giá trị thực ngẫu nhiên ta có Covariance của một biến giá trị vector ngẫu nhiên và được định nghĩa:

$$Cov(Z) = E[(Z - E[Z])(Z - E[Z]^T)] = E[Z.Z^T] - (E[Z])(E[Z])^T$$

Mặc khác, nếu $X \sim \mathcal{N}(\mu, \Sigma)$ thì:

$$Cov(X) = \Sigma$$

3.1.3 Các lý thuyết đối với Vector

- Trực giao – Orthogonality
  Hai vector trực giao với nhau nếu tích vô hướng hai vector bằng không. Trong không gian hai chiều tương đương với các vector vuông góc, hoặc là hai vector tạo thành một góc 90^o. Ví dụ, các vector $\begin{bmatrix}2 & 1 & -2 & 4\end{bmatrix}$ và $\begin{bmatrix}3 & -6 & 4 & 2\end{bmatrix}$ là trực giao bởi vì

  $$\begin{bmatrix}2 & 1 & -2 & 4\end{bmatrix}.\begin{bmatrix}3 & -6 & 4 & 2\end{bmatrix} = 2(3) + 1(-6) - 2(4) + 4(2) = 0$$

- Vector Trực chuẩn - Orthonormal Vectors
  Hai vector được gọi là trực chuẩn nếu đó là hai vector đơn vị (có chiều dài vecto là 1) và hai vecor đó trực giao với nhau. Ví dụ $\overrightarrow{u} = \begin{bmatrix}2/5 & 1/5 & -2/5 & 4/5\end{bmatrix}$ và $\overrightarrow{v} = \begin{bmatrix}3/\sqrt{65} & -6/\sqrt{65} & 4/\sqrt{65} & 2/\sqrt{65}\end{bmatrix}$ là hai vecto trực chuẩn với nhau vì

  $$|\overrightarrow{u}| = \sqrt{\left(\frac{2}{5}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{-2}{5}\right)^2 + \left(\frac{4}{5}\right)^2 +} = 1$$

  $$|\overrightarrow{v}| = \sqrt{\left(\frac{3}{\sqrt{65}}\right)^2 + \left(\frac{6}{\sqrt{65}}\right)^2 + \left(\frac{4}{\sqrt{65}}\right)^2 + \left(\frac{2}{\sqrt{65}}\right)^2 +} = 1$$

  $$\overrightarrow{u}.\overrightarrow{v} = \frac{6}{5/\sqrt{65}} - \frac{6}{5/\sqrt{65}} - \frac{8}{5/\sqrt{65}} + \frac{8}{5/\sqrt{65}} = 0$$

- Quy trình trực chuẩn hóa Gram Schmidt - Gram Schmidt Orthonormalization Process
  Gram Schmidt Orthonormalization Process là một phương pháp để chuyển đổi một tập các vectơ thành một tập các vectơ trực chuẩn. Về cơ bản bắt đầu bằng cách chuẩn hóa các vector đầu tiên bằng việc xem xét và lặp đi lặp lại việc viết lại các vectơ còn lại trừ một phép nhân của các vectơ đã chuẩn hóa

  Ví dụ, để chuyển đổi các vector cột của ma trận A

  $$A = \begin{bmatrix}1 & 2 & 1\\ 0 & 2 & 0\\ 2 & 3 & 1\\ 1 & 1 & 0\end{bmatrix}$$

thành các vector cột trực chuẩn

$$A = \begin{bmatrix} \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{6} & \frac{2}{3} \\ 0 & \frac{2\sqrt{2}}{3} & \frac{-1}{3} \\ \frac{\sqrt{6}}{3} & 0 & 0 \\ \frac{\sqrt{6}}{6} & \frac{-\sqrt{2}}{6} & \frac{-\sqrt{2}}{3} \end{bmatrix}$$

Ta chuẩn hóa vector $\overrightarrow{v} = \begin{bmatrix} 1 & 0 & 2 & 1 \end{bmatrix}$ là cột đầu tiên của ma trận A:

$$\overrightarrow{u_1} = \frac{\overrightarrow{v_1}}{|\overrightarrow{v_1}|} = \begin{bmatrix} \frac{1}{\sqrt{6}} & 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

Ta tính $\overrightarrow{w_2}$

$$\begin{aligned} \overrightarrow{w_2} &= \overrightarrow{v_2} - \overrightarrow{u_1}.\overrightarrow{v_2} * \overrightarrow{u_1} \\ &= \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{\sqrt{6}} & 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} . \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix} * \begin{bmatrix} \frac{1}{\sqrt{6}} & 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} \\ &= \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix} - \frac{9}{\sqrt{6}} * \begin{bmatrix} \frac{1}{\sqrt{6}} & 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} \\ &= \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix} - \begin{bmatrix} \frac{3}{2} & 0 & 3 & \frac{3}{2} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2} & 2 & 0 & \frac{-1}{2} \end{bmatrix} \end{aligned}$$

Chuẩn hóa $\overrightarrow{w_2}$ ta có:

$$\overrightarrow{u_2} = \frac{\overrightarrow{w_2}}{|\overrightarrow{w_2}|} = \begin{bmatrix} \frac{\sqrt{2}}{6} & \frac{2\sqrt{2}}{3} & 0 & \frac{-\sqrt{2}}{6} \end{bmatrix}$$

Tổng quát , nếu ta có bộ tập trực chuẩn vector $\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_{k-1}}$ thì $\overrightarrow{u_k}$ được tính bằng công thức

$$\overrightarrow{w_k} = \overrightarrow{w_k} - \sum_{i=1}^{k-1} \overrightarrow{u_i}.\overrightarrow{v_k} * \overrightarrow{u_i}$$

Và trực chuẩn hóa

$$\overrightarrow{u_k} = \frac{\overrightarrow{w_k}}{|\overrightarrow{w_k}|}$$

3.1.4 Các lý thuyết đối với ma trận

- Phép toán trace: Phép toán trace được ký hiệu là "tr" và nó thực hiện phép tính tổng đường chéo của ma trận vuông.

$$trA = \sum_{i=1}^{n} A_{ii}$$

Các tính chất của phép toán trace với A, B, C là ma trận vuông và a là hằng số:

1. $trABC = trCAB = trBCA$
2. $trA = trA^T$
3. $tr(A + B) = trA + trB$
4. $tr(a.A) = a.trA$

- Đạo hàm của hằng số theo ma trận (scalar-by-matrix): Với ma trận $A(m \times n)$

$$A = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

Thì

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{m1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{1n}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

Ví dụ: Cho ma trận A = $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ và $f(A) = A_{11} + 2A_{12}^2 + 3A_{21}A_{22}$ suy ra

$$\nabla_A f(A) = \begin{bmatrix} 1 & 3A_{22} \\ 4A_{12} & 3A_{21} \end{bmatrix}$$

Các tính chất của đạo hàm ma trận với A, B, C là ma trận vuông nxn và x là vector n chiều:

1. $\nabla_A trAB = B^T$
2. $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$
3. $\nabla_A trABA^TC = CAB + C^TAB^T$
4. $\nabla_A |A| = |A|(A^{-1})^T$
5. $\nabla_x(x^TAx) = x^T(A+A^T)$. Nếu ma trận A đối xứng thì $\nabla_x(x^TAx) = 2x^TA$

- Eigenvalues và Eigenvectors:

$$Av = \lambda v$$

Trong đó
A là ma trận $(n \times n)$
v là vector n chiều $(n \times 1)$
λ là một eigenvalue của A
v là một eigenvector của A
Mệnh đề liên quan:

1. Một ma trận A có thể có nhiều eigenvalue và nhiều eigenvector

2. Cho các eigenvector của A là $v_1, v_2, \ldots, v_n$ không phụ thuộc tuyến tính vào nhau, và $\lambda_1, \lambda_2, \ldots, \lambda_n$ là các eigenvalue tương ứng. Khi đó ta có:
$$A = PDP^{-1}$$
Với:
$$P = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$$
$$D = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix}$$

- Ma trận trực giao
Một ma trận A được gọi là trực giao nếu tích của ma trận A và chuyển vị của nó là một ma trận đơn vị. Ví dụ
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & -\frac{4}{5} \\ 0 & \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$
$$AA^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & -\frac{4}{5} \\ 0 & \frac{4}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & \frac{4}{5} \\ 0 & -\frac{4}{5} & \frac{3}{5} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Ma trận đường chéo
Ma trận đường chéo là ma trận có các giá trị $a_{ij} = 0$ nếu $i \neq j$ và các giá trị còn lại của ma trận là khác 0
$$\begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ 0 & a_{22} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix}$$

3.1.5 Phương pháp Lagrangian

Cho bài toán cực đại có ràng buộc: $max_w f(w)$, biết $h_i(w) = 0,\ i = 1, \ldots, l$
Ta định nghĩa Lagrangian là:
$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i . h_i(w)$$
Với β_i là hệ số nhân Lagrange. Để tìm cực đại của bài toán ban đầu ta đi đạo hàm từng phần của $\mathcal{L}(w, \beta)$ và đặt bằng 0, giải phương trình suy ra w và β:
$$\frac{\partial \mathcal{L}}{\partial w} = 0;\ \frac{\partial \mathcal{L}}{\partial \beta_i} = 0$$

3.2 Học máy

3.2.1 Khái niệm cơ bản

3.2.1.1 Học máy

Học máy có hai cách định nghĩa chính và đang được chấp nhận phổ biến:

- Theo Arthur Samuel: "Là một lĩnh vực nghiên cứu mà nó cung cấp cho máy tính khả năng học hỏi mà không cần lập trình một cách tường minh."
- Theo Tom Mitchell: "Một chương trình máy tính được chấp nhận là học hỏi được kinh nghiệm E bằng cách thực hiện một vài tác vụ T theo phép đo hiệu năng P, nếu và chỉ nếu việc thực thi các tác vụ trong T được đo bởi phép đo P đem lại kết quả là kinh nghiệm E được cải thiện."

3.2.1.2 Supervised Learning - Học có giám sát

Chúng ta được cho một tập dữ liệu đã biết với các input và output tương ứng nhau. Ý tưởng là chúng ta sẽ đi tìm mối quan hệ giữa input và output đó chính là Supervised Learning.

Vấn đề của Supervised Learning được phân loại thành hai vấn đề chính là "Regression" và "Classification". Trong vấn đề "Regression", chúng ta sẽ cố gắng dự đoán kết quả output tiếp theo một cách liên tục, nghĩa là chúng ta đi tìm ra một hàm đầu ra liên tục tổng quát với biến là các thuộc tính đầu vào. Còn với vấn đề "Classification", chúng ta thay vì cố gắng dự đoán kết quả liên tục thì ta sẽ đi dự đoán chúng theo hướng rời rạc, hiểu theo một cách khác là chúng ta đi tìm một phép phân loại rời rạc cho output với các biến input.

3.2.1.3 Unsupervised Learning - Học không giám sát

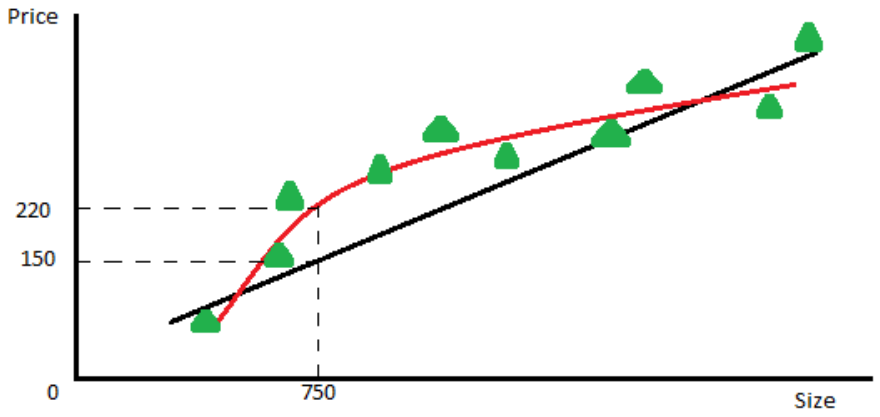
Học không giám sát cho phép chúng ta tiếp cận các vấn đề mà ta chưa hề hoặc biết rất ít kết quả của chúng ta sẽ trông như thế nào. Chúng ta có thể xây dựng cấu trúc của dữ liệu mà không cần thiết phải biết ảnh hưởng của các biến đó.

Chúng ta thực hiện việc này dựa trên ý tưởng gom cụm dữ liệu bằng cách xem xét mối quan hệ giữa các thuộc tính của dữ liệu.

3.2.2 Supervised Learning - Unsupervised Learning

3.2.2.1 Supervised Learning

Bắt đầu với bài toán dự đoán giá nhà.



Hình 3.1: Dự đoán giá nhà

Bạn của bạn muốn thuê một căn phòng với diện tích là $750 feet^2$ và người đó muốn biết giá trị khoảng bao nhiêu. Vậy các thuật toán của Machine Learning sẽ giúp ta như thế nào?

Một điều mà có lẽ thuật toán có thể làm là đặt ra một đường thẳng thông qua dữ liệu, dựa trên đó, có vẻ như giá ngôi nhà có thể là 150.000 $.

Nhưng đó không phải là cách duy nhất mà thuật toán có thể xây dựng. Một cách tốt hơn, thay vì một đường thẳng chúng ta có thể sử dụng một hàm quadratic hoặc một đa thức bậc hai cho dữ liệu. Khi đó hàm sẽ được biểu diễn dưới dạng một đường cong tương đối gần với dữ liệu được cho. Lúc này giá trị của ngôi nhà là 200.000 $.

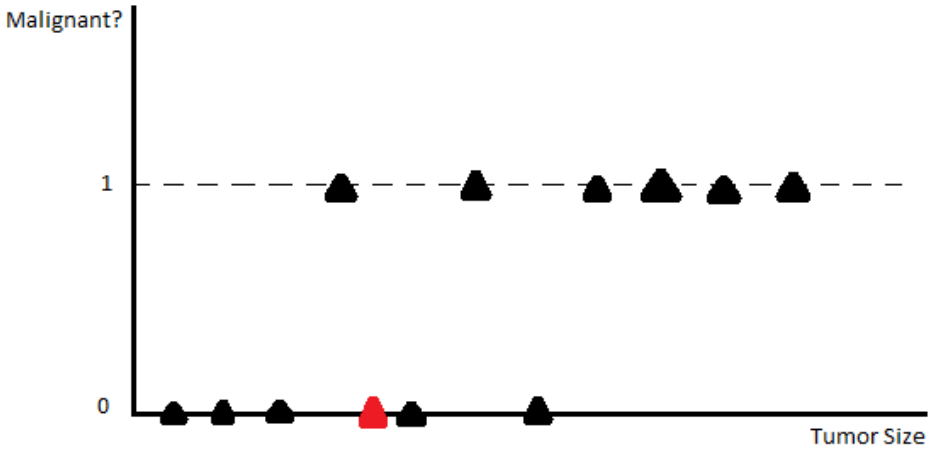
Vậy, quyết định như thế nào để chọn một trong hai phép phân tích trên?

Và đó là một ví dụ về Supervised Learning. Trong một số giới hạn nào đó, Supervised Learing có thể chỉ ra thuật toán mà chúng ta sử dụng là một thuật toán đúng.

- Thuật ngữ Regression (Hồi quy): dự đoán giá trị tiếp theo, hoặc dự đoán thuộc tính của giá trị.

  Một số ví dụ khác về Supervised Learning: Ta đi dự đoán bệnh ung

thư vú là ác tính hay lành tính từ thông tin một hồ sơ y tế.



Hình 3.2: Dự đoán ung thư

Giả sử chúng ta có một người bạn, người bạn ấy có một khối u với kích thước tại vị trí đánh dấu (vị trí màu đỏ). Câu hỏi của Machine Learning là có thể ước lượng xác suất là bao nhiêu? Cơ hội để nó là khối u lành tính là bao nhiêu?

- Thuật ngữ Classification: Sự phân loại. Ta đi làm rõ thuật ngữ này bằng ví dụ sau.

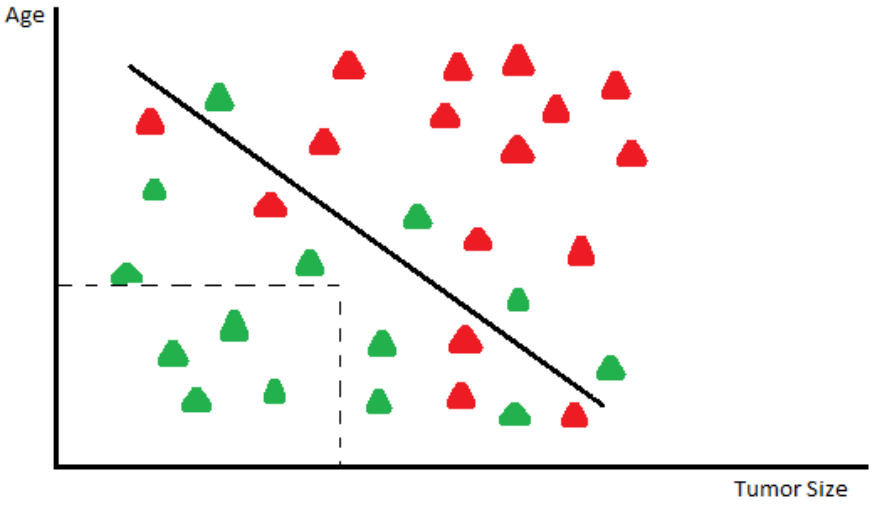
  Giả sử ta đơn giản hóa sơ đồ trên bằng sơ đồ sau:

Hình 3.3: Biễn diễn ung thư theo kích thước khối u

Trong đó:

Màu xanh là lành tính
Màu đỏ là ác tính

Bây giờ, không chỉ còn biết về kích thước khối u mà ta còn biết về độ tuổi người bệnh. Ta hình thành sơ đồ bên dưới:



Hình 3.4: Biểu diễn ung thư theo kích thước khối u và lứt tuổi

Ta có thể dựa vào độ tuổi của người bạn để cho ra kết quả dự đoán. Đó chính là Classification.

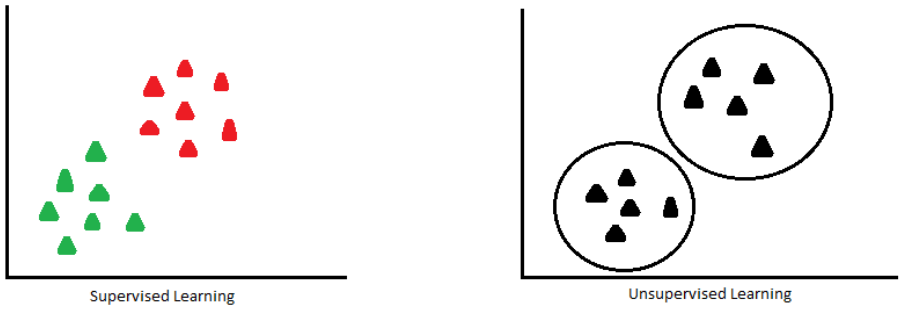
Một vấn đề khác, ở các ví dụ trên ta chỉ dựa vào vài yếu tố, thuộc tính nhưng Machine Learning lại khác, số lượng đầu vào các thuộc tính là rất nhiều. Vậy làm thế nào để lưu khi bộ nhớ máy tính là giới hạn? Ta đi đến với một thuật toán gọi là Support Vector Machine, và điều này sẽ được làm rõ ở các phần sau.

3.2.2.2 Unsupervised Learning

Trong phần trước, Supervised Learning được sỡ hữu một tập dữ liệu mà mỗi giá trị đều được gắn và phân chia nhãn.

Unsupervised Learning được sỡ hữu một tập dữ liệu khác hơn, tất cả đều cùng nhãn hay khác hơn là có thể không có nhãn. Câu hỏi đặt ra của bài toán này là làm sao tìm được cấu trúc của dữ liệu này?

Với dữ liệu trên, Unsupervised Learning sẽ quyết định dữ liệu tồn tại trong hai nhóm, và đó gọi là thuật toán gom cụm - Clustering Algorithm.



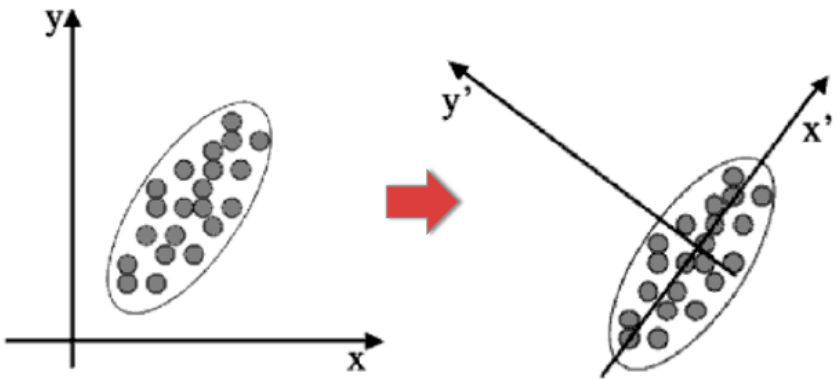
Hình 3.5: Phân biệt supervised và unsupervised

3.3 Các thuật toán thu giảm số chiều

3.3.1 Principal Component Analysis - PCA

Trước khi bắt đầu vào thuật toán, chúng ta sẽ đi tìm hiểu sơ qua về ý tưởng ban đầu của thuật toán này. Với mục đích ban đầu là loại bỏ đi các chiều hay thuộc tính không cần thiết của không gian ma trận dữ liệu đầu vào, ta sẽ đi xác định những thuộc tính nào có tính quan trọng thấp để loại bỏ. Trong thuật toán này, tính quan trọng của một thuộc tính được đo bằng khả năng phân loại của chúng, để rõ ràng hơn ta đi quan sát hình ảnh bên dưới.

(Hình ảnh được trích từ bộ slide môn học Machine Learning – GS. Cao Hoàng Trụ - Đại học Bách Khoa TP.HCM)



Hình 3.6: PCA

Hãy thử tưởng tượng, hình ảnh bên tay trái là bộ dữ liệu đầu vào ban đầu

và gồm hai thuộc tính là x và y. Ta đi thu giảm số chiều, nghĩa là bây giờ ta muốn chỉ cần dùng một thuộc tính để đại diện cho tập dữ liệu. Có hai lựa chọn x' và y' – hình ảnh bên tay phải.

Theo như một cách nghĩ trực quan nhất thì ta thấy việc chiếu theo chiều y' sẽ khiến cho tập dữ liệu bị gom lại rất gần nhau và nó được xem là không có tính phân loại cao. Xem xét việc chiếu theo chiều x', ta dễ dàng nhận thấy tập dữ liệu được kéo dãn ra so với y', do đó, chiều x' được xem là chiều giúp cho tập dữ liệu dù bị loại bớt thuộc tính nhưng vẫn giữ được tính phân loại bên trong nó.

Từ ý tưởng trên, chúng ta sẽ đi xây dựng mô hình toán học. Đơn giản, độ đo gom cụm hay kéo dãn của tập dữ liệu khi chiếu lên một chiều vector sẽ được biểu diễn bằng khái niệm phương sai của tập dữ liệu trên chiều vector đó.

Gọi tập dữ liệu là tập hợp n vector $\{x_i\}_{i=1,...,n}$. Ta có:
Vector trung bình:

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Phương sai của dữ liệu chiếu theo chiều u:

$$\frac{1}{n}\sum_{i=1}^{n}(u^T.x_i - u^T.m)^2 = u^T.S.u$$

Với:

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - m).(x_i - m)^T$$

Và S là ma trận đối xứng.

Bài toán là cần tìm một chiều vector dùng để chiếu xuống có tính phân loại cao đồng nghĩa với việc phương sai của tập dữ liệu trên chiều vector đó cực đại. Vậy nhiệm vụ của chúng ta sẽ đi tìm u để cực đại phương sai:

Bài toán cực đại:

$$\frac{1}{n}\sum_{i=1}^{n}(u^T.x_i - u^T.m)^2 = u^T.S.u$$

Không làm thay đổi tính chất bài toán, ta chọn u là vector đơn vị, nghĩa là:

$$u^T u = ||u||^2 = 1$$

Với bài toán cực đại có điều kiện, ta sẽ sử dụng Lagrangian.

Cực đại hàm Lagrangian:

$$L(u, \lambda) = u^T.S.u + \lambda(1 - u^T u)$$

Đạo hàm theo $L(u, \lambda)$ theo u:

$$\frac{\partial L}{\partial u} = \nabla_u u^T.S.u + \nabla_u \lambda(1 - u^T u) = \nabla_u u^T.S.u - \lambda \nabla_u u^T u$$

Đặt:

$$\nabla_u u^T.S.u \tag{3.1}$$
$$\lambda \nabla_u u^T u \tag{3.2}$$

Xét (3.1):

$$(3.1) = \nabla_u u^T.S.u = 2u^T S$$

Xét (3.2): Giả sử

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \Rightarrow x^T.x = x_1^2 + \cdots + x_n^2 = X \nabla_x x^T x$$
$$= \begin{bmatrix} \frac{\partial X}{\partial x_1} & \dots & \frac{\partial X}{\partial x_n} \end{bmatrix} = \begin{bmatrix} 2x_1 & \dots & 2x_n \end{bmatrix} = 2x^T$$

Suy ra:

$$(3.2) = \lambda \nabla_u u^T u = 2\lambda u^T$$

Từ biến đổi (3.1) và (3.2):

$$\frac{\partial L}{\partial u} = 2u^T S - 2\lambda u^T$$

Đặt vế phải biểu thức bằng 0 ta được:

$$\begin{aligned} & 2u^T S - 2\lambda u^T = 0 \\ & \Leftrightarrow 2u^T S = 2\lambda u^T \\ & \Leftrightarrow (u^T S)^T = (\lambda u^T)^T \\ & \Leftrightarrow Su = \lambda u \end{aligned}$$

Tại đây ta có thể nhận ra, phương sai của tập dữ liệu khi chiếu lên vector u cực đại khi và chỉ khi u là eigenvector của S và λ là eigenvalue của S. Từ đó ta suy ra:

$$S = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}^{-1}$$

Với $u_1, u_2, \ldots, u_n$ là các eigenvector của S và $\lambda_1, \lambda_2, \ldots, \lambda_n$ là các eigenvalue của S tương ứng với eigenvector.

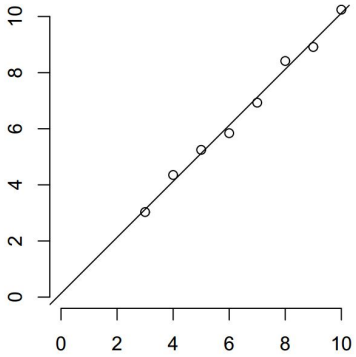
Ta thu giảm không gian vector bây giờ tương ứng với việc ta loại bỏ đi những vector u mà ta cho là không cần thiết. Công việc loại bỏ này được thực hiện bằng cách chọn một giá trị λ làm mốc, sau đó ta loại bỏ những eigenvalue của S có giá trị nhỏ hơn λ và cũng loại bỏ luôn những eigenvector tương ứng.

Vậy, sau khi loại bỏ xong, ta chiếu tập dữ liệu ban đầu lên không gian vector $u_1, u_2, \ldots, u_m$ mới với m $<$ n. Và lúc này ta được một tập dữ liệu, giữ lại được tương đối thông tin của tập dữ liệu ban đầu nhưng đã được thu giảm số chiều vector.

3.3.2 Singular Value Decomposition - SVD

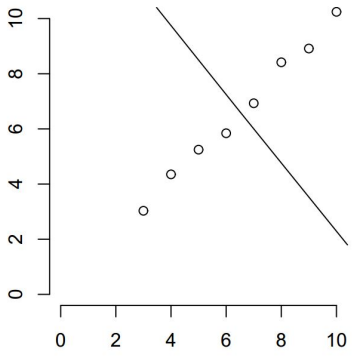
Singular Value Decomposition (SVD) có thể được xem xét từ ba điểm tương thích lẫn nhau của cách nhìn nhận. Một mặt, chúng ta có thể xem nó như là một phương pháp để chuyển biến tương quan vào một tập hợp của biến không tương quan mà trình bày tốt hơn các mối quan hệ khác nhau của dữ liệu gốc. Đồng thời, SVD là một phương pháp để xác định và sắp xếp thứ tự chiều dọc theo đó điểm dữ liệu trình bày tối đa phương sai. Điều này gắn với cách thứ ba của cách nhìn SVD, đó là một khi chúng ta đã xác định được nơi mà tối đa phương sai, nó có thể tìm thấy xấp xỉ tốt nhất của các điểm dữ liệu ban đầu sử dụng ít chiều hơn. Do đó, có thể SVD được xem như một phương pháp để giảm số chiều của dữ liệu.

Hãy xem xét các điểm dữ liệu 2 chiều trong hình 1. Đường hồi quy chạy qua chúng cho thấy xấp xỉ tốt nhất của dữ liệu gốc với một đối tượng 1 chiều (một đường thẳng). Nó là xấp xỉ tốt nhất theo nghĩa rằng nó là đường thẳng tối thiểu khoảng cách giữa mỗi điểm ban đầu và đường thẳng. Nếu chúng ta vẽ một đường thẳng vuông góc từ mỗi điểm đến đường hồi quy, và lấy giao điểm của những đường thẳng đó như xấp xỉ của các điểm dữ liệu ban đầu, chúng ta sẽ có một đại diện của dữ liệu gốc đã được thu giảm số chiều mà thu thập càng nhiều các phương sai gốc càng tốt. Lưu ý rằng có một đường hồi quy thứ hai, vuông góc với dữ liệu ban đầu, thể hiện trong hình 2. Đường thẳng này thu thập càng nhiều các phương sai nhất có thể theo chiều thứ hai của tập dữ liệu gốc. Nó thực hiện xấp xỉ dữ liệu ban đầu kém hơn so với hình 1, vì nó tương ứng với trình bày dữ liệu một chiều ít phương sai hơn so với ban đầu. Có thể sử dụng các đường hồi quy để tạo ra một tập hợp các điểm dữ liệu không tương quan sẽ hiển thị các tập con trong dữ liệu gốc không nhất thiết phải hiển thị ở cái nhìn đầu tiên.



Hình 3.7: Đường hồi quy phù hợp nhất giảm dữ liệu từ hai chiều thành một

Những ý tưởng cơ bản đằng sau SVD là: Ta lấy một bộ dữ liệu biến thiên rất cao và có số chiều lớn, thu giảm nó đến một không gian chiều thấp hơn cho thấy nhiều phần cấu trúc của các dữ liệu ban đầu rõ ràng hơn và thứ tự của nó từ những phương sai nhiều nhất đến ít nhất. Điều làm cho SVD thiết thực trong các ứng dụng xử lí ngôn ngữ tự nhiên chỉ đơn giản là có thể bỏ qua sự thay đổi dưới một ngưỡng cụ thể để giảm lượng lớn dữ liệu, nhưng vẫn đảm bảo rằng mối quan hệ chính của tỉ lệ được bảo đảm.



Hình 3.8: Đường hồi quy theo chiều hướng thứ hai thu thập ít sự thay đổi phương sai trong dữ liệu gốc

3.3.2.1 Thuật toán

SVD được dựa trên một định lý từ đại số tuyến tính rằng một ma trận chữ nhật A có thể được chia thành các tích của ba ma trận theo thứ tự - một ma trận trực giao U, một ma trận đường chéo S, và chuyển vị của một ma trận trực giao V. Định lý thường được trình bày theo công thức

$$A_{mn} = U_{mm} S_{mn} V_{nn}^{T}$$

mà $U^TU = I, V^TV = I$, trong đó các cột của ma trận U là eigenvectors trực giao của tích hai ma trận A và A^T, các cột của ma trận V là eigenvectors trực giao của tích hai ma trận A^T và A, và S là một ma trận đường chéo chứa các căn bậc hai của các eigenvalue của ma trận U hoặc ma trận V theo thứ tự giảm dần.

Các bước để hiện thực SVD
Bước 1: Từ ma trận thuộc tính A ban đầu ta tính ma trận trực giao U

- Tính tích hai ma trận theo thứ tự là A và A^T.
- Tìm eigenvalues và eigenvectors tương ứng của tích hai ma trận A và A^T.
- Các eigenvectors trở thành vector cột trong một ma trận được sắp xếp theo kích thước của eigenvalue tương ứng. Nói cách khác, các eigenvectors của eigenvalue lớn nhất là cột một, các eigenvectors của eigenvalue lớn thứ hai theo là cột hai,... và như vậy cho đến khi có các eigenvectors của eigenvalue nhỏ nhất như cột cuối cùng của ma trận.
- Cuối cùng, chúng ta phải chuyển đổi ma trận này thành một ma trận trực giao bằng cách áp dụng quá trình trực chuẩn hóa Gram-Smith với vector cột.

Bước 2: Từ ma trận thuộc tính A ban đầu ta tính ma trận trực giao V giống như tính toán ma trận trực giao U tuy nhiên ta sẽ tính bằng tích hai ma trận theo thứ tự là A^T và A (của U là A và A^T).
Bước 3: Đối với ma trận S ta lấy căn bậc hai của các eigenvalues khác không và đặt trong đường chéo của ma trận S, eigenvalues lớn nhất là s_{11}, eigenvalues tiếp theo là s_{22} và cứ thế cho đến giá trị eigenvalues nhỏ nhất là s_{mm}. Các eigenvalues khác không của ma trận U và V là luôn luôn giống nhau, vì vậy đó là lý do tại sao nó không quan trọng trong việc chúng ta lấy eigenvalues của ma trận nào. Tất cả các giá trị còn lại của ma trận S là giá trị không.
Bước 4: Thu giảm số chiều của ba ma trận U, S và V.

3.3.2.2 Thu giảm số chiều của ma trận thuộc tính

S chứa căn bậc hai của các giá trị theo thứ tự từ lớn nhất đến nhỏ nhất của nó theo đường chéo. Những giá trị này cho thấy phương sai của các thành phần độc lập tuyến tính dọc theo mỗi chiều. Ta thực hiện loại bỏ các giá trị nhỏ theo đường chéo của ma trận bằng cách loại bỏ toàn bộ hàng chứa giá trị đó. Để cho phép nhân ma trận, chúng ta phải loại bỏ sự tương ứng vector hàng của U và vector cột tương ứng của V^T để được kết quả là xấp xỉ của ma trận A ban đầu.

Đây là một ví dụ cụ thể

$$A = \begin{bmatrix} 2 & 0 & 8 & 6 & 0 \\ 1 & 6 & 0 & 1 & 7 \\ 5 & 0 & 7 & 4 & 0 \\ 7 & 0 & 8 & 5 & 0 \\ 7 & 0 & 8 & 5 & 0 \end{bmatrix}$$

Ta đưa ma trận A thành USV^T

$$U = \begin{bmatrix} -0.54 & 0.07 & 0.82 & -0.11 & 0.12 \\ -0.10 & -0.59 & -0.11 & -0.79 & -0.06 \\ -0.53 & 0.06 & -0.21 & 0.12 & -0.81 \\ -0.53 & 0.06 & -0.21 & 0.12 & -0.81 \\ -0.06 & -0.80 & 0.09 & 0.59 & 0.04 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.46 & 0.02 & -0.87 & 0 & 0.17 \\ -0.07 & -0.76 & 0.06 & 0.60 & 0.23 \\ -0.07 & -0.76 & 0.06 & 0.60 & 0.23 \\ -0.48 & 0.03 & 0.40 & -0.33 & 0.70 \\ -0.07 & -0.64 & -0.04 & -0.69 & -0.32 \end{bmatrix}$$

Eigenvalue trong quá trình tính toán là $\lambda = 321.07, \lambda = 230.17, \lambda = 12.70, \lambda = 3.94, \lambda = 0.12$. Ta sẽ có ma trận S là ma trận có đường chéo là các eigenvalue theo thứ tự từ lớn đến nhỏ. Để minh họa ảnh hưởng của giảm chiều trên tập dữ liệu này, chúng ta sẽ hạn chế S với ba giá trị số ít đầu tiên để có được

$$\begin{bmatrix} 17.92 & 0 & 0 \\ 0 & 15.17 & 0 \\ 0 & 0 & 3.56 \end{bmatrix}$$

Ta thu giảm tương ứng số cột của ma trận U và số hàng của ma trận V^T xuống bằng số hàng của ma trận S mà tính phép nhân ma trận ta có được ma trận $\widehat{A}$ xấp xỉ ma trận A bằng

$$\widehat{A} = \begin{bmatrix} -0.54 & 0.07 & 0.82 \\ -0.10 & -0.59 & -0.11 \\ -0.53 & 0.06 & -0.21 \\ -0.53 & 0.06 & -0.21 \\ -0.06 & -0.80 & 0.09 \end{bmatrix} \begin{bmatrix} 17.92 & 0 & 0 \\ 0 & 15.17 & 0 \\ 0 & 0 & 3.56 \end{bmatrix} \begin{bmatrix} -0.46 & 0.02 & -0.87 & 0 & 0.17 \\ -0.07 & -0.76 & 0.06 & 0.60 & 0.23 \\ -0.74 & 0.10 & 0.28 & 0.22 & -0.56 \end{bmatrix}$$

$$= \begin{bmatrix} 2.29 & -0.66 & 9.33 & 1.25 & -3.09 \\ 1.77 & 6.76 & 0.90 & -5.50 & -2.13 \\ 4.86 & -0.96 & 8.01 & 0.38 & -0.97 \\ 6.62 & -1.23 & 9.58 & 0.24 & -0.71 \\ 1.14 & 9.19 & 0.33 & -7.19 & -3.13 \end{bmatrix}$$

Tuy nhiên, trong thực tế, mục đích không phải là để tái tạo lại các ma trận ban đầu, nhưng sử dụng để giảm số chiều đại diện để xác định thuộc tính và các dữ liệu đầu vào tương tự. Những dữ liệu đầu vào (observation) được đại diện bởi các vector hàng trong ma trận V, và dữ liệu đầu vào tương tự thu được bằng cách so sánh các hàng trong tích ma trận V.S (lưu ý rằng các dữ liệu đầu vào được biểu diễn như là vector hàng bởi vì chúng ta đang làm việc với V, không phải V^T). thuộc tính được biểu diễn bằng vector hàng trong U, và thuộc tính tương tự có thể thu được bằng cách tính tương tự hàng ở tích ma trận U.S. Thay vì xử lí thuộc tính của dữ liệu trên một ma trận có 5 cột như ma trận A ban đầu, giờ đây ta chỉ cần xử lí thuộc tính của dữ liệu trên một ma trận có giá trị xấp xỉ ma trận A ban đầu nhưng chỉ có 3 cột.

Chúng ta biết rằng các vector chứa phần tử theo thứ tự từ lớn nhất đến ít nhất của phương sai trong dữ liệu gốc. Bằng cách xóa các phần tử đại diện cho số chiều mà không thể hiện phương sai có nghĩa, chúng ta có thể loại bỏ sự nhiễu dữ liệu hiệu quả trong thuộc tính. Bây giờ vector thuộc tính là ngắn hơn, và chỉ chứa các phần tử giải thích cho sự tương quan đáng kể nhất trong số các thuộc trong tập dữ liệu ban đầu.

3.4 Tổng quan về hệ thống phát hiện xâm nhập trên host - HIDS

3.4.1 Khái niệm về hệ thống phát hiện xâm nhập - IDS

IDS là viết tắt của từ Intrusion detection system, có nghĩa là hệ thống phát hiện xâm nhập. Hệ thống phát hiện xâm nhập là hệ thống được phát triển nhằm mục đích bảo vệ máy tính khỏi sự xâm nhập trái phép hoặc sự sử dụng trái với quy định.

Hệ thống này có thể là phần mềm cài đặt trên máy tính, thiết bị mạng, hoặc là thiết bị phần cứng đặt trên đường truyền để giám sát đường truyền mạng.

Hệ thống phát hiện xâm nhập (IDS) chỉ thực hiện công việc giám sát các hoạt động và đưa ra cảnh báo cho admin. Admin sẽ dựa vào cảnh báo từ hệ thống IDS để xác định hành động kế tiếp.

Hệ thống IDS không có chức năng tự động ngắt bỏ hay ngăn chặn hành động trái phép, thay vào đó là hệ thống IPS (Intrusion prevention system), hệ thống ngăn chặn xâm nhập sẽ thực hiện công việc này.

Hệ thống IDS có 2 loại: HIDS và NIDS.

3.4.2 Hệ thống phát hiện xâm nhập trên mạng - NIDS

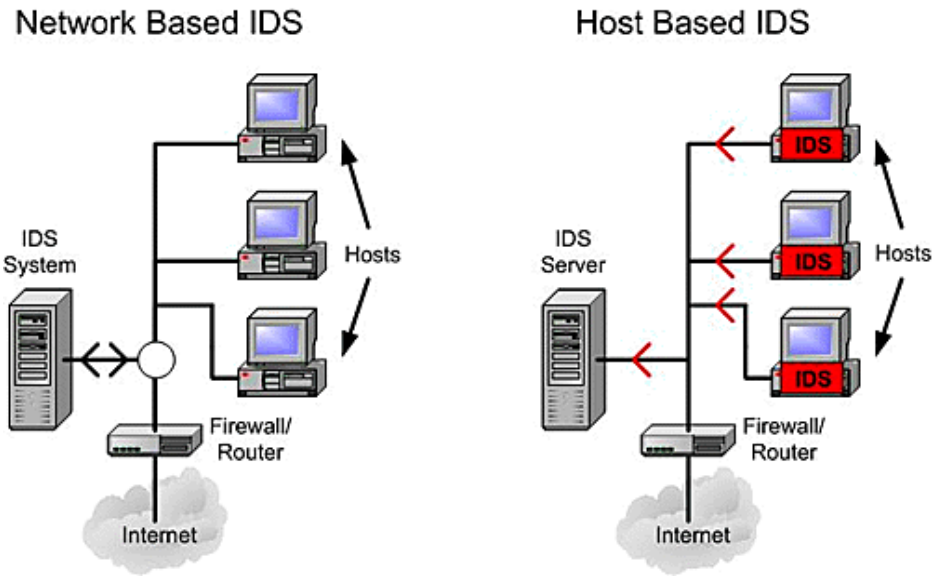
NIDS là viết tắt của từ Network-base Intrusion Detection System, có nghĩa là hệ thống phát hiện xâm nhập trên network. Hệ thống này thường được cài đặt trên các thiết bị mạng là gateway của mạng nội bộ, hoặc là thiết bị phần cứng lắp đặt trên đường truyền giữa mạng ngoài và mạng nội bộ.

Chức năng chính của hệ thống này là giám sát hoạt động mạng của toàn bộ các thiết bị nằm trong phần mạng nội bộ với mạng ngoài, mục đích là để bảo vệ toàn bộ mạng nội bộ khỏi các cuộc tấn công mạng và các virus.

Dữ liệu kiểm tra của NIDS là các gói tin truyền trên mạng, NIDS thực hiện kiểm tra các giao thức, header gói tin, nội dung gói tin để tìm ra các dấu hiệu khả nghi.

Nhưng việc kiểm tra các gói tin trên đường truyền mạng có các bất cập sau:

- Dữ liệu truyền nhận có thể đã bị mã hóa do các giao thức bảo mật, như vậy việc rút trích thông tin để phát hiện xâm nhập là vô cùng khó.
- Gói tin trên đường truyền có thể bị phân mảnh, nội dung gói tin không còn hoàn chỉnh, như vậy việc phân tích gói tin cũng không khả thi.



Hình 3.9: So sánh giữa NIDS và HIDS

3.4.3 Hệ thống phát hiện xâm nhập trên host - HIDS

HIDS là viết tắt của từ Host-base Intrusion Detection System, có nghĩa là hệ thống phát hiện xâm nhập trên host. Hệ thống này được cài đặt trên từng máy tính.

Chức năng chính của hệ thống này là giám sát hoạt động của 1 máy tính riêng lẻ, kể cả các dữ liệu mạng cũng như các hoạt động trong bản thân máy tính, kiểm tra hoạt động của máy tính để phát hiện sự xâm nhập và cảnh báo cho admin.

Dữ liệu kiểm tra của HIDS là các gói tin hoàn chỉnh, các lời gọi hàm (system call), các file log trong máy tính. HIDS thực hiển kiểm tra các header, địa chỉ IP gói tin, cũng như nội dung gói tin, ngoài ra còn có giám sát việc thực thi chương trình thông qua lời gọi hàm và các file log.

3.4.4 Hình thức phát hiện xâm nhập

Cả 2 loại IDS đều có 2 hình thức phát hiện xâm nhập sau:

1. Signature base: Kiểu phát hiện xâm nhập dựa trên đặc điểm nhận dạng của đối tượng. Trong hình thức này, tất cả mọi kiểu xâm nhập đều đã được đưa ra các đặc điểm nhận dạng và lưu vào trong cơ sở dữ liệu trong hệ thống. Hệ thống khi bắt được 1 sự kiện sẽ so sánh với tập dữ liệu nhận dạng này, nếu trùng với đặc điểm nhận dạng thì hệ thống sẽ báo ngay.

    - Ưu điểm: Có tốc độ nhanh, do chỉ việc so sánh trùng với dữ liệu.
    - Nhược điểm: Đối với kiểu xâm nhập chưa được nhận diện thì không phát hiện được, nên phải thường xuyên cập nhật các dấu hiệu nhận dạng của đối tượng xâm nhập.

2. Behavior base: Kiểu phát hiện xâm nhập dựa trên hành vi của chương trình. Trong hình thức này, các hoạt động của chương trình được lưu lại. Một khi có những hoạt động không bình thường trong chương trình, hệ thống sẽ kích hoạt cảnh báo. Các hoạt động bất thường có thể là sự tăng đột ngột số request tới server, công suất CPU bị đẩy lên tối đa tại 1 thời điểm.

    - Ưu điểm: Có thể phát hiện ra những kiểu xâm nhập mà signature base không phát hiện được.
    - Khuyết điểm: Cơ hội nhận diện sai thường sẽ cao hơn, do việc so sánh hành vi không thật sự chính xác.

3.4.5 Các chức năng chính trong HIDS

- Kiểm tra tính toàn vẹn của file: Sử dụng các hàm mật mã như hàm hash để tạo ra chuỗi checksum cho file, sau đó theo thời gian định trước so sánh checksum của file với checksum cũ để kiểm tra tính toàn vẹn.

- Giám sát thanh ghi trên hệ điều hành Windows: Hệ thống thanh ghi trên hệ điều hành Windows chứa các cấu hình về thiết bị phần cứng cũng như phần mềm của máy, mỗi lần có sự thay đổi trong hệ thống thanh ghi này đều được HIDS ghi lại để phát hiện những hành động trái quy định.

- Kiểm tra rootkit: Rootkit là các chương trình được cài ẩn trong hệ thống mà người dùng không biết, rootkit được sử dụng để tạo ra các dịch vụ, chương trình chạy ngầm, nhằm mục đích là để cho Intruders có thể khống chế máy tình hoặc làm việc theo ý muốn. Hệ thống HIDS có cơ chế kiểm tra việc cài đặt và thực thi cũng những chương trình này.

- Active response: Active response là việc 1 chương trình thực thi sẽ kích hoạt thực thi của chương trình khác, cơ chế này giúp cho HIDS sau khi phát hiện xâm nhập có thể có những hành động kịp thời để bảo vệ máy tính.

Chương 4

Các giải thuật gom cụm

4.1 Giải thuật K-Means

Vì giải thuật K-means khá đơn giản và có thể hữu hình hóa nên chúng ta sẽ đi thẳng vào thuật toán để xem các bước thực hiện của giải thuật K-means như thế nào, từ đó ta cũng có thể hiểu được cách tiếp cận vấn đề của giải thuật.

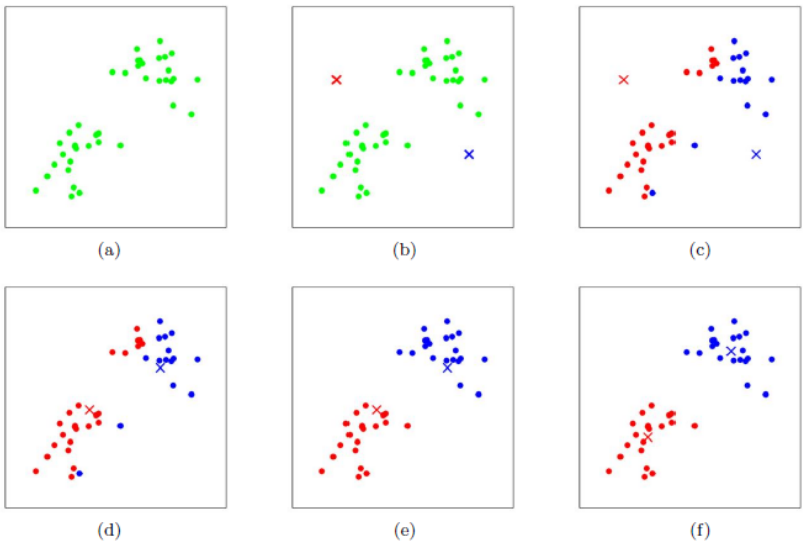
Các bước của giải thuật K-means được thực hiện theo trình tự:

1. Gọi k là số cụm mong muốn, bắt đầu với việc chọn ngẫu nhiên các vector "trọng tâm" của k nhóm $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$.

2. Lặp cho đến khi hội tụ: {

   Với mỗi i, tính:
   $$c^{(i)} := arg\, min_j ||x^{(i)} - \mu_j||^2$$
   Với mỗi j, tính:
   $$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$
   }

   Dưới đây là hình mô phỏng cách thuật toán làm việc, với ví dụ điển hình là vector 2 chiều (Hình ảnh được trích từ CS229 Lecture Notes – 7a. The k-means clustering algorithm – Andrew Ng):



Hình 4.1: Thuật toán K-means với vector 2 chiều

Giải thích hình trên:

(a) Ta có tập dữ liệu đầu vào không nhãn, ta muốn chia tập dữ liệu này thành hai cụm.

(b) Ta chọn hai vector "trọng tâm" ban đầu bất kỳ, mỗi vector đại diện cho một cụm và được biểu diễn bằng điểm x trên đồ thị. Đây chính là bước (a) của thuật toán.

(c) Ở đây, ta giả sử vector "trọng tâm" nằm góc trên bên phải có giá trị μ_1 đại diện cho nhãn $\{c^{(i)} = 1\}$ và vector "trọng tâm" nằm góc dưới bên phải có giá trị μ_2 đại diện cho nhãn $\{c^{(i)} = 2\}$. Bắt đầu vào vòng lặp đầu tiên của bước (b). Thuật toán đi tính khoảng cách Euclid của tất cả các vector phần tử trong tập dữ liệu đến hai vector "trọng tâm". Phần tử được đánh nhãn $\{c^{(i)} = 1\}$ khi khoảng cách Euclid đến vector "trọng tâm" μ_1 gần hơn so với vector "trọng tâm" μ_2 hoặc ngược lại. Cứ tiếp tục như vậy ta sẽ gán nhãn cho tất cả các phần tử trong tập dữ liệu đầu vào.

(d) Ta cập nhật lại giá trị của vector "trọng tâm" bằng cách gán $\mu_j (j = 1 \vee 2)$ bằng với giá trị trung bình của các phần tử thuộc nhãn do chính nó sở hữu. Đây chính là vòng lặp thứ 2 trong bước (b).

(e) Ta quay lại từ đầu của bước (b).

(f) Tiếp tục cập nhật cho đến khi hội tụ.

4.2 Giải thuật kết hợp Gaussian và EM (Mixture of Gaussians and the EM Algorithm)

Để giải quyết bài toán phân cụm, ta sẽ tiếp cận với một giải thuật dựa trên nguyên lý của phân phối Gaussian và thuật toán EM – Expectation Maximization.

Giả sử ta có bộ dữ liệu đầu vào được biểu diễn dưới dạng $\{x^{(1)}, \ldots, x^{(m)}\}$, yêu cầu của bài toán là đưa ra kết quả với mỗi phần tử trong tập dữ liệu được phân thành các cụm với các phần tử khác có "tính chất tương tự nhau". Lưu ý, tập dữ liệu của chúng ta không tồn tại nhãn.

Bắt đầu với "động lực" ban đầu của giải thuật, ta mong muốn chia tập dữ liệu thành k nhóm và mỗi nhóm được đại diện bởi một mô hình phân phối xác suất. Ở đây, nhận thấy mô hình phân phối Gaussian là thích hợp, vì giới hạn của báo cáo, chúng ta sẽ khoan xem xét đến việc tại sao mô hình phân phối Gaussian được chọn? Và nó thích hợp như thế nào? Mà thay vào đó sẽ đi làm rõ cách mà mô hình phân hình Gaussian tham số hóa cho từng cụm dữ liệu của ta.

Xét xác suất $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$. Trong công thức này ta có $p(x^{(i)}|z^{(i)})$ là xác suất để phần tử $x^{(i)}$ thuộc vào cụm được đánh nhãn $z^{(i)}$ và $p(z^{(i)})$ là xác suất để cụm được gán nhãn $z^{(i)}$.

Từ cách giải thích đó, ta có được $z^{(i)} \sim Multinomial(\phi)$ (Multinominal – Phân phối đa thức) với $p(z^{(i)} = j) = \phi_j \geq 0$, $\sum_{j-1}^{k} \phi_j = 1$, và $(x^{(i)}|z^{(i)} = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$.

Mặc khác, như đã nó bên trên, vì $(x^{(i)}|z^{(i)} = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$ nên ta có thể viết lại xác suất ban đầu dưới dạng phụ thuộc vào μ_j, Σ_j:

$$p(x^{(i)}, z^{(i)}) = p(x^{(i)};\, \phi,\, \mu,\, \Sigma)$$

Tiếp tục phân tích:

$$p(x^{(i)};\, \phi,\, \mu,\, \Sigma) = \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)};\, \mu,\, \Sigma)p(z^{(i)};\, \phi)$$

Tại đây ta thấy được bài toán thực sự được sử dụng Mixtures of Gaussian như thế nào vào việc mô hình hóa.

Công việc tiếp theo chúng ta sẽ đi cực đại giá trị xác suất $p(x^{(i)};\, \phi,\, \mu,\, \Sigma)$ cho từng cụm – vì ta giả sử có k cụm nên việc này đồng nghĩa ta phải cực

đại cho k xác suất như vậy. Việc này cũng có ý nghĩa là, khi ta xét một phần tử bất kỳ ta sẽ đi tính xác suất của điểm này cho k cụm, nghĩa là thử nó với từng mô hình xác suất, và nó sẽ được kết luận thuộc cụm có giá trị xác suất là cao nhất.

Để giải quyết bài toán cực đại ta sử dụng thuật toán Maximum Likelihood. Ta có biểu thức likelihood của xác suất ban đầu với m là số phần tử của tập đầu vào:

$$l(\phi,\ \mu,\ \Sigma) = \sum_{i=1}^{m} \log\ p(x^{(i)};\ \phi,\ \mu,\ \Sigma) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)};\ \mu,\ \Sigma)p(z^{(i)};\ \phi)$$

Tạm dừng bài toán tại đây. Chúng ta sẽ chuyển sang Giải thuật EM, để cung cấp cho chúng ta một công cụ có thể giúp tiếp tục giải quyết bài toán.

Định nghĩa hàm lồi – lõm: Hàm f được gọi là lồi khi $f'' \geq 0$. Ngược lại, $f'' \leq 0$ thì f được gọi là hàm lõm.

Bất đẳng thức Jensen: Cho hàm lồi f, với bất kỳ X ta luôn có:

$$E[f(X)] \geq f[E(X)]$$

Điều kiện đẳng thức xảy ra khi $X = E[X]$

Ta có thể suy rộng ra cho trường hợp f là hàm lõm, dấu của bất đẳng thức sẽ được đảo chiều:

$$E[f(X)] \leq f[E(X)]$$

Điều kiện đẳng thức vẫn giữ nguyên.

Xét $f(x) = \log(x)$ và $x = \frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}$. Khoan hãy nghi vấn tại sao chúng ta đặt như vậy mà hãy xem phép biến đổi dưới đây để hiểu rõ hơn:

$$\sum_i \log\ p(x^{(i)};\ \theta) = \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\ \theta) = \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\ \theta)}{Q_i(z^{(i)})}$$
$$\geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)})\ \log \frac{p(x^{(i)}, z^{(i)};\ \theta)}{Q_i(z^{(i)})}$$

Tới đây, ta thấy được việc đặt ở trên để đưa về dạng chuẩn của bất đẳng thức Jensen cho hàm lõm.

Bây giờ ta đã có đầy đủ công cụ để giải quyết tiếp bài toán ban đầu.

Ta đa dừng lại ở việc phải đi cực đại biểu thức Likelihood:

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}; \phi, \mu, \Sigma) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)} = j; \mu, \Sigma) p(z^{(i)=j}; \phi)$$

Với $w_j^{(i)} = P(z^{(i)} = j|x^{(i)};\ phi,\ \mu,\ \Sigma)$, ta có:

$$\begin{aligned}
\sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi) &= \sum_{i=1}^{m} \log \sum_{j=1}^{k} w_j^{(i)} \frac{p(x^{(i)}|z^{(i)};\ \mu,\ \Sigma) p(z^{(i)};\ \phi)}{w_j^{(i)}} \\
&\geq \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\right) .\phi_j}{w_j^{(i)}}
\end{aligned}$$

Để cực đại biểu thức bên phải, chúng ta thực hiện bằng cách đạo hàm từng phần lần lượt với μ, ϕ và Σ sau đó đặt bằng 0, từ biểu thức thu được ta giải nghiệm.

Bắt đầu với đạo hàm riêng theo μ_l với $l = 1, \ldots, k$

$$\begin{aligned}
&\nabla_{u_l} \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\right) .\phi_j}{w_j^{(i)}} \\
&= \nabla_{u_l} \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} \left(\log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \phi_j}{w_j^{(i)}} - \frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j) \right) \\
&= \nabla_{u_l} \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \phi_j}{w_j^{(i)}} - \frac{1}{2} \nabla_{u_l} \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} (x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j) \\
&= -\frac{1}{2} \nabla_{u_l} \sum_{i=1}^{m} \sum_{j=1}^{k} w_j^{(i)} (x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j) \\
&= -\frac{1}{2} \sum_{i=1}^{m} \nabla_{u_l} \sum_{j=1}^{k} w_j^{(i)} (x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j) \\
&= -\frac{1}{2} \sum_{i=1}^{m} \nabla_{u_l} w_l^{(i)} (x^{(i)} - \mu_l)^T \Sigma_l^{-1} (x^{(i)} - \mu_l) \\
&= -\frac{1}{2} \sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l} (x^{(i)^T} - \mu_l^T) \Sigma_l^{-1} (x^{(i)} - \mu_l) \\
&= -\frac{1}{2} \sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l} (x^{(i)^T} \Sigma_l^{-1} x^{(i)} - \mu_l^T \Sigma_l^{-1} x^{(i)} - x^{(i)^T} \Sigma_l^{-1} \mu_l + \mu_l^T \Sigma_l^{-1} \mu_l) \\
&= -\frac{1}{2} \sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l} (-\mu_l^T \Sigma_l^{-1} x^{(i)} - x^{(i)^T} \Sigma_l^{-1} \mu_l + \mu_l^T \Sigma_l^{-1} \mu_l)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l}(\mu_l^T \Sigma_l^{-1} x^{(i)} + x^{(i)^T}\Sigma_l^{-1}\mu_l - \mu_l^T\Sigma_l^{-1}\mu_l) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l} tr(\mu_l^T \Sigma_l^{-1} x^{(i)} + x^{(i)^T}\Sigma_l^{-1}\mu_l - \mu_l^T\Sigma_l^{-1}\mu_l) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l}(tr(\mu_l^T \Sigma_l^{-1} x^{(i)}) + tr(x^{(i)^T}\Sigma_l^{-1}\mu_l) - tr(\mu_l^T\Sigma_l^{-1}\mu_l)) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l}(tr(\mu_l^T \Sigma_l^{-1} x^{(i)}) + tr(x^{(i)^T}\Sigma_l^{-1}\mu_l)^T - tr(\mu_l^T\Sigma_l^{-1}\mu_l)) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l}(2tr(\mu_l^T \Sigma_l^{-1} x^{(i)}) - tr(\mu_l^T\Sigma_l^{-1}\mu_l)) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{u_l}(2(\mu_l^T \Sigma_l^{-1} x^{(i)}) - (\mu_l^T\Sigma_l^{-1}\mu_l)) \\
&= \sum_{i=1}^{m} w_l^{(i)} (\Sigma_l^{-1} x^{(i)} - \Sigma_l^{-1}\mu_l)
\end{aligned}
$$

Đặt $\sum_{i=1}^{m} w_l^{(i)}(\Sigma_l^{-1}x^{(i)} - \Sigma_l^{-1}\mu_l) = 0$. Giải phương trình ta được:

$$\mu_l = \frac{\sum_{i=1}^{m} w_l^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_l^{(i)}}$$

Tiếp tục với đạo hàm riêng theo ϕ_j. Ta nhận thấy biểu thức chỉ có một biến duy nhất phụ thuộc vào ϕ_j , ta dễ dàng viết được:

$$
\begin{aligned}
&\nabla_{\phi_j} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1}(x^{(i)} - \mu_j)\right).\phi_j}{w_j^{(i)}} \\
&= \nabla_{\phi_j} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \phi_j
\end{aligned}
$$

Điều này đồng nghĩa với việc bài toán cực đại ban đầu theo ϕ_j sẽ tương ứng với việc đi cực đại $\sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \phi_j$ theo ϕ_j.

Tuy nhiên, ϕ_j chịu ràng buộc $\sum_{j=1}^{k} \phi_j = 1$ nên ta cần dùng đến kỹ thuật Lagrangian:

Đặt:

$$\mathcal{L}(\phi) = \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \phi_j + \beta\left(\sum_{j=1}^{k} \phi_j - 1\right)$$

Để cực đại $\mathcal{L}(\phi)$ ta đơn giản chi đi đạo hàm riêng theo ϕ_j

$$\frac{\partial}{\partial \phi_j}\mathcal{L}(\phi) = \sum_{i=1}^{m} \frac{w_j^{(i)}}{\phi_j} + \beta$$

Giải phương trình $\sum_{i=1}^{m} \frac{w_j^{(i)}}{\phi_j} + \beta = 0$ ra được $\phi_j = -\frac{1}{\beta}\sum_{i=1}^{m} w_j^{(i)}$.

Ta nhận thấy:

$$\phi_j = -\frac{1}{\beta}\sum_{i=1}^{m} w_j^{(i)} \Rightarrow \sum_{j=1}^{k}\phi_j = -\frac{1}{\beta}\sum_{j=1}^{k}\sum_{i=1}^{m} w_j^{(i)}$$

Mặt khác:

$$\sum_{j=1}^{k}\phi_j = 1 \Rightarrow -\beta = \sum_{j=1}^{k}\sum_{i=1}^{m} w_j^{(i)} = \sum_{i=1}^{m} 1 = m$$

Vậy:

$$\phi_j = \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)}$$

Cuối cùng với đạo hàm riêng theo Σ_l . Tuy nhiên việc đạo hàm riêng theo Σ_l sẽ dẫn đến các biến đổi toán học phức tạp. Tiếp cận với một hướng khác, ta đạo hàm riêng theo Σ_l^{-1}, việc này vẫn giữ được bản chất vấn đề nhưng lại dẫn đến các biến đổi toán học dễ dàng hơn so với hướng tiếp cận ban đầu.

$$\begin{aligned}
&\nabla_{\Sigma_l^{-1}} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_j)^T\Sigma_j^{-1}(x^{(i)}-\mu_j)\right).\phi_j}{w_j^{(i)}} \\
&= \sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \left(\log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_l|^{1/2}}\phi_l}{w_l^{(i)}} - \frac{1}{2}(x^{(i)}-\mu_l)^T\Sigma_l^{-1}(x^{(i)}-\mu_l) \right)
\end{aligned}$$

Đặt:

$$\sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_l|^{1/2}}\phi_l}{w_l^{(i)}} \tag{4.1}$$

$$\sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \left(-\frac{1}{2}(x^{(i)}-\mu_l)^T\Sigma_l^{-1}(x^{(i)}-\mu_l) \right) \tag{4.2}$$

Biến đổi (4.1):

$$(4.1) = \sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \left(\log \frac{\frac{1}{(2\pi)^{n/2}}\phi_l}{w_l^{(i)}} - \log|\Sigma_l|^{1/2} \right) = \sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \left(-\log|\Sigma_l|^{1/2} \right)$$

$$= -\frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} - \log|\Sigma_l| = -\frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} - \log\frac{1}{|\Sigma_l^{-1}|}$$

$$= -\frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \frac{\nabla_{\Sigma_l^{-1}} \frac{1}{|\Sigma_l^{-1}|}}{\frac{1}{|\Sigma_l^{-1}|}} = \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \frac{\frac{\nabla_{\Sigma_l^{-1}} \Sigma^{-1}}{|\Sigma_l^{-1}|^2}}{\frac{1}{|\Sigma_l^{-1}|}}$$

$$= \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \frac{|\Sigma_l^{-1}||\Sigma_l|^T}{|\Sigma_l^{-1}|} = \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \Sigma_l^T = \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} |\Sigma_l|$$

Trước khi bước vào việc biến đổi (4.2) ta cần đi chứng minh một bổ đề: Với x là vector n chiều và A là ma trận $(n \times n)$ thì $\nabla_A(x^T.Ax) = x.x^T$
Thật vậy:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}; \quad x^T = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}; \quad A = \begin{bmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nn} \end{bmatrix}$$

$$x^T Ax = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} x_1A_{11} + \dots + x_nA_{n1} & x_1A_{12} + \dots + x_nA_{n2} & \dots & x_1A_{1n} + \dots + x_nA_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= (x_1A_{11}x_1 + \dots + x_nA_{n1}x_1) + (x_1A_{12}x_2 + \dots + x_nA_{n2}x_2) + \dots$$
$$+ (x_1A_{1n}x_n + \dots + x_nA_{nn}x_n)$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{n} x_iA_{ij}x_j = F$$

Suy ra:

$$\nabla_A(x^TAx) = \nabla_A F = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \dots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{n1}} & \dots & \frac{\partial f}{\partial A_{nn}} \end{bmatrix} = \begin{bmatrix} x_1x_1 & \dots & x_1x_n \\ \vdots & \ddots & \vdots \\ x_nx_1 & \dots & x_nx_n \end{bmatrix} = x.x^T$$

Từ bổ đề trên ta biến đổi (4.2):

$$(4.2) = -\frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \nabla_{\Sigma_l^{-1}} \left((x^{(i)} - \mu_l)^T \Sigma_l^{-1} (x^{(i)} - \mu_l)\right) = -\frac{1}{2}\sum_{i=1}^{m} w_j^{(i)} \left((x^{(i)} - \mu_l)(x^{(i)} - \mu_l)^T\right)$$

Kết hợp (4.1) và (4.2) sau biến đổi:

$$\nabla_{\Sigma_l^{-1}} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\right).\phi_j}{w_j^{(i)}} = (4.1) + (4.2)$$

$$= \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)}|\Sigma_l| - \frac{1}{2}\sum_{i=1}^{m} w_j^{(i)}\left((x^{(i)}-\mu_l)(x^{(i)}-\mu_l)^T\right)$$

Đặt biểu thức trên bằng 0 và giải phương trình theo Σ suy ra:

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)}\left((x^{(i)}-\mu_j)(x^{(i)}-\mu_l)^T\right)}{\sum_{i=1}^{m} w_j^{(i)}}$$

Từ việc cực đại bằng phương pháp lấy đạo hàm riêng ta được kết quả cuối cùng:

$$\phi_j = \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)},$$
$$\mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$
$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)}(x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

Từ kết quả này ta đưa ra các bước lặp cho giải thuật Mixtures of Gaussian và EM.

Lặp cho đến khi hội tụ: {

(E-step) Với mỗi i, j, tính:

$$w_j^{(i)} := p(z^{(i)} = j \vee x^{(i)}; \phi,\ \mu,\ \Sigma)$$

(M-step) Cập nhật thông số:

$$\phi_j = \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)},$$
$$\mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$
$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)}(x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

}

4.3 Giải thuật Self Organizing Maps - SOM

4.3.1 Giới thiệu

SOM là một giải thuật dựa trên mô hình học không giám sát (unsupervised learning), trong đó các hình thành một bản đồ để tạo thành phân loại riêng

về dữ liệu huấn luyện mà không cần sự giúp đỡ từ bên ngoài. Để làm điều này chúng ta phải giả định rằng các thành viên lớp được định nghĩa rộng của các mô hình đầu vào chia sẻ các tính năng phổ biến (common features), và rằng mạng sẽ có thể xác định những tính năng trên phạm vi của mô hình đầu vào.

Một phần đặc biệt thú vị của hệ thống unsupervised được dựa trên competitive learning, trong đó các neuron ra cạnh tranh với nhau để được kích hoạt, với kết quả là chỉ có một được kích hoạt tại bất kỳ thời điểm nào. Neuron kích hoạt này được gọi là winner-takes-all neuron hoặc chỉ đơn giản là neuron chiến thắng (winning neuron) hay Best Matching Unit (BMU). Cạnh tranh như vậy có thể được thực hiện bằng cách kết nối biên (lateral inhibition connections - đường dẫn thông tin phản hồi tiêu cực) giữa các neuron. Không chỉ vậy mà winning neuron còn tác động đến các neuron xung quanh nó. Kết quả là các neuron bị buộc phải tự tổ chức. Vì thế nên được gọi là tự tổ chức Self Organizing Map (SOM).

Làm thế nào chúng ta có thể thực hiện đưa các thuộc tính vào trong unsupervised. Điều đầu tiên để nhận ra là ta cần một số tương tác giữa các neuron trong mạng, do đó, khi một neuron hoạt động, nó ảnh hưởng đến những neuron xung quanh. Làm thế nào để sự tương tác này làm việc? Neuron gần nhau trong mô hình đại diện cho các thuộc tính tương tự. Điều này có nghĩa rằng các winning neuron nên kéo neuron khác gần với nó trong mạng gần hơn với bản thân trong không gian trọng số. Tương tự như vậy, neuron ở xa đại diện thuộc tính khác nhau, do đó winning neuron đẩy các neuron đó đi bằng cách sử dụng các kết nối tiêu cực Neuron mà rất xa trong mạng đại diện cho các thuộc tính khác với winning neuron, vì vậy chỉ cần bỏ qua chúng.

4.3.2 Thành phần của SOM

Mục tiêu chính của một SOM là biến đổi một mô hình không gian nhiều chiều thành mô hình có một hoặc hai chiều rời rạc.

Do đó thiết lập SOM bằng cách đặt các neuron tại các nút của lưới một hoặc hai chiều. Bản đồ có nhiều chiều hơn có thể thực hiện được nhưng không được sử dụng nhiều.

Các neuron chọn lọc để điều chỉnh mô hình đầu vào khác nhau (kích thích) hoặc các lớp của mô hình đầu vào trong quá trình học tập cạnh tranh.

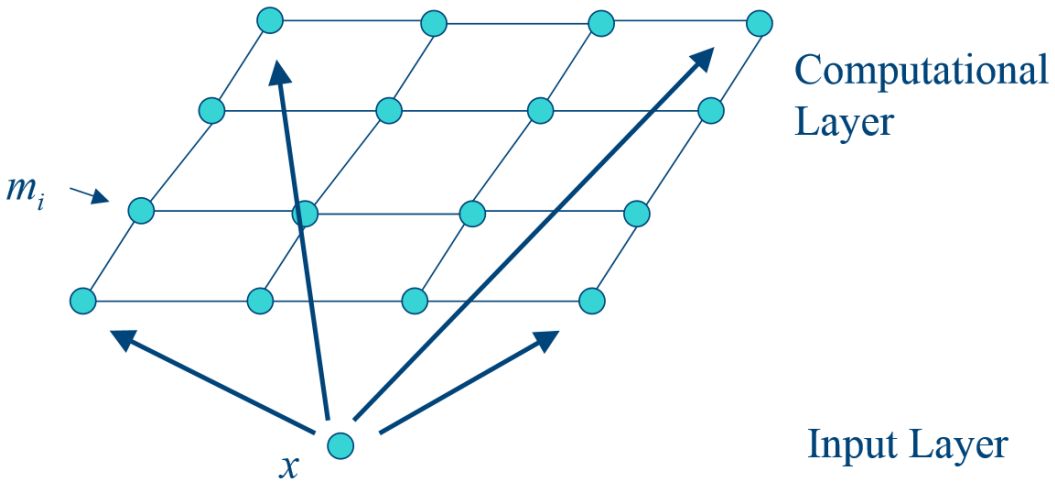
Vị trí của các neuron để điều chỉnh trở nên có trật tự và một hệ thống phối hợp có ý nghĩa cho các đầu vào các tính năng được tạo ra trên mạng.

SOM do đó tạo thành bản đồ cần thiết của mô hình đầu vào.

Chúng ta có thể xem đây là một sự tổng quát phi tuyến phân tích thành phần chính (PCA).

4.3.2.1 Kiến trúc SOM

Một kiến trúc mạng neuron dựa trên học tập cạnh tranh phát minh bởi Teuvo Kohonen năm 1981. Không phụ thuộc vào sự lựa chọn ưu tiên của số cụm để tìm kiếm - sẽ tìm thấy số lượng các cụm thích hợp cho các bộ instances. Đôi khi được coi là không gian hai chiều của cụm trong không gian nhiều chiều. SOM có một cấu trúc feed-forward với một lớp tính toán đơn sắp xếp thành hàng và cột. Mỗi neuron được kết nối đầy đủ với tất cả các nút nguồn trong lớp nhập:



Hình 4.2: Kiến trúc SOM

4.3.2.2 Các thành phần

1. Tập huấn luyện: Đầu vào bao gồm một tập dữ liệu huấn luyện nhiều phần tử. Mỗi phần tử là một Vector có n chiều, và bao gồm p phần tử:

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \dots & x_{pn} \end{bmatrix}$$

2. Bản đồ các neuron: Bản đồ một hoặc hai chiều với thành phần là các neuron với trọng số là một Vector có số chiều bằng với phần tử đầu

vào

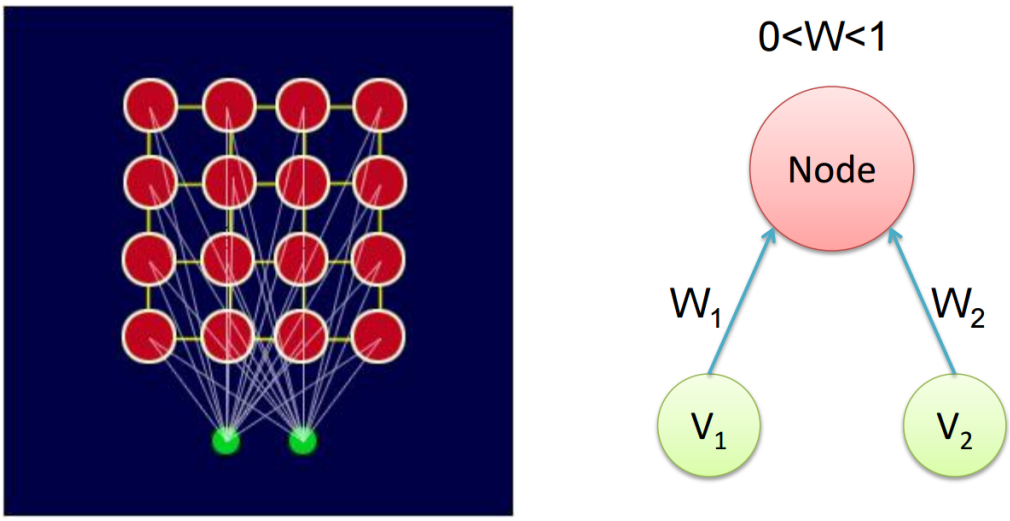
$$\begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q1} & w_{q2} & \dots & w_{qn} \end{bmatrix}$$

Các neuron sẽ hình thành các cụm của bản đồ với mỗi cụm đại diện cho sự tương đồng thuộc tính của các phần tử trong tập huấn luyện. Mỗi cụm bao gồm một hoặc nhiều neuron đại diện. Số neuron được tạo ra ban đầu có thể nhỏ hơn, bằng hoặc lớn hơn số chiều của phần tử đầu vào nhưng phải nhỏ hơn tổng số phần tử của tập huấn luyện.

4.3.2.3 Các quá trình hình thành thuật toán

Quá trình tự tổ chức liên quan đến bốn quá trình chính:

1. Khởi tạo
   - Chọn số lượng neuron thích hợp và số chiều của bản đồ
   - Tất cả các trọng số kết nối được khởi tạo với giá trị ngẫu nhiên nhỏ.



Hình 4.3: Khởi tạo SOM

2. Cạnh tranh
   Đối với mỗi mẫu đầu vào, các neuron tính toán giá trị tương ứng của một hàm biệt thức (discriminant function) cung cấp cơ sở cho sự cạnh tranh. Các neuron đặc biệt với giá trị nhỏ nhất của hàm biệt thức được tuyên bố là neuron chiến thắng (winning neuron).

3. Hợp tác
   Các neuron giành chiến thắng xác định các vị trí không gian topo của neuron bị kích thích, do đó cung cấp cơ sở cho việc hợp tác giữa các neuron lân cận.

4. Thích ứng
   Các neuron bị kích thích giảm giá trị trọng số về hàm biệt thức liên quan đến mô hình đầu vào thông qua điều chỉnh phù hợp của trọng số kết nối liên quan, như vậy là phản ứng của winning neuron trên các ứng dụng tiếp theo của một mô hình đầu vào tương tự được tăng cường.

4.3.2.4 Thuật toán SOM

Thuật toán SOM có thể được chia thành 7 bước:
Bước 1 : Số lượng neuron và trọng số của mỗi neuron được khởi tạo. Tương đương với quá trình khởi tạo.
Bước 2 : Một dữ liệu được chọn ngẫu nhiên từ tập dữ liệu huấn luyện và đưa vào mạng.
Bước 3 : Mỗi neuron trong mạng được kiểm tra để tính toán mà trọng số của nó gần giống như vector đầu vào. Các neuron chiến thắng thường được gọi là các đơn vị tốt nhất - Best Matching Unit (BMU). Đây là quá trình cạnh tranh.
Bước 4: Bán kính ảnh hưởng của BMU được tính toán. Giá trị này ban đầu là khá lớn. Thông thường nó được thiết lập để được bán kính của mạng, giảm dần từng bước lặp của thuật toán.
Bước 5 : Bất kỳ các neuron lân cận được tìm thấy trong vòng bán kính ảnh hưởng của BMU đã tính trong bước 4 được điều chỉnh để làm cho chúng giống như các vector đầu vào. Càng nhiều neuron gần BMU, càng có nhiều trọng số được thay đổi (quá trình cộng tác).
Bước 6: Lặp lại bước 2 với N vòng lặp.
Bước 7: Sau khi kết thúc N vòng lặp, bản đồ đã được định hình thành với số cụm nhất định. Mỗi cụm đại diện bởi một hoặc nhiều neuron. Sau đó ta sẽ đưa các phần tử của tập huấn luyện vào. Nếu neuron nào gần với phần tử đầu vào nhất thì phần tử đầu vào sẽ thuộc cụm của neuron đó. Sau khi đưa tất cả các phần tử của tập huấn luyện vào, các phần tử sẽ thuộc vào các cụm. Như vậy ta đã tiến hành phân cụm thành công các phần tử của tập huấn luyện.

4.3.2.5 Các quá trình thực hiện

1. Quá trình cạnh tranh - The Competitive Process

   - Nếu không gian đầu vào là D chiều (tức là D đơn vị đầu vào), có thể viết các mẫu đầu vào như $x = \{x_i : i = 1, \ldots, D\}$ và trọng

số wj của neuron trong lớp tính toán có thể được viết $w_j = \{w_i : j = 1, \ldots, N;\ i = 1, \ldots, D\}$ trong đó N là tổng số neuron.

- Sau đó chúng tôi có thể xác định chức năng biệt thức của chúng tôi là khoảng cách Euclide bình phương giữa các vector đầu vào x và wj vector trọng số cho mỗi neuron j

$$d_j(x) = \sum_{i=1}^{D}(x_i - w_{ji})^2$$

- Phương trình được tính bằng công thức khoảng cách Euclide bình phương vì ta không quan tâm đến khoảng cách thực tế từ đầu vào. Chúng ta chỉ cần một số loại quy mô thống nhất để so sánh mỗi nút với vector đầu vào. Nói cách khác, các neuron có trọng số vector gần nhất với vector đầu vào được xác định là winning neuron.
- Bằng cách này, không gian đầu vào liên tục có thể được ánh xạ vào không gian đầu ra rời rạc của neuron của một quá trình đơn giản của cuộc cạnh tranh giữa các neuron.

2. Quá trình hợp tác - The Cooperative Process
Trong các nghiên cứu sinh học thần kinh chúng ta thấy rằng có sự tương tác bên trong một tập hợp các neuron bị kích thích. Khi một neuron hoạt động, các neuron lân cận của nó có xu hướng dễ kích thích hơn so với những neuron ở xa. Có một topo lân cận phân rã theo khoảng cách. Chúng ta áp dụng lý thuyết đó vào SOM với công thức

$$T_{j,I(x)} = e^{-\frac{S^2_{j,I(x)}}{2\sigma^2}}$$

- $T_{j,I(x)}$ sử dụng để xác định các neuron gần với winning neuron nhiều hơn các neuron khác nằm ngoài bán kính ảnh hưởng. Các neuron bên ngoài bán kính ảnh hưởng được bỏ qua hoàn toàn.
- $I(x)$ là vị trí của winning neuron. Điều này có một số đặc tính quan trọng: nó là tối đa tại các neuron chiến thắng, nó là đối xứng về neuron đó, nó làm giảm về giá trị 0 khi khoảng cách đi đến vô cùng, và nó không di chuyển (nghĩa là độc lập với vị trí của các neuron chiến thắng).
- Trong đó σ là bán kính ảnh hưởng của neuron

$$\sigma(t) = \sigma_0 e^{\frac{-t}{\lambda}}$$

t: vòng lặp hiện tại
λ: hằng số = số vòng lặp / kích thước bản đồ σ_0. Giá trị gần như là tùy ý. Bất kỳ giá trị hằng số nào cũng có thể được lựa

chọn. Tuy nhiên nó phụ thuộc trực tiếp vào kích thước bản đồ và số lần lặp để thực hiện.

σ_0: bán kính ảnh hưởng của neuron tại thờ điểm t_0

Một thuộc tính đặc biệt của SOM là σ kích thước của bán kính ảnh hưởng cần phải giảm theo thời gian. Một hàm thời gian phụ thuộc phổ biến là một phân rã theo hàm số mũ.

Tại t = 0 giá trị là lớn nhất. Khi t (số lần lặp hiện hành) tăng lên, giá trị tiệm cận bằng không. Đây chính là điều chúng ta muốn. Bán kính bắt đầu như bán kính của mạng tinh thể, khi tiệm cận bằng không, lúc đó bán kính chỉ đơn giản là nút BMU
$S_{j,I(x)}$ là khoảng cách Euclid từ neuron lân cận đến winning neuron

3. Quá trình thích ứng - The Adaptive Process
Không phải chỉ có winning neuron được cập nhật trọng số mà các neuron lân cận cũng được cập nhật. Trong thực tế, phương trình cập nhật trọng số thích hợp là

$$\Delta w_{ji} = \eta(t).T_{j,I(x)}.(x_i - w_{ji})$$

$$w_{ji} = w_{ji} + \Delta w_{ji}$$

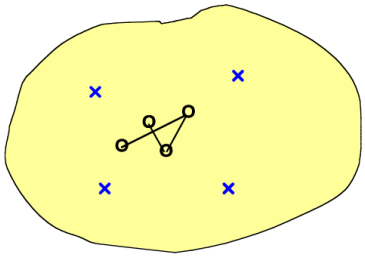
$\eta(t)$: tốc độ học tập - learning rate. Với $\eta(t+1) = \alpha\eta(t)^{k/k_{max}}$ trong đó $0 \leq \alpha \leq 1$ quyết định lượng giảm tốc độ học, k là số lần lặp của thuật toán đã được chạy, và k_{max} là thông số khi muốn việc học dừng lại.

Mỗi lần cập nhật trọng số học tập là để di chuyển các vector trọng số w_i của winning neuron và các neuron lân cận về phía vector đầu vào x. Quá trình lặp đi lặp lại của dữ liệu huấn luyện do đó dẫn đến trật tự topo.

Có hai giai đoạn mang tính chất của quá trình thích nghi này:

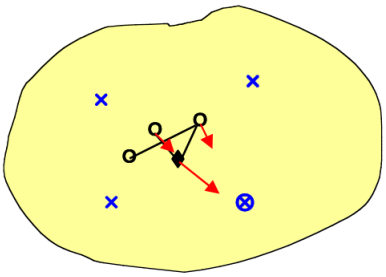
- Thứ tự hoặc giai đoạn tự tổ chức (Ordering or self-organizing phase) - trong đó thứ tự topo của vector trọng số diễn ra. Thông thường điều này sẽ mất là 1000 lần lặp lại các thuật toán SOM, và xem xét cẩn thận cần phải được đưa ra để lựa chọn bán kính ảnh hưởng và tham số tốc độ học.
- Hội tụ - trong đó các bản đồ feature được tinh chỉnh và đến khi cung cấp một lượng thống kê chính xác về không gian đầu vào. Thông thường các số lần lặp lại trong giai đoạn này sẽ có ít nhất hơn 500 lần số lượng neuron trong mạng, và một lần nữa các thông số phải được lựa chọn cẩn thận.

4.3.3 Ví dụ của SOM



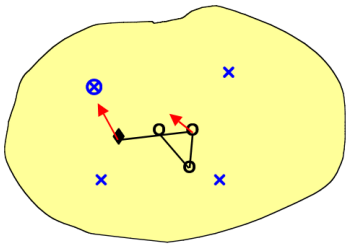
Hình 4.4: Bước 1: Khởi tạo giá trị bảng đầu

Giả sử ta có bốn điểm dữ liệu trong không gian đầu vào hai chiều liên tục, và muốn đưa vào bốn điểm trong một không gian đầu ra một chiều rời rạc. Các nút đầu ra (neuron) được đưa các điểm trong không gian đầu vào (vòng tròn). Trọng số ban đầu được chọn ngẫu nhiên là vòng tròn tại các vị trí ngẫu nhiên trong trung tâm của không gian đầu vào.



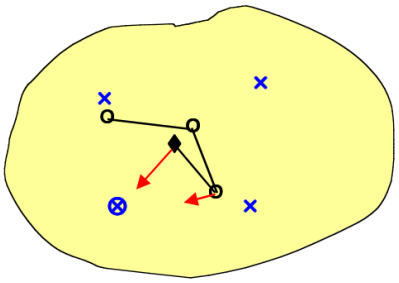
Hình 4.5: Bước 2: Đưa giá trị đầu vào học

Ta chọn ngẫu nhiên một trong những điểm dữ liệu cho vào học (kí hiệu là vòng tròn bao quanh dấu x). Các điểm đầu ra gần nhất đại diện cho các neuron chiến thắng (tứ giác được tô đen). Đó là winning neuron đang di chuyển về phía điểm dữ liệu bằng lượng nhất định, và hai neuron lân cận di chuyển bởi một lượng nhỏ hơn (mũi tên).



Hình 4.6: Bước 3: Học giá trị tiếp theo

Tiếp tục chọn ngẫu nhiên một điểm dữ liệu cho vào học (qua trong vòng tròn). Winning neuron đang tiến về điểm đầu vào bằng lượng nhất định, và neuron lân cận di chuyển bởi một lượng nhỏ hơn (mũi tên).

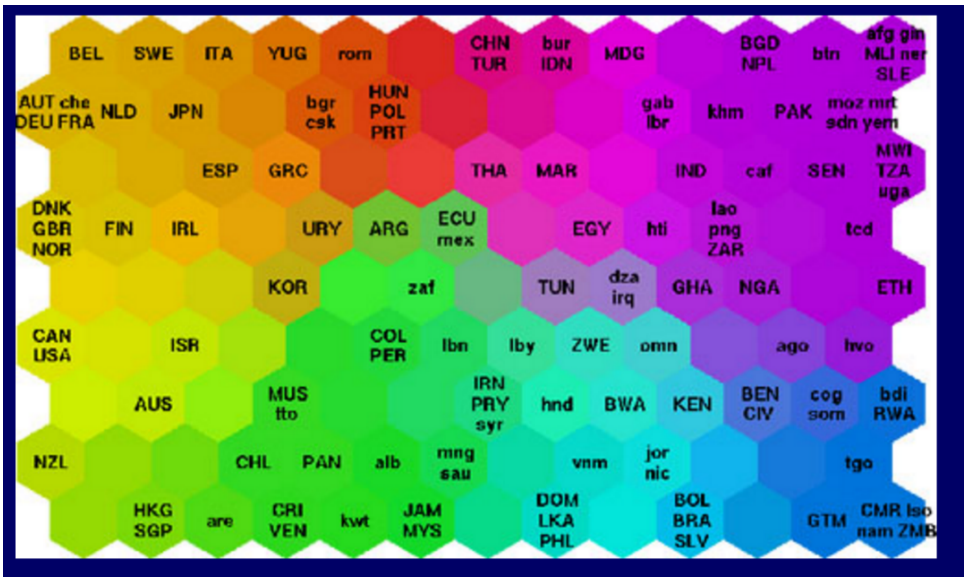


Hình 4.7: Bước 4: Tiếp tục cho các giá trị vào học

Tiếp tục chọn ngẫu nhiên một điểm dữ liệu cho học (qua trong vòng tròn). Winning neuron đang tiến về điểm dữ liệu bằng một lượng nhất định, và một trong những neuron láng giềng di chuyển bởi một số lượng nhỏ hơn (mũi tên). Cuối cùng toàn bộ lưới được xác định để đại diện cho không gian đầu vào.

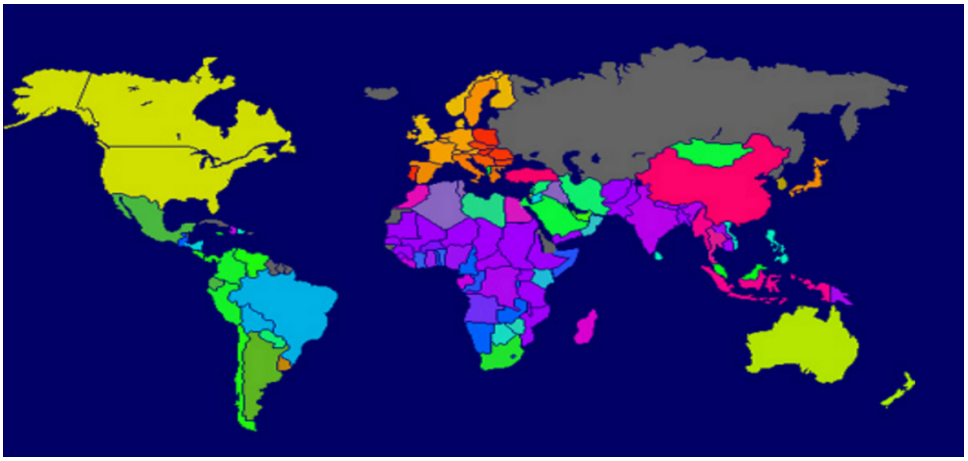
4.3.4 Ứng dụng của SOM

SOM thường được sử dụng làm công cụ trực quan. Nó giúp chúng ta hình dung dễ dàng về mối quan hệ giữa khối lượng lớn dữ liệu.
SOM đã được sử dụng để phân loại dữ liệu thống kê mô tả khác nhau chất



Hình 4.8: Biểu đồ thể hiện chất lượng cuộc sống các quốc gia

lượng của cuộc sống theo các yếu tố như tình trạng sức khỏe, dinh dưỡng, các dịch vụ giáo dục, ... Các quốc gia có chất lượng cuộc sống với các yếu tố tương tự cùng nhóm nhau. Các quốc gia có chất lượng cuộc sống tốt hơn nằm phía trên bên trái và các nước nghèo khó nhất nằm ở phía dưới bên phải. Lưới lục giác là một ma trận khoảng cách thống nhất, thường được biết đến như một U-ma trận. Mỗi hình lục giác đại diện cho một nút trong SOM
Ngoài ra còn có một số ứng dụng khác như phân loại hình ảnh hay màu sắc.



Hình 4.9: Bản đồ chất lượng cuộc sống thế giới

Chương 5

Hệ thống phát hiện xâm nhập OSSEC

5.1 Tổng quan về OSSEC

OSSEC HIDS (Open Source HIDS) là 1 HIDS mã nguồn mở, đa nền tảng và có tính mở rộng cao. OSSEC cung cấp các chức năng để giám sát hoạt động trên máy tính và từ đó đảm bảo an toàn cũng như bảo mật cho máy tính.

OSSEC có các chức năng chính như sau:

- Công cụ phân tích mạnh mẽ
- Tích hợp phân tích file log
- Kiểm tra tính toàn vẹn của file
- Giám sát thanh ghi trên Windows
- Cung cấp cơ chế quy định tập trung (policy centralize)
- Kiểm tra rootkit
- Phát cảnh báo thời gian thực
- Active response

OSSEC có thể chạy trên nhiều hệ điều hành khác nhau, bao gồm Linux, OpenBSD, FreeBSD, Mac OS X, Sun Solaris, và cả Microsoft Windows.

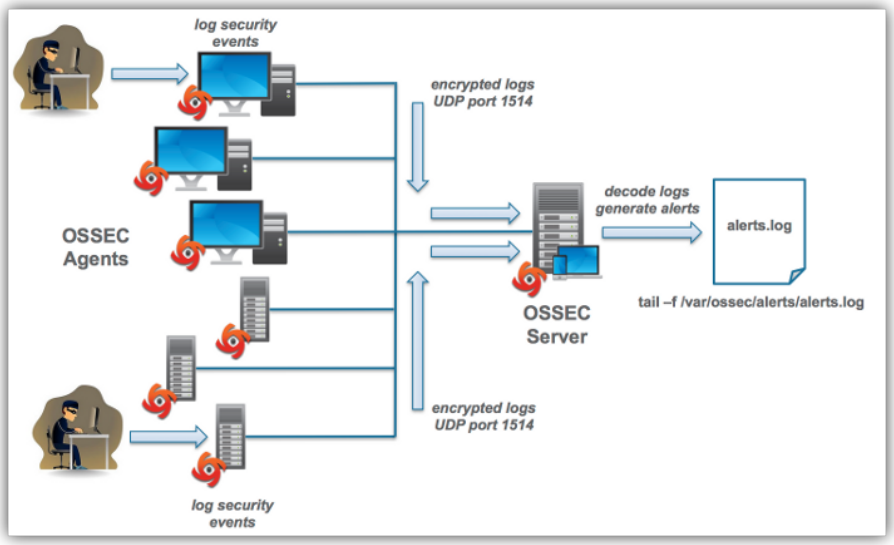
OSSEC được xây dựng dựa trên ngôn ngữ C, các file cấu hình và file rules được lưu trữ dưới dạng xml, cấu trúc tổ chức này giúp việc cấu hình dễ hiểu, dễ chỉnh sửa.

5.2 Mô hình của OSSEC

5.2.1 Mô hình local

- Mô hình local thích hợp khi cài đặt lên hệ thống chỉ có 1 máy tính, ví dụ như máy tính cá nhân, hoặc là server riêng lẻ.
- Mô hình local dễ dàng quản lý và có thể chỉnh sửa theo ý muốn ngay trên thiết bị cài đặt.
- Với mô hình local, máy tính được cài đặt OSSEC sẽ có đầy đủ các tính năng cung cấp bởi OSSEC.

5.2.2 Mô hình server

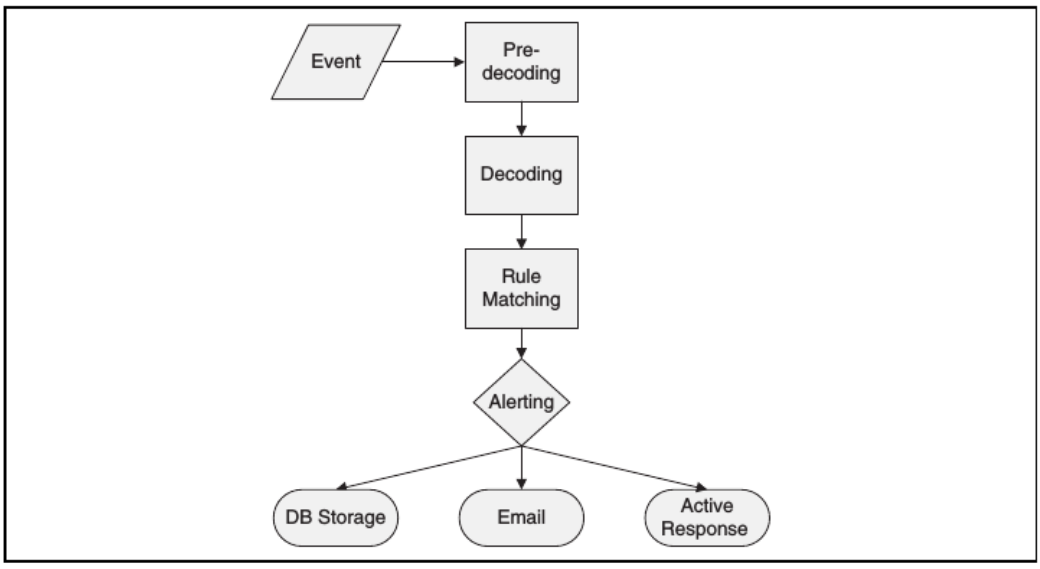


Hình 5.1: Mô hình Server-agent trong OSSEC

- Mô hình server-agent thích hợp khi cài đặt lên hệ thống có nhiều máy tính, ví như như trong 1 công ty, 1 phòng nghiên cứu, nhằm mục đích quản lý tập trung toàn bộ hệ thống. Mô hình này đòi hỏi thiết lập 1 máy tính đóng vai trò server, và các máy tính còn lại là các agent.
- Về phía server: Server sẽ có đầy đủ tính năng của OSSEC, như việc kiểm tra toàn vẹn file, giám sát thanh ghi, phát hiện rootkit,... ngoài ra có thêm các chức năng như thu thập log file, ghi file log cảnh báo, đặt luật cho toàn bộ hệ thống IDS.

- Về phía agent: Agent chỉ có các chức năng sau: kiểm tra toàn vẹn file, giám sát thanh ghi, phát hiện rootkit, active response. Các chức năng như ghi file log cảnh báo, xem file log, thu thập syslog, đặt luật đều được tập trung tại server.
- Mô hình này cung cấp khả năng triển khai hệ thống an ninh để bảo vệ hệ thống trên các máy tính và tập trung dữ liệu xử lý trên server, các agent sẽ loại bỏ được công việc xử lý dữ liệu, và cơ chế tập trung về quy định và xử lý giúp quản lý tốt hơn.

5.3 Cách thức hoạt động trong OSSEC



Hình 5.2: Cách thức hoạt động của OSSEC

5.3.1 Nhận event

Event là những log file của các chương trình khi được thực thi, các chương trình trên máy tính khi thực thi đều tạo ra file log để ghi lại hành động. Chương trình OSSEC sẽ đọc tất cả các log file này mỗi khi chương trình khác thực thi. Từ file log này, OSSEC sẽ thực hiện công việc tiếp theo.

5.3.2 Rút trích dữ liệu

Tiến hành Decode và rút trích những thông tin liên quan từ event, phần decode được thực hiện thông qua 2 giai đoạn:

5.3.2.1 Predecoding

Rút trích các thông tin static từ event từ các trường quen thuộc và đơn giản, như ngày, giờ, log, hostname, tên chương trình.

```
Apr 14 17:32:06 linux_server sshd[1025]: Accepted password for dcid from
192.168.2.180 port 1618 ssh2
```

Field	Description
hostname	linux_server
program_name	sshd
log	Accepted password for dcid from 192.168.2.180 port 1618 ssh2
time/date	Apr 14 17:32:06

Hình 5.3: Predecoder

Predecoding hoạt động không dựa vào nguồn dữ liệu đầu vào của event. Vì tất cả các event đều sẽ có chung các thông tin đơn giản mà predecoding rút trích.

Khi xử lý log: OSSEC sẽ chuẩn hóa các kiểu log khác nhau thành 1 kiểu chuẩn do OSSEC định nghĩa, từ đó có thể viết các rules theo kiểu chuẩn này để rút trích được các thông tin cần thiết. Ngoài ra còn có thể định nghĩa rule để xét từng trường riêng với nhau: hostname hoặc program_name.

Ví dụ: Syslog ở hình trên và ASL ở hình dưới có log file hoàn toàn khác nhau, nhưng cũng có thể rút trích được theo các trường như decoder ở trên.

```
[Time 2006.12.28 15:53:55 UTC] [Facility auth] [Sender sshd] [PID 483]
[Message error: PAM: Authentication failure for username from 192.168.0.2]
[Level 3] [UID -2] [GID -2] [Host mymac]
```

Hình 5.4: Log của ASL

5.3.2.2 Decoding

Dây là bộ phận quan trọng nhất trong quá trình hoạt động của OSSEC, các thông tin quan trọng để xác định sự kiện được lấy ra trong quá trình này.

Decoding rút trích những thông tin mà phần predecoding chưa lấy được, như usernames hoặc địa chỉ IP nguồn, ...

Khác với predecoder, các thông tin do decoder rút trích khác nhau giữa

các event, do vậy mỗi 1 event đều có 1 decoder riêng, toàn bộ các decoder đều được đặc tả trong file decoders.xml.

Các trường có trong 1 decoder được miêu tả trong bảng dưới.

Tag	Miêu tả
program_name	Thực thi decoder khi tên chương trình trong file log trùng với program_name
prematch	Thực thi decoder khi prematch trùng với bất kỳ phần nào trong log
regex	Regular Expression để so sánh với log, rút trích những dữ liệu trùng
offset	Thuộc tính của regex. Để báo so sánh regex bắt đầu từ đâu, có 2 giá trị là after_prematch và after_parent
order	Thứ tự trong regex khi rút trích dữ liệu. Có thể là sự kiện thường gặp (srcip, user, dstip, dstport, ...)
parent	Decoder cha trước khi xét đến decoder này.

Bảng 5.1: Các trường trong decoder

Với những trường này, decoder cung cấp các tính năng như so sánh theo tên chương trình (program_name) hoặc so trùng thông tin (prematch). Ngoài ra decoder còn xếp sắp theo nhiều cấp trong mô hình cây giúp cho việc tạo decoder dễ dàng hơn, có tính kế thừa và mở rộng cao.

Dưới đây là 1 ví dụ về decoder của 1 đoạn log của ACL. Trong đoạn log sử dụng các tag <parent> để kế thừa decoder cha, <type> để xét định loại decoder, xác định tường để so sánh bằng <prematch>, sau đó xác định thông tin cần rút trích bằng <regex> và <order>.

```
<decoder name="sshd-success">
  <parent>sshd</parent>
  <prematch>^Accepted</prematch>
  <regex offset="after_prematch">^ password for (\S+) from (\S+) port </regex>
  <order>user, srcip</order>
</decoder>

<decoder name="ssh-failed">
  <parent>sshd</parent>
  <prematch>^Failed password </prematch>
  <regex offset="after_prematch">^for invalid user \S+ from (\S+) </regex>
  <order>srcip</order>
</decoder>
```

Hình 5.5: Decoder

```
%SEC-6-IPACCESSLOGP: list 102 denied tcp 10.0.6.56(3067) -> 172.36.4.7(139),
 1 packet
%SEC-6-IPACCESSLOGP: list 199 denied tcp 10.0.61.108(1477) -> 10.0.127.20(44
5), 1 packet
```

Hình 5.6: Đoạn log của ACL

```
<decoder name="cisco-ios-acl">
  <parent>cisco-ios</parent>
  <type>firewall</type>
  <prematch>^%SEC-6-IPACCESSLOGP: </prematch>
  <regex offset="after_prematch">^list \S+ (\w+) (\w+) </regex>
  <regex>(\S+)\((\d+)\) -> (\S+)\((\d+)\),</regex>
  <order>action, protocol, srcip, srcport, dstip, dstport</order>
</decoder>
```

Hình 5.7: Decoder cho đoạn log ACL

5.3.3 Rule matching

Sau khi rút trích được dữ liệu từ event, 1 cơ chế so sánh rules được gọi để xác định tính chất của event và quyết định điều khiển báo động.

5.4 Về rules trong OSSEC

Rules được định nghĩa trong các file xml riêng cho từng vấn đề (apache, syslog, ...), 1 file chứa nhiều rule.

Mỗi rules được gán 1 ID cố định và 1 mức độ (level) của rules đó.

5.4.1 Về ID

- Rules trong OSSEC có ID từ 00000 đến 50799, các rule được chia nhóm theo từng chức năng riêng, có các nhóm quan trọng như:
  - 01000 - 01999: Rule cho syslog
  - 05100 - 05299: Rule cho nhân hệ thống Linux, UNIX, BSD
  - 30100 - 30999: Rule cho Apache HTTP server
  - 31100 - 31199: Rule cho truy cập web
  - 50100 - 50299: Rule cho MySQL database

- Ngoài ra, người dùng có thể tự định nghĩa rule có ID từ 100000 đến 119999
- Khi sử dụng mô hình server-agent, các agent có thể có rules riêng cho bản thân, các rules này sẽ được viết vào trong file local_rules.xml.

5.4.2 Về mức

- OSSEC gồm có 16 mức: 0 - 15, số mức cáng cao càng có độ cảnh báo cao.
- Mức 0 là mức đặc biệt, mang ý nghĩa là bỏ qua. Khi 1 rules được đánh giá là mức 0, event đó sẽ được bỏ qua và không xét tới nữa.
- Các luật trong OSSEC được lưu trữ theo cấu trúc cây, luật với mức cao được đánh giá trước, nhưng trước tiên phải xét luật với mức 0 vì đây là trường hợp đặc biệt, sẽ bỏ qua tất cả.

5.4.3 Atomic rules

- Là các luật chỉ xét riêng cho từng event riêng lẻ.
- Các tag chính trong rule:
  - Tất cả các rule trong OSSEC đều được đặt trong tag <group>, <group> là tag để quản lý các loại rule có chung đặc điểm, mỗi group có tên để nêu lên đặc điểm của rule trong đó, như syslog, sshd, ...
  - Trong OSSEC cung cấp cơ chế kế thừa giữa các rules, 1 event sau khi được rút trích, thông tin đó sẽ được so sánh với rule cha, nếu trùng sẽ so sánh tiếp xuống rule con, như vậy sẽ tiết kiếm được chi phí so sánh khi không cần thiết phải so sánh hết với toàn bộ tập rule, và sẽ cung cấp khả năng để các rules mở rộng theo nhiều hướng từ 1 rules chính. Sử dụng thừa kế bằng cách thừa kế group chứa rule với tag <group>.
  - Có các group lớn mà OSSEC đã định nghĩa sẵn, có thể sử dụng những group này làm group cha khi người dùng tự định nghĩa các rule.
  - Sử dụng tag <decoded_as> để xác định decoder để decode event, rule được xác định decoder chỉ so sánh khi được decoder đó decode.
  - Sử dụng tag <match> để hỗ trợ cho tag <decoder_as>, tag <match> để so sánh trùng với nội dung mà decoder rút trích được.

```
<group name="syslog,sshd,">
  <rule id="100120" level="5">
    <group>authentication_success</group>
    <description>SSHD testing authentication success</description>
  </rule>
  <rule id="100121" level="6">
    <description>SSHD rule testing 2</description>
  </rule>
</group>
```

Hình 5.8: Subgroup trong group

```
<rule id="100124" level="5">
  <decoded_as>sshd</decoded_as>
  <match>^Failed password</match>
  <description>Failed SSHD password attempt</description>
</rule>
```

Hình 5.9: Tag match trong rule

```
<rule id="100126" level="12">
  <if_sid>100124</if_sid>
  <group>authentication_failure</group>
  <hostname>main_sys</hostname>
  <srcip>!192.168.2.0/24</srcip>
  <description>Severe SSHD password failure.</description>
</rule>
```

Hình 5.10: Tag scrip trong rule

- Ngoài ra còn có các tag như <hostname>, <srcip> để kiểm tra cụ thể theo yêu cầu như hostname hoặc srouce ip.
- Tag <time> để kiểm tra về thời gian của event, để xác định chính xác hơn về quy định. Ví dụ như việc login vào server chỉ nên xảy ra trong thời gian làm việc, việc mọi sự đăng nhập ngoài giờ làm việc sẽ được cảnh báo ở mức cao hơn.
- Lưu ý là 1 sự kiện chỉ tạo ra 1 alert, rule được so sánh theo sơ đồ cây và node cuối cùng được so sánh trùng sẽ phát ra alert với mức của rule đó.

```
<rule id="100127" level="10">
  <if_sid>100125</if_sid>
  <time>6 pm - 8:30 am</time>
  <description>Login outside business hours.</description>
  <group>policy_violation</group>
</rule>
```

Hình 5.11: Tag time trong rule

5.4.4 Composite rules

- Composite rules không xét có event riêng lẻ với nhau, mà so sánh event hiện tại với các event đã bắt được trước đó. Để các rule có để giữ được trạng thái, 1 rule phải có thêm hai thuộc tính sau:
  - Frequency: để lấy tần số xuất hiện của event (event/pattern), tần số đủ lớn mới cảnh báo.
  - Timeframe: xem xét lại thời gian từ event bây giờ tới thời điểm event trước đó là bao nhiêu giây.

```
<rule id="100130" level="10" frequency="5" timeframe="600">
  <if_matched_sid>100124</if_matched_sid>
  <description>5 Failed passwords within 10 minutes</description>
</rule>
```

Hình 5.12: Tần số trong composite rule

- Event được đếm tần số xuất hiện có thể dựa trên id <if_match_sid>, dựa trên group <if_match_group>, dựa trên regex <if_match_regex>.
- Ngoài ra có thể tăng độ chính xác và giảm lỗi bằng các tag so sánh cùng ip, port, hostname, program name.

Tag	Miêu tả
same_source_ip	Địa chỉ IP của Source phải giống nhau
same_dest_ip	Địa chỉ IP của Dest phải giống nhau
same_src_port	Địa chỉ Port của Source phải giống nhau
same_dst_port	Địa chỉ Port của Dest phải giống nhau
same_location	Phải cùng 1 host hoặc agent
same_user	Username được rút trích phải giống nhau
same_id	ID được rút trích phải giống nhau

Bảng 5.2: Các tag trong composite rule

5.5 Cách thức cấu hình để OSSEC hoạt động

Tất cả các cấu hình cơ bản cho OSSEC đều nằm trong file ossec.conf, file này được viết bằng xml. File cấu trúc xml được chọn vì nhiều lý do: dễ đọc và dễ tìm kiếm tới những phần cụ thể, cấu trúc kế thừa trong xml giúp cho việc xác định nơi bắt đầu và kết thúc của từng phần cấu hình dễ dàng hơn, file xml có thể được ghi 1 cách tự động từ chương trình cũng có thể dễ dàng thay đổi từ bất cứ 1 công cụ đọc text nào.

Trong file ossec.conf có các tag lớn sau:

Thành phần	Miêu tả
<global>	Những cấu hình chung cho hệ thống trong mô hình local hoặc là riêng server
<alerts>	Tùy chọn cho cảnh báo email và log
<email_alerts>	Tùy chọn chi tiết cho cảnh báo email
<remote>	Cấu hình riêng cho các agent từ xa (chỉ áp dụng cho server)
<database_output>	Tùy chọn cho việc ghi log vào trong database
<rules>	Liệt kê tất cả những file rule được sử dụng
<client>	Cấu hình liên quan đến agent
<localfile>	Cấu hình cho việc giám sát log file
<syscheck>	Cấu hình cho việc kiểm tra tính toàn vẹn
<rootcheck>	Cấu hình cho việc kiểm tra rootkit và điều lệ
<command>	Cấu hình cho active response
<active-response>	Cấu hình nâng cao cho active response

Bảng 5.3: Các thành phần trong file ossec.conf

Cách thức cấu hình cụ thể sẽ được trình bày theo từng module riêng trong mục các module trong OSSEC.

5.6 Các module trong OSSEC

5.6.1 Log alert và email alert

- Đây là module cơ bản và quan trọng nhất trong OSSEC, cần điều chỉnh module này phù hợp với hệ thống cài đặt để tối ưu theo ý muốn người sử dụng.
- Theo mặc định, khi event tạo ra alert với mức từ 1 đến 15 đều được OSSEC ghi vào trong log file. Các sự kiện trên mức 7 sẽ được OSSEC gửi email để alert cho admin.
- Để thay đổi mức cảnh báo này, có thể sử dụng 2 trường trong tag <alert>, đó là <log_alert_level> và <email_alert_level>.

```
<ossec_config>
  <alerts>
    <log_alert_level>2</log_alert_level>
    <email_alert_level>8</email_alert_level>
  </alerts>
</ossec_config>
```

Hình 5.13: Cấu hình mức cảnh báo email của OSSEC

- Email alert là 1 tính năng rất hữu dụng trong OSSEC, vì nó cung cấp khả năng phản ứng nhanh chóng với các cảnh báo.
- Để cấu hình tính năng email alert, sử dụng các tag trong tag <global>.

```
<ossec_config>
  <global>
    <email_notification>yes</email_notification>
    <email_to>john@fakeinc.com</email_to>
    <email_to>mike@fakeinc.com</email_to>
    <smtp_server>smtpserver.fakeinc.com.</smtp_server>
    <email_from>ossecm@fakeinc.com</email_from>
    <email_maxperhour>20</email_maxperhour>
  </global>
</ossec_config>
```

Hình 5.14: Cấu hình email cảnh báo trong OSSEC

- Ngoài ra còn có các tính năng hỗ trợ thêm trong tag <email_alerts>:
  - Tag <group>: xác định nhóm alert của chương trình nào sẽ gửi email đến địa chỉ trên.
  - Tag <level>: xác định mức của cảnh báo để gửi email.
  - Tag <format>: xác định cách thức gửi cảnh báo. Sms để gửi cảnh báo bằng tin nhắn.
  - Tag <event_location>: xác định chỉ những cảnh báo đến từ các agent nào mới gửi cảnh báo.

5.6.2 Xác định file rules

Trong OSSEC cho phép include các file rules cần thiết để xác định rule nào được kiểm tra, các file rule đều nằm trong đường dẫn ossec/rules. File rules được xác định phải nằm trong tag <include>.

```
<ossec_config> <!- rules global entry ->
  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
  </rules>
<ossec_config>
```

Hình 5.15: Liệt kê file rule trong ossec.conf

5.6.3 Kiểm tra toàn vẹn hệ thống

- Phần cơ bản và quan trọng nhất trong 1 HIDS, cơ bản là kiểm tra checksum của file so với checksum bản file tốt để xem file có bị thay đổi hay không.
- Kiểm tra toàn vẹn file có thể giúp chống lại các loại trojans, phần mềm độc hại có khả năng thay đổi mã chương trình.
- Trong OSSEC việc kiểm tra tính toàn vẹn thông qua 3 tag chính trong tag <syscheck>:
  - Tag <directory>: cung cấp cơ chế để xác định folder được kiểm tra toàn vẹn.
  - Tag <ignore>: xác định file, folder có thể bỏ qua trong lúc kiểm tra.

```
<ossec_config>
  <syscheck>
    <frequency>86400</frequency>
    <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin</directories>
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/mnttab</ignore>
  </syscheck>
</ossec_config>
```

Hình 5.16: Cấu hình kiểm tra tính toàn vẹn trong ossec.conf

  - Tag <frequency>: xác định tần số kiểm tra.

- Tạo rules mới nên sử dụng <if_group>syscheck</if_group> để bao hàm group syscheck của OSSEC.

5.6.4 Kiểm tra và phát hiện rootkit

- OSSEC chỉ có thể kiểm tra rootkit trên hệ điều hành Linux, Unix, và BSD, bên cạnh đó cũng cung cấp khả năng kiểm tra về điều lệ (policy) trên hệ thống Windows, Linux, Unix và BSD.
- OSSEC kiểm tra rootkit mức ứng dụng thông qua dấu hiệu và rootkit mức hệ thống thông qua so sánh system call. Ngoài ra còn cung cấp khả năng kiểm tra theo kiểu bất thường (anomaly) để có thể đạt yêu cầu người sử dụng.
- Trong mô hình server-agent, việc cấu hình kiểm tra điều lệ (policy) và kiểm tra rootkit được hiện thực trên server và đưa xuống agent.

5.7 Giao diện web OSSEC - OSSEC WUI

5.7.1 Phần giới thiệu

OSSEC HIDS Web User Interface (WUI) cung cấp một giao diện dựa trên web mà thực hiện các phân tích thống kê và mối tương quan của dữ liệu thay vì sử dụng các thao tác dòng lệnh (command line) để thực hiện.
Các trang điều khiển chính mang lại cho chúng ta một cái nhìn tổng quát vào những gì đang xảy ra khi triển khai OSSEC HIDS. Một công cụ tìm kiếm mạnh mẽ có sẵn để theo dõi các sự kiện mà OSSEC HIDS thu thập được và cảnh báo. Tất cả những tập tin bị thay đổi tính toàn vẹn được ghi lại, và

một giao diện thân thiện cho phép xem các thay đổi tập tin theo thời gian. Cuối cùng, một thống kê hữu ích cho phép tổng hợp tất cả các sự kiện và cảnh báo được thu thập để cung một cái nhìn thống kê những gì đang xảy ra trên mạng của máy tính chúng ta.

5.7.2 Phần cài đặt

Các WUI được hỗ trợ trên các hệ điều hành Linux, Unix, và hệ điều hành BSD với điều kiện hệ điều hành có khả năng chạy:

- OSSEC HIDS phiên bản 0,9-3 và mới hơn.
- Apache HTTP Server là phần mềm để thực hiện HTTP Web Server.
- PHP 4.1 và mới hơn: PHP là một sử dụng rộng rãi, ngôn ngữ general – purpose đặc biệt phù hợp để phát triển Web và có thể được nhúng vào HTML.

5.7.3 Thành phần

Web User Interface có nhiều tab, mỗi trong số đó phục vụ một mục đích, chức năng cụ thể.

- Main: Giao diện điều khiển chính.
- Search: Cho phép bạn tìm kiếm thông qua thu thập HIDS OSSEC cảnh báo.
- Integrity Checking: Cho phép bạn tìm kiếm thông qua thu thập từ cảnh báo syscheck OSSEC HIDS.
- Stats: Hiển thị thống kê về thu OSSEC HIDS cảnh báo.
- About: Hiển thị cấp giấy phép và thông tin bản quyền về các HIDS OSSEC và các WUI.

Hình 5.17: Thanh công cụ quản lí WUI của OSSEC

1. Main
   Main là giao diện chính của WUI cho phép người dùng có thông tin hợp lệ của WUI để xem được những thông tin diễn ra trong việc triển khai OSSEC HIDS. Gồm có ba phần chi tiết với nhiệm vụ cụ thể là: Available agents, Latest modified files, Latest events.

   - Available Agents
     Tất cả các cấu hình được hiển thị có sẵn trên tab Main. Các tên agent và địa chỉ IP liên kết được hiển thị cho từng agent. Nếu agent không hoạt động hoặc không thể kết nối đến máy chủ OSSEC HIDS, inactive được hiển thị bên cạnh tên agent. Nhấp chuột vào các agent cho bạn thông tin chi tiết hơn về máy tính mà OSSEC HIDS agent được cài đặt.

Hình 5.18: Giao diện hiển thị agent hiện có

   - Latest Modified Files
     - Các tập tin mới được sửa đổi (Latest Modified Files) cho thấy một danh sách đầy đủ của các tập tin thay đổi mới nhất cho tất cả các agent. Mỗi tập tin sửa đổi được liệt kê theo ngày mà dịch vụ syscheck Báo cáo thay đổi để các máy chủ OSSEC HIDS. Như đã thấy trong hình 7.2, khi bạn nhấn vào tập tin sửa đổi,chi tiết cho các tập tin được hiển thị.
     - Số lượng các tập tin sửa đổi phụ thuộc vào số lượng agent bạn có cấu hình. Điều này được xác định bởi agent_count + 4 phương trình, agent_count là số lượng agent bạn đã cấu hình. Nếu bạn có 10 agent, nó sẽ chỉ hiển thị 14 tập tin sửa đổi cuối cùng. Tương tự như vậy, nếu bạn chỉ có một agent, nó sẽ chỉ hiển thị năm tập tin sửa đổi cuối cùng. Phương trình agent_count + 4 được đặt ra để đảm bảo rằng các giao diện người dùng vẫn duy trì cấu trúc như agent thêm vào được cấu hình.

- Latest Events
  The Latest events đưa ra các sự kiện mới nhất mà được báo cáo bởi OSSEC HIDS agents cho OSSEC HIDS server. Bao gồm một số thông tin cơ bản như: Thời gian, chỉ số rule, cấp độ, địa chỉ IP agent, miêu tả, ...

2. Search
   Các WUI cung cấp một công cụ tìm kiếm cho phép bạn lọc các tiêu chí cụ thể và xem kết quả tìm kiếm trong thời gian thực hoặc thực hiện các phân tích lịch sử trên dữ liệu sự kiện được lưu trữ. Tab tìm kiếm được chia thành ba phần, mỗi phần có một mục đích cụ thể: Tùy chọn tìm kiếm thông báo (Alert search options), kết quả (results) và danh sách thông báo (Alert list)

   - Alert Search Options
     Cho phép ta xác định các tiêu chí tìm kiếm cảnh báo mà bạn muốn xem. Đồng thời nó còn cho phép chúng ta chọn một ngày và phạm vi thời gian cho các cảnh báo được lưu trữ mà chúng ta cần. Nếu chúng ta muốn xem cảnh báo trong thời gian thực thì ta chọn Real time monitoring và chọn các tiêu chí phù hợp. Một số tiêu chí như:
     - Minimum level: Đưa ra mức cảnh báo thấp nhất để lọc. Mặc định là tất cả (All) và phạm vi là từ 2 đến 15.
     - Pattern: Chỉ định một mô hình phù hợp để các cảnh báo được tạo ra. Mô hình đơn giản matching và công thức Perl regular có thể được sử dụng.
     - Srcip: Lọc trên một IP nguồn cụ thể trong thông báo được tạo ra.
     - Location: Bộ lọc trên một OSSEC HIDS agent cụ thể.

     Khi đã hoàn thành việc cấu hình tiêu chuẩn các tiêu chí của bộ lọc tìm kiếm cảnh báo, chúng ta phải bấm nút Tìm kiếm (Search) để lấy hồ sơ cảnh báo.
   - Results
     Phần kết quả cho thấy các kết quả truy vấn thông báo dựa trên các bộ lọc thiết lập trong Alert Search Options. Ở phía trên của phần kết quả là tổng số cảnh báo tìm thấy phù hợp với tiêu chí tìm kiếm. Thông tin sau đó được hiển thị trong ba nhóm riêng biệt cho phép chúng ta xem các phân tích kết quả bởi : Mức độ nghiêm trọng (Severity), Rule và IP nguồn (Src IP).
     Trong mỗi nhóm, chúng ta có thể tùy chọn để hiển thị hoặc ẩn các thông báo của các loại được chọn. Nhấp vào liên kết màu đỏ bên cạnh các mục nhập bộ lọc và có thể bao gồm các mục khác trong bộ lọc.

Ở dưới phần kết quả là các đường dẫn đến các sự kiện đầu tiên và cuối cùng nhìn thấy dựa trên bộ lọc cảnh báo hiện tại. Hai liên kết hiển thị ngày tháng và thời gian đóng dấu cho mỗi sự kiện.

- Alert List
  Danh sách cảnh báo sẽ hiển thị kết quả tìm kiếm với các bộ lọc được xác định trong Alert Search Options và phần Result.

3. Integrity Checking
   Phần kiểm tra tính toàn vẹn cung cấp hai tính năng rất quan trọng của WUI: một danh sách các tập tin bị sửa đổi của tất cả các agent, và một phương pháp dump toàn bộ syscheck cơ sở dữ liệu cho một agent cụ thể.
   Latest Modified Files (cho tất cả Agents):
   Phần tập tin sửa đổi mới nhất cho thấy một danh sách đầy đủ của các tập tin bị thay đổi mới nhất cho tất cả các agent. Mỗi tập tin sửa đổi được liệt kê theo ngày mà các dịch vụ syscheck báo thay đổi cho máy chủ OSSEC HIDS.
   Dump Database
   Kiểm tra tính toàn vẹn cho phép bạn cấu hình một agent, và dump toàn bộ nội dung của cơ sở dữ liệu syscheck cho agent đó. Dưới phần Latest Modified Files, phần Integrity Checking database cung cấp một danh sách hoàn chỉnh của tất cả các tập tin hoặc các khóa registry của Windows mà dịch vụ syscheck đã thông báo một sự thay đổi tính toàn vẹn. Một vài thông số như : Tên tập tin, mã hash (MD5 hay SHA-1), kích thước tập tin.

4. Stats
   WUI có một công cụ báo cáo thống kê mạnh mẽ cho phép nhanh chóng hiển thị của các cảnh báo của bạn OSSEC, thay đổi syscheck, và cảnh báo tường lửa cho bất kỳ ngày, tháng, hoặc năm. Điều này mang đến khả năng chọn một điểm và hiển thị các thông báo liên quan được tạo ra bởi các agent và thu thập bởi máy chủ.
   WUI được chia thành ba phần báo cáo chính, mỗi phần cung cấp một tổng hợp thu thập cảnh báo khác nhau: giá trị tổng hợp bởi mức độ nghiêm trọng, giá trị tổng hợp theo rule, tổng giá trị mỗi giờ.

   - Giá trị tổng hợp bởi mức độ nghiêm trọng (Aggregate values by severity) cho bạn thấy tổng số các cảnh báo được chia nhỏ bởi mức độ nghiêm trọng của các cảnh báo.
   - Giá trị tổng hợp theo rule (Aggregate Values by Rule) cho thấy tổng số các cảnh báo được chia nhỏ bởi các rule mà tạo ra các cảnh báo.
   - Tổng giá trị mỗi giờ (Total Values per Hour) cho thấy tổng số các cảnh báo được chia nhỏ theo giờ mà các cảnh báo đã báo cáo với

OSSEC HIDS

5. About
   Phần giới thiệu của WUI cung cấp bản quyền và thông tin giấy phép cho WUI và các OSSEC HIDS. Giấy phép chỉ ra rằng OSSEC HIDS là phần mềm miễn phí và rằng nó có thể phân phối hoặc sửa đổi theo các điều khoản của GNU General Public License (phiên bản 3). Giấy phép này như được xuất bản bởi Free Software Foundation.

Chương 6

Phân tích và thiết kế hệ thống

6.1 Ý tưởng đề xuất

Các hệ thống HIDS hiện nay sử dụng cơ chế rules-based, tuy cách này có lợi thế về mặt thời gian, nhưng trên thực tế, các kiểu tấn công luôn thay đổi, nên hệ thống phát hiện được những cuộc tấn công mà chưa được biết đến là 1 điều cần thiết.

Các giải thuật gom cụm được áp dụng cho ra các kết quả khá chính xác về việc phân cụm sự kiện mạng, giúp cho việc đánh giá cuộc tấn công mạng có thể có hướng tiếp cận mới khá khả quan.

Từ các kết luận trên, nhóm đưa ra ý tưởng như sau: tích hợp chương trình học máy vào trong hệ thống HIDS, để cải tiến quá trình phát hiện các hành vi xâm nhập, từ đó nâng cao tính an toàn, tính bảo mật cho hệ thống.

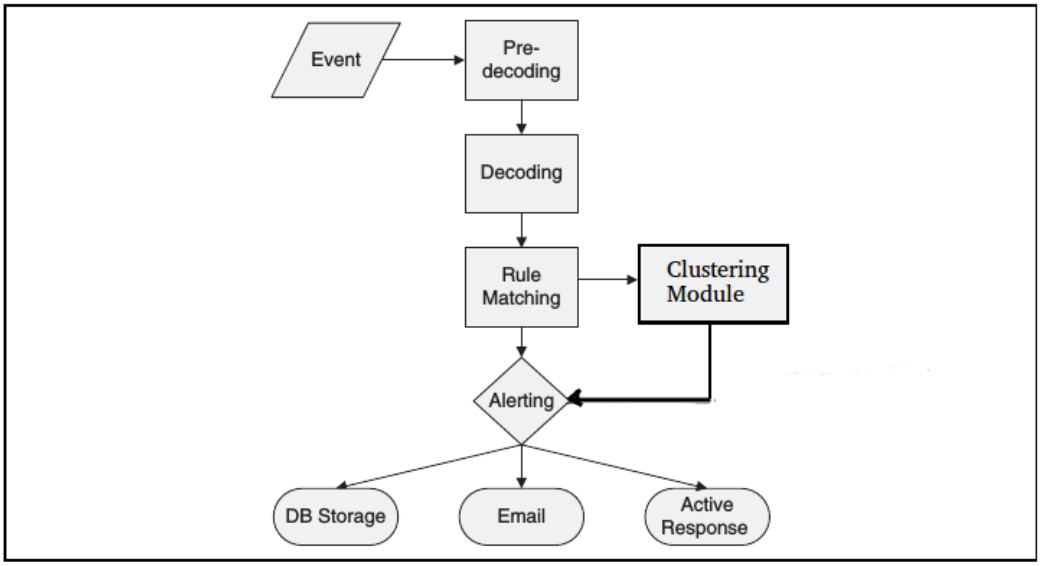
6.2 Cách thiết kế hệ thống

6.2.1 Nhận sự kiện

Mỗi khi chương trình trong máy tính hoạt động sẽ tạo ra các log và ghi vào trong file log, mỗi 1 log được ghi vào đều là 1 sự kiện. Module này sẽ đảm nhiệm việc bắt những đoạn log này cho hệ thống, hệ thống sẽ lấy những đoạn log này làm input để thực hiện công việc phát hiện xâm nhập.

6.2.2 Predecode

Module này thực hiện nhiệm vụ rút trích các thông tin cơ bản của sự kiện. Các thông tin như ngày, giờ, tên chương trình của sự kiện, người dùng thực hiện sự kiện đều được rút trích ra, để làm thông tin kiểm tra.



Hình 6.1: Cải tiến quá trình phát hiện xâm nhập trong OSSEC

Cơ chế rút trích trong module này là sử dụng 1 chuẩn do hệ thống tự định nghĩa, các đoạn log của các chương trình khác nhau đều được chuẩn hóa về dạng chuẩn này, từ đó hệ thống sẽ rút ra những thông tin cần thiết.

6.2.3 Decode

Module này thực hiện nhiệm vụ rút trích các thông tin chính của sự kiện. Các thông tin như việc đăng nhập thành công hoặc thất bại các người dùng, việc nhập sai mật khẩu, việc chặn gói tin trên ACL, ... đều được rút trích trong quá trình này.

Chúng ta có thể dễ dàng nhận thấy những thông tin trên vô cùng đa dạng. Cấu trúc của đoạn log hoàn toàn khác nhau, do vậy việc decode sẽ dễ ra khác nhau đối với từng sự kiện, dẫn đến việc mỗi 1 sự kiện đều có 1 decoder riêng.

Việc rút trích thông tin trước hết sẽ dự vào tên chương trình mà predecoder đã rút trích được, sau đó sẽ tiến hành so sánh lần lượt với các decoder của chương trình đó để xác định decoder dùng để rút trích thông tin.

Sau khi tiến hành quá trình rút trích, 1 dãy thông tin của quá trình predecode và quá trình decode được đưa vào quá trình sau để tiến hành so sánh.

6.2.4 Rule matching

Module này thực hiện công việc so sánh các thông tin rút trích được với những rule được định nghĩa sẵn trong hệ thống.

Quá trình này trước hết sẽ dựa vào decoder được sử dụng để rút trích thông tin, tìm ra tất cả các rule của decoder đó. Sau đó tiến hành so sánh lần lượt thông tin rút trích được từ sự kiện với từng luật. Do luật trong hệ thống được cấu hình theo sơ đồ cây, khi trùng luật ở node cha, hệ thống vẫn tiếp tục so sánh với tất cả các luật ở node con. Cảnh báo được phát theo quy định của luật cuối cùng được so sánh trùng.

Khi so sánh sự kiện với luật sẽ xảy ra 2 trường hợp sau:

1. Nếu phát hiện trùng: hệ thống sẽ phát ra cảnh báo.
2. Nếu phát hiện không trùng: dữ liệu rút trích sẽ được đưa vào Clustering module.

6.2.5 Clustering Module

Module này sẽ thực hiện công việc kiểm tra lần 2 cho thông tin rút trích được, việc kiểm tra này dựa vào học máy.

Module sẽ nhận toàn bộ dữ liệu mà hệ thống rút trích được, dựa vào mô hình đã được train với tập dữ liệu tốt, gom cụm sự kiện vào 1 trong 2 cụm tấn công hoặc bình thường.

Việc gom cụm này sẽ dựa vào các thuật toán gom cụm.

6.2.6 Alerting

Module này sẽ thực hiện nhiệm vụ phát ra cảnh báo khi phát hiện tấn công, được điều khiển bởi Module Rule matching và Module Clustering.

6.3 Đánh giá hệ thống

Với mô hình trên, nhóm tạo ra hệ thống HIDS kiểu hybrid, kết hợp giữa hệ thống Rules based và hệ thống Behavior based.

Các dữ liệu rút trích từ sự kiện trước hết phải thông qua bộ so sánh rule của bản thân OSSEC, nếu sự kiện không trùng mới đi qua hệ thống Clustering.

Như vậy, hệ thống mới vẫn giữ được tốc độ của việc phát hiện xâm nhập

trong hệ thống cũ, các dấu hiệu tấn công đã biết vẫn được sử dụng, đảm bảo việc chính xác trong khi xét các sự kiện đã biết. Bên cạnh đó, các sự kiện không có dấu hiệu trong rule, được kiểm tra lần 2 trong hệ thống clustering, tăng thêm tính bảo mật.

Ngoài ra, có thể thêm tính năng tự cập nhật rule. Một khi sự kiện được clustering module phân vào cụm tấn công, và được người dùng xác thực về sự tấn công đó, hệ thống có thể rút trích các thông tin của sự kiện và viết thành rule. Sau khi được cập nhật rule, khi gặp lại sự kiện tương tự, hệ thống có thể nhanh chóng phát hiện tấn công bằng rules.

Clustering module có thể thêm khả năng phân các cuộc tấn công thành nhiều cụm, chia thành từng loại tấn công, tăng tính rõ ràng cho dữ liệu.

Chương 7

Kết luận và hướng phát triển

Trong quá trình tìm hiểu kiến thức, nhóm nhận thấy có rất nhiều giải thuật học máy không giám sát trong việc phân cụm dữ liệu. Trong nội dung đề tài thực tập tốt nghiệp, nhóm đã trình bày nền tảng kiến thức của ba giải thuật gom cụm đơn giản và phù hợp nhất với nội dụng đề tai của nhóm đó là Kmeans, Mixture of Gaussian and the EM Algorithm và Self Oraganizing Maps. Bên cạnh đó, nhóm đã tìm hiểu nền tảng kiến thức của hai giải thuật giúp thu giảm số chiều của ma trận thuộc tính trước khi áp dụng giải thuật học máy vào ma trận là Principal Component Analysis và Singular Value Decomposition. Điều này giúp cho việc giảm thời gian xử lí gom cụm từ ma trận thuộc tính đồng thời còn giúp tăng độ chính xác.

Sau khi tìm hiểu các kĩ thuật học máy gom cụm và áp dụng vào tập dữ liệu đầu vào thì nhóm nhận thấy kĩ thuật N-gram và Mixture of Gaussian and the EM Algorithm cho kết quả gần với kết quả kì vọng của nhóm nhất mặc dù chưa có độ chính xác cao.

Nhóm đã thực hiện một mô hình áp dụng mới so với các công trình liên quan. Nhóm thực hiện chỉ một mô hình học máy giúp thu giảm thời gian xử lí dữ liệu đầu vào với độ chính xác trong ngưỡng chấp nhận được. Với OSSEC thì các đề tài trước chỉ xây dựng hệ thống rule cho hệ thống HIDS, nhóm đưa ra đề xuất đưa các kĩ thuật học máy gom cụm để xây dựng luật riêng cho OSSED HIDS.

Vì hạn chế về kiến thức mà nhóm chỉ thực hiện một số chức năng chính có sẵn của công cụ bảo mật OSSEC HIDS mà chưa áp dụng cụ thể học máy vào để tiến hành phân cụm các cuộc tấn công an ninh mạng.

Hướng phát triển của đề tài khi sang luận văn tốt nghiệp của nhóm là:

1. Thực hiện nâng cao hiệu suất và độ chính xác của giải thuật trong việc phân cụm tập dữ liệu đầu vào.

2. Áp dụng các kĩ thuật học máy vào hệ thống mã nguồn mở OSSEC HIDS đồng thời kết hợp các chức năng có sẵn để xây dựng một ứng dụng cụ thể có thể phát hiện xâm nhập, tấn công từ bên ngoài.

Tài liệu tham khảo

[1] John A. Bullinaria, *Self Organizing Maps: Fundamentals- Introduction to Neural Networks: Lecture 16*. 2004

[2] Shyam M. Guthikonda, *Kohonen Self-Organizing Maps*. Wittenberg University. 2005

[3] Mehotra, K., Mohan, C. K., Ranka. S, *Self-Organizing Maps (SOMs)*. Elements of Artificial Neural Networks. MIT Press. 1997.

[4] Kirk Baker, *Singular Value Deomposition Tutorial*. 2005.

[5] Eleazar Eskin, Wenke Lee, Salvatore J. Stolfo, *Modeling System Calls for Intrusion Detection with Dynamic Window Sizes*. DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings. 2001.

[6] Léon Bottou, Yoshua Bengio. *Convergence Properties of the K-Means Algorithms*. MIT Press. 1995.

[7] Gideon Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks*. 2014.

[8] Andrew Ng. *Lecture Notes – CS229 Machine Learning*. 2012

[9] Andrew Hay, Daniel Cid, Rory Bay, *OSSEC Host-Based Intrusion Detection Guide*. Syngress. 2008