|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**  **KHOA CÔNG NGHỆ THÔNG TIN**  **----🙚🕮🙘----** |
| **Logo  Description automatically generated** |
| **LÊ ĐỖ TRÀ MY: 18133030**  **TRẦN THỊ LỆ XUÂN: 18133066** |
| Đề tài: |
| **PHÁT HIỆN XÂM NHẬP MẠNG**  **VỚI HỌC SÂU** |
| **KHÓA LUẬN TỐT NGHIỆP**  **NGÀNH KỸ THUẬT DỮ LIỆU** |
| GIÁO VIÊN HƯỚNG DẪN  **ThS. Quách Đình Hoàng** |
| **KHÓA 2018 - 2022** |

Shape, square

Description automatically generated

Shape, square

Description automatically generated

|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**  **KHOA CÔNG NGHỆ THÔNG TIN**  **----🙚🕮🙘----** |
| **Logo  Description automatically generated** |
| **LÊ ĐỖ TRÀ MY: 18133030**  **TRẦN THỊ LỆ XUÂN: 18133066** |
| Đề tài: |
| **PHÁT HIỆN XÂM NHẬP MẠNG**  **VỚI HỌC SÂU** |
| **KHÓA LUẬN TỐT NGHIỆP**  **NGÀNH KỸ THUẬT DỮ LIỆU** |
| GIÁO VIÊN HƯỚNG DẪN  **ThS. Quách Đình Hoàng** |
| **KHÓA 2018 - 2022** |

|  |  |  |
| --- | --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP.HCM**  **KHOA CNTT**  **\*\*\*\*\*\*\*\*** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  **\*\*\*\*\*\*\*\*** | |
| **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN** | | |
| Họ và tên Sinh viên 1: Lê Đỗ Trà My  Họ và tên Sinh viên 2: Trần Thị Lệ Xuân | | MSSV: 18133030  MSSV: 18133066 |
| Ngành: Kỹ thuật dữ liệu  Tên đề tài: TÌM HIỂU CHỦ ĐỀ PHÁT HIỆN XÂM NHẬP MẠNG  Họ và tên giáo viên hướng dẫn: ThS.Quách Đình Hoàng | | |
| **NHẬN XÉT**  1. Về nội dung đề tài khối lượng thực hiện: ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 2. Ưu điểm: ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 3. Khuyết điểm  ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 4. Điểm: | | |
| Tp*. Hồ Chí Minh, ngày    tháng     năm   2022*  Giáo viên hướng dẫn  *(Ký & ghi rõ họ tên)* | | |

|  |  |  |
| --- | --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP.HCM**  **KHOA CNTT**  **\*\*\*\*\*\*\*\*** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  **\*\*\*\*\*\*\*\*** | |
| **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN** | | |
| Họ và tên Sinh viên 1: Lê Đỗ Trà My  Họ và tên Sinh viên 2: Trần Thị Lệ Xuân | | MSSV: 18133030  MSSV: 18133066 |
| Ngành: Kỹ thuật dữ liệu  Tên đề tài: TÌM HIỂU CHỦ ĐỀ PHÁT HIỆN XÂM NHẬP MẠNG  Họ và tên giáo viên phản biện: | | |
| **NHẬN XÉT**  1. Về nội dung đề tài khối lượng thực hiện: ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 2. Ưu điểm: ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 3. Khuyết điểm  ………………………………………………………………………………………………………  ………………………………………………………………………………………………………  ……………………………………………………………………………………………………… | | |
| 4. Điểm: | | |
| Tp*. Hồ Chí Minh, ngày    tháng     năm   2022*  Giáo viên phản biện  *(Ký & ghi rõ họ tên)* | | |

# **LỜI CAM ĐOAN**

Tiểu luận này là công trình nghiên cứu của chúng tôi, được thực hiện dưới sự hướng dẫn khoa học của thầy Quách Đình Hoàng. Các số liệu, những kết luận nghiên cứu và sản phẩm được tạo ra bởi chúng tôi được trình bày trong khoá luận này là trung thực.

Chúng tôi xin hoàn toàn chịu trách nhiệm về lời cam đoan này.

|  |  |
| --- | --- |
| Sinh viên thực hiện  (Ký và ghi rõ họ tên) | Sinh viên thực hiện  (Ký và ghi rõ họ tên) |
| **Lê Đỗ Trà My** | **Trần Thị Lệ Xuân** |

# **LỜI CẢM ƠN**

Một kỳ thực hiện khóa luận tốt nghiệp đã trôi qua nhưng để lại trong chúng tôi rất nhiều cảm xúc. Chúng tôi xin được gửi lời cảm ơn chân thành nhất đến Thầy Quách Đình Hoàng. Mặc dù trong tình hình dịch phức tạp và học online nhưng thầy đã cung cấp tài liệu và hướng dẫn tận tình cho chúng tôi trong suốt quá trình thực hiện tiểu luận chuyên ngành, Thầy luôn theo dõi tiến độ và giải đáp, chia sẻ giúp chúng tôi vượt qua những khó khăn. Chúng tôi rất trân quý sự tâm huyết và trách nhiệm của Thầy trong công việc giảng dạy và truyền đạt kiến thức.

Chúng tôi cũng xin gửi lời cảm ơn sâu sắc đến Thầy Cô khoa Công nghệ Thông tin - Đại học Sư phạm Kỹ thuật TP.HCM đã đồng hành và hỗ trợ chúng tôi trong suốt quá trình học tập và thực hiện tiểu luận. Chúng tôi xin cảm ơn trường Đại học Sư phạm Kỹ thuật đã tạo nhiều điều kiện thuận lợi cho hoạt động phục vụ học tập của sinh viên chúng tôi, đặc biệt là thư viện số với nguồn tri thức vô tận. Chúng tôi cũng gửi lời cảm ơn chân thành đến các bạn khóa 2018 ngành Kỹ thuật Dữ liệu, cảm ơn những góp ý và chia sẻ quý giá từ tất cả các bạn. Cảm ơn sự động viên từ các bạn để nhóm chúng tôi có thể giữ vững tinh thần và thực hiện khóa luận đúng tiến độ.

Những giá trị cốt lõi nhà trường, Thầy Cô và bạn bè mang đến, chúng tôi sẽ luôn ghi nhớ để làm động lực thúc đẩy bản thân phát triển và hoàn thiện hơn nữa. Cuối cùng, chúng tôi xin cám ơn đến tác giả của những bài báo khoa học mà chúng tôi đã tham khảo. Các bài báo này giúp chúng tôi tiếp thu được nhiều kiến thức mới và quan trọng là hiểu rõ hơn về đề tài đang nghiên cứu. Chúng tôi cũng nhận thấy bản thân có những khuyết điểm và thiếu sót cần cố gắng cải thiện để tốt hơn, hướng tới mục tiêu lớn trong tương lai.

Xin chân thành cảm ơn!

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM | |
| KHOA CÔNG NGHỆ THÔNG TIN | |
| --o-- | |
| ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP | |
| Họ và tên SV thực hiện 1: Lê Đỗ Trà My | MSSV: 18133066 |
| Họ và tên SV thực hiện 2: Trần Thị Lệ Xuân | MSSV: 18133030 |
| Thời gian làm luận văn: Từ: 04/2022 Đến: 07/2022  Chuyên ngành: Kỹ thuật dữ liệu  Tên luận văn: Tìm hiểu và áp dụng cho bài toán phát hiện xâm nhập mạng  Giáo viên hướng dẫn: Ths. Quách Đình Hoàng | |

**NHIỆM VỤ CỦA LUẬN VĂN**

Nhiệm vụ của luận văn là áp dụng một vài thuật toán học máy và học sâu vào tập dữ liệu phổ biến cho vấn đề xâm nhập mạng. Sau đó, luận văn tập trung vào tiến hành đánh giá hiệu quả một số thuật toán trong phát hiện xâm nhập mạng. Để đạt được điều đó, chúng tôi tập trung tìm hiểu một số vấn đề sau:

1. Tìm hiểu tổng quan về phát hiện xâm nhập mạng

2. Tìm hiểu tổng quan về về các thuật toán học máy

3. Tìm hiểu tổng quan các thuật toán học sâu

4. Tìm hiểu về các tập dữ liệu mạng phổ biến được sử dụng cho bài toán phát hiện xâm nhập mạng

5. Đánh giá và so sánh một số thuật toán học máy cho bài toán phát hiện xâm nhập mạng

**KẾ HOẠCH THỰC HIỆN**

|  |  |  |
| --- | --- | --- |
| STT | Thời gian | Công việc dự kiến |
| 1 | 14/3 - 27/3 | Tìm hiểu về bài toán xâm nhập mạng và định hướng giải quyết bài toán.  Tìm hiểu về bộ dữ liệu được sử dụng. |
| 2 | 28/3 – 10/4 | Tìm hiểu khái quát về học máy, học sâu và tổng quan về xâm nhập mạng. |
| 3 | 11/4 – 17/4 | Tìm hiểu về neural network |
| 4 | 18/4 – 24/4 | Tìm hiểu về RNN |
| 5 | 25/4 – 8/5 | Tìm hiểu về LSTM |
| 6 | 9/5 – 15/5 | Tìm hiểu về LSTM (tiếp theo) |
| 7 | 16/5 – 29/5 | Viết báo cáo  Tìm hiểu tiền xử lí dữ liệu với tập dữ liệu đầu vào đã chọn |
| 8 | 30/5 – 5/6 | Viết báo cáo.  Tìm hiểu các thư viện liên quan, Google Colab và công cụ WEKA.  Thực nghiệm thử trên một số thuật toán cổ điển |
| 9 | 6/6 – 12/6 | Viết báo cáo.  Thực nghiệm theo hướng không tiến hành lựa chọn thuộc tính |
| 10 | 13/6 – 19/6 | Viết báo cáo.  Thực nghiệm theo hướng có tiến hành lựa chọn thuộc tính |
| 11 | 20/6 – 26/6 | Viết báo cáo.  Thực nghiệm theo hướng có tiến hành lựa chọn thuộc tính (tiếp theo) |
| 12 | 27/6 – 3/7 | Viết báo cáo.  Thực nghiệm theo hướng có tiến hành lựa chọn thuộc tính (tiếp theo).  Chuẩn bị phần thuyết trình. |
| 13 | 4/7 – 10/7 | Hoàn thiện báo cáo.  Thực nghiệm theo hướng có tiến hành lựa chọn thuộc tính (tiếp theo).  Chuẩn bị phần thuyết trình |

|  |  |
| --- | --- |
| Ý kiến giảng viên hướng dẫn | TP. Thủ Đức, Ngày …. Tháng…Năm |
| (Ký và ghi rõ họ tên) | Người viết đề cương |

**MỤC LỤC**

[**LỜI CAM ĐOAN** iii](#_Toc108949687)

[**LỜI CẢM ƠN** iv](#_Toc108949688)

[**DANH MỤC HÌNH VẼ** x](#_Toc108949689)

[**DANH MỤC BẢNG** xii](#_Toc108949690)

[**PHẦN 1: MỞ ĐẦU** 2](#_Toc108949691)

[**1.** **GIỚI THIỆU BÀI TOÁN** 2](#_Toc108949692)

[**2.** **MỤC TIÊU** 2](#_Toc108949693)

[**3.** **BỐ CỤC** 3](#_Toc108949694)

[**PHẦN 2: NỘI DUNG** 4](#_Toc108949695)

[**CHƯƠNG 1: TỔNG QUAN VỀ MẠNG** 4](#_Toc108949696)

[**1.1** **KHÁI NIỆM VỀ XÂM NHẬP MẠNG** 4](#_Toc108949697)

[**1.2** **MỘT SỐ KIỂU TẤN CÔNG MẠNG** 5](#_Toc108949698)

[**1.2.1** **Tấn công mạng Malware** 5](#_Toc108949699)

[**1.2.2** **Tấn công từ chối dịch vụ DoS (Denial of service)** 5](#_Toc108949700)

[**1.2.3** **Tấn công từ chối dịch vụ phân tán DDoS (Distributed Denial of Service)** 6](#_Toc108949701)

[**1.3** **MỘT SỐ KỸ THUẬT PHÒNG CHỐNG XÂM NHẬP MẠNG** 7](#_Toc108949702)

[**1.3.1** **Tường lửa (Firewall)** 7](#_Toc108949703)

[**1.3.2** **Mã hóa dữ liệu** 7](#_Toc108949704)

[**1.3.3** **Mạng riêng ảo (Virtual private network – VPN)** 8](#_Toc108949705)

[**1.4** **HỆ THỐNG PHÁT HIỆN XÂM NHẬP MẠNG (INSTRUCTION DETECT SYSTEM - IDS)** 9](#_Toc108949706)

[**1.4.1** **Khái niệm** 9](#_Toc108949707)

[**1.4.2** **Phân loại** 9](#_Toc108949708)

[**1.4.3** **Ưu nhược điểm** 10](#_Toc108949709)

[**CHƯƠNG 2: TỔNG QUAN VỀ MACHINE LEARNING** 12](#_Toc108949710)

[**2.1 KHÁI QUÁT VỀ HỌC MÁY (MACHINE LEARNING)** 12](#_Toc108949711)

[**2.1.1 Học máy là gì?** 12](#_Toc108949712)

[**2.1.2 Phân nhóm thuật toán** 12](#_Toc108949713)

[*2.1.2.1 Thuật toán phân theo phong cách học tập* 12](#_Toc108949714)

[*2.1.2.2 Các thuật toán được nhóm theo sự tương đồng* 14](#_Toc108949715)

[**2.2 MỘT SỐ THUẬT TOÁN CỔ ĐIỂN** 21](#_Toc108949716)

[**2.2.1 Thuật toán Support Vector Machine (SVM)** 21](#_Toc108949717)

[**2.2.2 Thuật toán naïve bayes** 23](#_Toc108949718)

[**2.2.3 Thuật toán Logistic Regression** 24](#_Toc108949719)

[**2.2.4 Thuật toán Random Forest** 25](#_Toc108949720)

[**CHƯƠNG 3: TỔNG QUAN VỀ HỌC SÂU (DEEP LEARNING)** 27](#_Toc108949721)

[**3.1 KHÁI QUÁT VỀ HỌC SÂU** 27](#_Toc108949722)

[**3.1.1 Học sâu là gì?** 27](#_Toc108949723)

[**3.2** **KHÁI QUÁT VỀ MẠNG NƠ-RON NHÂN TẠO** 28](#_Toc108949724)

[**3.2.1** **Khái niệm** 28](#_Toc108949725)

[**3.2.2** **Các thành phần cơ bản** 29](#_Toc108949726)

[**3.2.3** **Một số kiến trúc mạng nơ-ron nhân tạo** 30](#_Toc108949727)

[**3.3** **MỘT SỐ THUẬT TOÁN VỀ HỌC SÂU** 32](#_Toc108949728)

[**3.3.1** **Mạng neural hồi quy – Recurrent Neural Network (RNN)** 32](#_Toc108949729)

[*3.3.1.1* *Khái niệm* 32](#_Toc108949730)

[*3.3.1.2* *Phân loại bài toán RNN* 33](#_Toc108949731)

[*3.3.1.3* *Mô hình* 34](#_Toc108949732)

[*3.3.1.4* *Hàm loss* 36](#_Toc108949733)

[*3.3.1.5* *Nhược điểm* 36](#_Toc108949734)

[*3.3.1.6* *Giải pháp* 37](#_Toc108949735)

[**3.3.2** **Bộ nhớ ngắn dài hạn – Long Short Term Memory (LSTM)** 37](#_Toc108949736)

[*3.3.2.1* *Giới thiệu* 37](#_Toc108949737)

[*3.3.2.2* *Mô hình* 37](#_Toc108949738)

[*3.3.2.3* *Lan truyền xuôi và lan truyền ngược* 39](#_Toc108949739)

[**CHƯƠNG 4: XÂY DỰNG MÔ HÌNH** 46](#_Toc108949740)

[**4.1.** **MÔI TRƯỜNG CÀI ĐẶT VÀ CÁC THƯ VIỆN LIÊN QUAN** 46](#_Toc108949741)

[**4.1.1.** **Môi trường cài đặt** 46](#_Toc108949742)

[**4.1.2.** **Một số thư viện sử dụng** 47](#_Toc108949743)

[**4.2.** **MÔ TẢ DỮ LIỆU** 50](#_Toc108949744)

[**4.3** **TIỀN XỬ LÝ** 54](#_Toc108949745)

[**4.4** **THỰC NGHIỆM** 58](#_Toc108949746)

[**4.4.1** **Phân loại không lựa chọn thuộc tính** 59](#_Toc108949747)

[**4.4.2** **Phân loại có lựa chọn thuộc tính** 62](#_Toc108949748)

[**4.5** **KẾT QUẢ VÀ ĐÁNH GIÁ** 68](#_Toc108949749)

[**PHẦN 3: KẾT LUẬN** 71](#_Toc108949750)

[**1.** **ĐÓNG GÓP** 71](#_Toc108949751)

[**2.** **HẠN CHẾ** 71](#_Toc108949752)

[**3.** **HƯỚNG PHÁT TRIỂN** 71](#_Toc108949753)

[DANH MỤC THAM KHẢO 72](#_Toc108949754)

# **DANH MỤC HÌNH VẼ**

[Hình 2. 1 Học có giám sát (supervised learning) [9] 9](#_Toc108536825)

[Hình 2. 2 Học không giám sát (unsupervised learning) [9] 10](#_Toc108536826)

[Hình 2. 3 Học bán giám sát (semi-supervised learning) [9] 10](#_Toc108536827)

[Hình 2. 4 Thuật toán hồi quy (regression algorithms) [9] 11](#_Toc108536828)

[Hình 2. 5 Thuật toán instance-based (instance-based algorithms) [9] 12](#_Toc108536829)

[Hình 2. 6 Thuật toán regularization (regularization algorithms) [9] 13](#_Toc108536830)

[Hình 2. 7 Thuật toán cây quyết định (decision tree algorithms) [9] 13](#_Toc108536831)

[Hình 2. 8 Thuật toán Bayesian (Bayesian Algorithms) [9] 14](#_Toc108536832)

[Hình 2. 9 Thuật toán gom cụm (clustering algorithms) [9] 15](#_Toc108536833)

[Hình 2. 10 Thuật toán mạng neural nhân tạo (artifical neural network algorithms) [9] 15](#_Toc108536834)

[Hình 2. 11 Thuật toán deep learning (deep learning algorithms) [9] 16](#_Toc108536835)

[Hình 2. 12 Thuật toán giảm chiều (dimensionality reduction algorithms) [9] 16](#_Toc108536836)

[Hình 2. 13 Thuật toán tổng hợp (ensemble algorithms) [9] 17](#_Toc108536837)

[Hình 2. 14 Phân tích bài toán SVM [10] 18](#_Toc108536838)

[Hình 2. 15 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn [10] 19](#_Toc108536839)

[Hình 2. 18 Đồ thị hàm sigmoid [30] 22](#_Toc108536840)

[Hình 2. 19 Mô hình RF dang phân loại đơn giản với Ntree = 3 [31] 23](#_Toc108536841)

[Hình 3. 1 Mối liên quan của Deep Learning, Machine Learning, Artifical Intelligence [11] 24](#_Toc108443838)

[Hình 3. 2 Cấu trúc một mạng nơ-ron sinh học [12] 25](#_Toc108443839)

[Hình 3. 3 Mô hình của một nơ-ron nhân tạo được gán nhãn k [13] 26](#_Toc108443840)

[Hình 3. 4 Kiến trúc mạng Neural Network [14] 27](#_Toc108443841)

[Hình 3. 5 Perceptron neural [15] 28](#_Toc108443842)

[Hình 3. 6 Cấu trúc mạng nơ-ron hàm cơ sở xuyên tâm [16] 28](#_Toc108443843)

[Hình 3. 7 RNN rút gọn (trái) và RNN triển khai cụ thể (phải) [19] 32](#_Toc108443844)

[Hình 3. 8 Mô hình RNN đơn giản [20] 32](#_Toc108443845)

[Hình 3. 9 Công thức minh họa trạng thái ẩn ht [19] 32](#_Toc108443846)

[Hình 3. 10 Công thức minh họa output yt [19] 33](#_Toc108443847)

[Hình 3. 11 Mô hình biểu diễn kiến trúc bên trong của một tế bào LSTM [21] 35](#_Toc108443848)

[Hình 4. 1 Cấu trúc phần cứng Colab cung cấp [22] 43](#_Toc108949374)

[Hình 4. 2 Minh họa graph trong Tensorflow [23] 44](#_Toc108949375)

[Hình 4. 3 Minh họa sử dụng kỹ thuật mã hóa one-hot trong bài toán 52](#_Toc108949376)

[Hình 4. 4 Minh họa sự phân bố nhãn kết nối trên dữ liệu huấn luyện 55](#_Toc108949377)

[Hình 4. 5 Minh họa sự phân bố nhãn kết nối trên dữ liệu thử nghiệm 56](#_Toc108949378)

[Hình 4. 6 Cấu trúc mạng LSTM 57](#_Toc108949379)

[Hình 4. 7 Quá trình huấn luyện LSTM 58](#_Toc108949380)

[Hình 4. 8 Quá trình huấn luyện LSTM 1 62](#_Toc108949381)

[Hình 4. 9 Quá trình huấn luyện LSTM 2 62](#_Toc108949382)

[Hình 4. 10 Cấu trúc LSTM 3, LSTM 4 và LSTM 5 63](#_Toc108949383)

[Hình 4. 11 Quá trình huấn luyện LSTM 3 64](#_Toc108949384)

[Hình 4. 12 Quá trình huấn luyện LSTM 4 64](#_Toc108949385)

[Hình 4. 13 Quá trình huấn luyện LSTM 5 65](#_Toc108949386)

# **DANH MỤC BẢNG**

[Bảng 3. 1 Phân loại chi tiết các bài toán RNN [18] 30](#_Toc105831091)

[Bảng 4. 1 Thông tin các thuộc tính của bộ dữ liệu 48](#_Toc108536804)

[Bảng 4. 2 Minh họa cho kỹ thuật mã hóa one-hot 52](#_Toc108536805)

[Bảng 4. 3 Minh họa cho kỹ thuật mã hóa one-hot (tiếp theo) 52](#_Toc108536806)

[Bảng 4. 4 Minh họa cách biến đổi cho thuộc tính nhãn kết nối 53](#_Toc108536807)

[Bảng 4. 5 Minh họa kết quả cho phân loại không lựa chọn thuộc tính 58](#_Toc108536808)

[Bảng 4. 6 Mô tả kết quả trên một số thuật toán khác 59](#_Toc108536809)

[Bảng 4. 7 Mô tả kết quả trên LSTM 60](#_Toc108536810)

[Bảng 4. 8 Tổng hợp kết quả thực nghiệm 65](#_Toc108536811)

# **PHẦN 1: MỞ ĐẦU**

1. **GIỚI THIỆU BÀI TOÁN**

Hiện nay, chúng ta đang bước vào cuộc cách mạng 4.0 với sự bùng nổ và phát triển mạnh mẽ của Internet kèm theo nhu cầu trao đổi thông tin dữ liệu ngày càng lớn và đa dạng. Đi kèm theo sự phát triển mạnh mẽ đó là các yếu tố: tốc độ, chất lượng, bảo mật, sự đa dạng các dịch vụ,... Trong đó, bảo mật là một trong những vấn đề quan trọng nhất đối với cả nhà cung cấp dịch vụ cũng như người sử dụng, không chỉ đối với các cá nhân mà còn đặc biệt quan trọng trong các ngành mang tính đặc thù yêu cầu về bảo mật cao như quân sự, ngân hàng tài chính, …

Vấn đề bảo mật cũng trở thành mục tiêu lợi dụng của những kẻ tấn công, xâm nhập trái phép nhằm thực hiện những mưu đồ xấu, đe dọa tới tính an toàn về bảo mật thông tin của các tổ chức hay những người dùng kết nối mạng. Mặc dù, mỗi hệ thống máy tính đều có những cơ chế tự bảo vệ riêng nhưng có thể chưa đủ để phát hiện hay ngăn chặn những cuộc tấn công ngày một tinh vi hơn.

Vấn đề đặt ra là làm sao xây dựng được một hệ thống có thể phát hiện sớm và có hiệu quả các cuộc tấn công hay xâm nhập trái phép từ đó đưa ra những cảnh báo và biện pháp xử lý kịp thời. Những hệ thống phát hiện xâm nhập mạng đã được xây dựng vẫn chưa đủ linh hoạt, khả năng mở rộng không cao, cũng như không đủ mạnh để đối phó với các cuộc tấn công nói trên. Một vài nghiên cứu gần đây đưa ra một hướng tiếp cận mới dựa vào học máy cho bài toán phát hiện xâm nhập mạng.

Trong khóa luận này, chúng tôi nghiên cứu và áp dụng các kỹ thuật học máy và học sâu cho bài toán hiện xâm nhập mạng.

1. **MỤC TIÊU**

Mục tiêu: Tìm hiểu bài toán phát hiện xâm nhập mạng với học sâu

Nhiệm vụ:

* Tìm hiểu tổng quan về phát hiện xâm nhập mạng
* Tìm hiểu tổng quan về các thuật toán học máy
* Tìm hiểu tổng quan về các thuật toán học sâu
* Tìm hiểu về các tập dữ liệu mạng phổ biến được sử dụng cho bài toán phát hiện xâm nhập mạng
* Đánh giá và so sánh một số thuật toán học máy cho bài toán phát hiện xâm

1. **BỐ CỤC**

Chúng tôi định chia bố cục tiểu luận thành các phần như sau:

* Phần 1: Mở đầu
* Phần 2: Nội dung. Phần này gồm 4 chương
  + Chương 1: Tổng quan về mạng
  + Chương 2: Tổng quan về học máy
  + Chương 3: Tổng quan về học sâu
  + Chương 4: Xây dựng và đánh giá mô hình
* Phần 3: Kết luận
* Tài liệu tham khảo

# 

# **PHẦN 2: NỘI DUNG**

## **CHƯƠNG 1: TỔNG QUAN VỀ MẠNG**

* 1. **KHÁI NIỆM VỀ XÂM NHẬP MẠNG**

Tấn công mạng (cyber attack) là hình thức xâm nhập vào các hệ thống mạng máy tính, cơ sở dữ liệu, hạ tầng mạng, website hoặc thiết bị kỹ thuật số của cá nhân/tổ chức, nhằm mục đích làm tổn hại đến quyền riêng tư của dữ liệu được lưu trữ trong đó hoặc hủy hoại thiết bị, hệ thống cho các mục đích xấu.

Mục tiêu của một cuộc tấn công mạng rất đa dạng, có thể là vi phạm dữ liệu (đánh cắp, thay đổi, mã hóa, phá hủy), cũng có thể nhắm tới sự toàn vẹn của hệ thống (gây gián đoạn, cản trở dịch vụ), hoặc lợi dụng tài nguyên của nạn nhân (hiển thị quảng cáo, mã độc đào tiền ảo) [[1]](#_Danh_mục_tham).

Cụm từ “tấn công mạng” có 2 nghĩa hiểu:

+ Hiểu theo cách tích cực (positive way): tấn công mạng (penetration testing) là việc hacker mũ trắng xâm nhập vào một hệ thống mạng, thiết bị hay website để tìm ra những lỗ hổng bảo mật và các rủi ro nhằm bảo vệ tổ chức, doanh nghiệp.

+ Hiểu theo cách tiêu cực (negative way): tấn công mạng (network attack) là việc hacker mũ đen tấn công vào một hệ thống mạng, thiết bị hay website để thay đổi, phá hoại tổ chức, doanh nghiệp.

***Đối tượng bị tấn công***

Có thể là cá nhân, doanh nghiệp, các tổ chức chính phủ hoặc phi chính phủ, cơ quan nhà nước, thậm chí đối tượng có thể là cả một quốc gia. Tuy nhiên, đối tượng phổ biến nhất của các cuộc tấn công mạng là các doanh nghiệp, do mục tiêu chính của những kẻ tấn công là vì lợi nhuận.

***Mục đích tấn công***

Bên cạnh những mục đích phổ biến như trục lợi phi pháp, tống tiền doanh nghiệp, hiện thị quảng cáo kiếm tiền, thì còn tồn tại một số mục đích khác phức tạp và nguy hiểm hơn: cạnh tranh không lành mạnh giữa các doanh nghiệp, tấn công an ninh hoặc kinh tế của một quốc gia, tấn công đánh sập một tổ chức tôn giáo, v.v [[2]](#_Danh_mục_tham).

* 1. **MỘT SỐ KIỂU TẤN CÔNG MẠNG**
     1. **Tấn công mạng Malware**

Malware là kiểu tấn công mạng được nhiều người biết đến nhất nó còn được gọi là phần mềm ác ý hoặc phần mềm độc hại được viết ra nhằm mục đích xâm nhập trái phép, vô hiệu hóa hệ thống bảo mật và mạng, cản trở hoạt động của website, theo dõi thiết bị. Khi cài đặt mát tính, nó có thể truy cập và chia sẻ dữ liệu máy tính từ xa.

Các hacker có thể tấn công malware thông qua hình thức lừa đảo bằng cách gửi các email có chứa các phần mềm độc hại đến người dùng. Phần mềm độc hại cao cấp có thể sử dụng các kỹ thuật lẩn tránh và lừa đảo để tránh người dùng và các sản phẩm bảo mật. Để tấn công bằng hình thức malware các hacker đã sử dụng một số phần mềm như: viruses, trojans, rootkits, worm, spyware, … [[3]](#_TÀI_LIỆU_THAM_1)

Một số dấu hiệu khi máy tính người dùng bị tấn công mạng malware:

+ Hệ thống liên tục gặp sự cố, màn hình hiển thị màu xanh

+ Tốc độ xử lý hệ thống chậm

+ Dung lượng ổ cứng và tài nguyên giảm bất thường

+ Mất và lỗi các file dữ liệu

* + 1. **Tấn công từ chối dịch vụ DoS (Denial of service)**

*+ Tấn công kiểu SYN flood (half-open attack):* là một dạng của tấn công từ chối dịch vụ phân tán bằng việc lợi dụng cách thức hoạt động của kết nối TCP/IP gửi liên tục các packet tin yêu cầu kết nối ban đầu (SYN). Những kẻ tấn công áp đảo tất cả các cổng có sẵn trên server được chọn làm mục tiêu, khiến các thiết bị client hợp pháp được đáp ứng một cách rất chậm chạp hoặc không được đáp ứng kịp thời.

*+ Tấn công kiểu land attack:* tương tự như SYN flood, nhưng hacker sử dụng chính IP của mục tiêu cần tấn công để dùng làm địa chỉ IP nguồn trong gói tin, đẩy mục tiêu vào một vòng lặp vô tận khi cố gắng thiết lập kết nối với chính nó.

*+ Tấn công UDP flood:* hacker thực hiện bằng cách gửi một số lượng lớn các gói tin UDP có kích thước lớn đến hệ thống mạng, khi hệ thống bị tấn công UDP flood sẽ bị quá tải và chiếm hết băng thông của đường truyền, vì thế nó gây ra những ảnh hưởng rất lớn đến đương truyền, tốc độ của mạng, gây khó khăn cho người dùng khi truy cập vào mạng này. UDP Flood cần có ít nhất 2 hệ thống máy tham gia. Hackers tự làm hệ thống của mình đi vào vòng lặp trao đổi các dữ liệu qua giao thức UDP bằng việc giả mạo địa chỉ IP của các gói tin là địa chỉ lookback và gửi gói tín đến máy của victim thông qua cổng UDP echo (7). Máy của nạn nhân (victim) sẽ yêu cầu (request) các gói tin do 127.0.0.11 (chính nó) gửi đến, kết quả là nó sẽ đi vòng một vòng lặp vô tận.

*+ Tấn công smurf attack:* Kiểu tấn công này cần một hệ thống rất quan trọng, đó là mạng khuyếch đại. Hacker dùng địa chỉ của máy tính cần tấn công bằng cách gửi gói tin ICMP echo cho toàn bộ mạng (broadcast). Các máy tính trong mạng sẽ đồng loạt gửi gói tin ICMP reply cho máy tính mà hacker muốn tấn công. Kết quả là máy tính này sẽ không thể xử lý kịp thời một lượng lớn thông tin và dẫn tới bị treo máy. Hai thành phần chính của tấn công smurf là việc sử dụng các gói tin giả mạo địa chỉ IP của nạn nhân và địa chỉ broadcast [[4]](#_TÀI_LIỆU_THAM_1).

* + 1. **Tấn công từ chối dịch vụ phân tán DDoS (Distributed Denial of Service)**

Tấn công DDoS chính là một phương pháp tấn công bằng nhiều máy tính hoặc nhiều hệ thống máy tính yêu cầu tài nguyên trên máy chủ đích cùng 1 thời điểm. Làm quá tải tài nguyên của hệ thống máy chủ đích, khiến hệ thống bị chậm đi một cách đáng kể hoặc người dùng không còn truy cập và sử dụng được nữa. Đây là một biến thể của tấn công từ chối dịch vụ, nó gây ngập hệ thống bằng các gói tin TCP SYN, UDP SYN hay là các gói tin ICMP. Mục đích cuối cùng của loại tấn công này là làm cho hệ thống của nạn nhân bị tắc nghẽn và gây nên sự ùn tắc dữ liệu.

Nạn nhân của các cuộc tấn công từ chối dịch vụ phân tán thường là các trang mạng hay server tiêu biểu như: web game, cổng thanh toán thẻ tín dụng, ngân hàng và thậm chí là DNS root servers.

Đối tượng tấn công DDoS không chỉ sử dụng máy tính của họ để tấn công mà còn sử dụng máy tính của những người dùng khác. Bằng cách xâm nhập và chiếm quyền kiểm soát máy tính của người dùng một các bất hợp pháp. Tấn công từ chối dịch vụ phấn tán là vi phạm chính sách sử dụng Internet Architecture Board và những người tấn công DDoS sẽ vi phạm luật dân sự [[5]](#_TÀI_LIỆU_THAM_1).

* 1. **MỘT SỐ KỸ THUẬT PHÒNG CHỐNG XÂM NHẬP MẠNG**

Đối với các vấn đề xâm nhập được đề cập, một số biện pháp phòng chống phổ biến được đề xuất gồm tường lửa (firewall), mã hóa dữ liệu, mạng riêng ảo (virtual private network – VPN), …

* + 1. **Tường lửa (Firewall)**

Trong lĩnh vực công nghệ thông tin, tường lửa là kĩ thuật được tích hợp vào hệ thống mạng với mục đích chống lại sự truy cập trái phép, giúp bảo vệ dữ liệu quan trọng. Nó điều khiển truy cập bằng việc cho phép hoặc từ chối các dòng dữ liệu vào/ra. Từ việc kiểm soát được các luồng thông tin, tường lửa sẽ đưa ra quyết định cho phép hay không cho phép đối với chúng. Tường lửa có thể là hệ thống phần cứng, phần mềm hoặc kết hợp cả hai.

Tường lửa có thể hoạt động trên nhiều tầng khác nhau của mạng. Chúng được phân chia thành nhiều loại tùy thuộc vào vị trí, mục đích hoạt động nhưng chúng đều có chung đặc điểm: Chặn bắt dữ liệu, kiểm tra rồi quyết định có cho dữ liệu đi qua hay không, thay đổi thông tin, … Ngoài ra, một số tường lửa phức tạp hơn còn yêu cầu xác thực trước khi cho phép truy cập.

Chức năng của một tường lửa thường gồm:

o Bảo vệ dữ liệu quan trọng và tài nguyên hệ thống

o Giữ vai trò như một chốt an ninh, kiểm soát hoạt động dịch vụ truy cập vào trong hoặc ra ngoài

o Ngăn chặn một số cuộc tấn công mạng như tấn công dò tìm mật khẩu trực tiếp

Được biết đến như một biện pháp phổ biến nhưng tường lửa vần tồn tại nhiều nhược điểm:

o Không thể ngăn chặn các tấn công mà không “đi qua” nó

o Không thể chống lại các cuộc tấn công bằng dữ liệu

o Không thật sự thông minh để hiểu một gói tin là tốt hay xấu để đưa ra lệnh cho phép hoặc không [[6]](#_TÀI_LIỆU_THAM_1)

* + 1. **Mã hóa dữ liệu**

Mã hóa dữ liệu là chuyển đổi kiểu dữ liệu ban đầu thành một kiểu khác mà không thể hiểu được theo cách thông thường. Tác dụng của nó là ngăn chặn việc nghe trộm và chỉnh sửa dữ liệu trên đường truyền. Bên thứ ba có thể lấy được các gói tin đã mã hóa, nhưng không thể đọc được nội dung thông điệp từ các gói tin này. Trong mã hóa dữ liệu, dữ liệu ban đầu được gọi là plaintext, dữ liệu đã được mã hóa là ciphertext.

Quá trình mã hóa từ plaintext sang ciphertext sẽ biến nội dung sang một dạng mới, vì thế sẽ tăng thêm một lớp bảo mật cho dữ liệu. Ứng với mỗi cách mã hóa là một cách giải mã nhất định, bao gồm thuật toán và khóa là bí mật. Giả sử dữ liệu bị đánh cắp thì phải mất một khoảng thời gian khá lâu và một khoảng tài nguyên nhất định đủ để tên trộm có thể giải mã dữ liệu. Thực tế, việc mã hóa dữ liệu không thể hoàn toàn ngăn việc dữ liệu có thể bị đánh cắp nhưng lại là việc vô cùng cần thiết để tăng tính bảo mật cho dữ liệu.

* + 1. **Mạng riêng ảo (Virtual private network – VPN)**

Mạng riêng ảo là một kĩ thuật giúp tạo kết nối an toàn khi tham gia vào mạng công cộng bằng cách tạo một mạng ảo. Nó ngăn chặn những người không được phép nghe lén và cho phép người dùng thực hiện công việc từ xa. Vì vậy, các tập đoàn lớn, các cơ sở giáo dục và cơ quan chính phủ sử dụng công nghệ này để cho phép người dùng từ xa kết nối an toàn đến mạng riêng của cơ quan mình.

Khi kết nối máy tính với VPN, máy tính sẽ hoạt động như thể đang kết nối với mạng cục bộ. Điều này cho phép người dùng truy cập nguồn tài nguyên mạng cục bộ ngay cả khi họ đang ở đầu bên kia của thế giới. Ngoài ra, người dùng cũng có thể sử dụng Internet giống như thể họ đang hiện diện tại vị trí của VPN. Ví dụ, nếu VPN đặt tại Nhật Bản thì khi người dùng kết nối đến một ứng dụng như Netflix, nó sẽ ghi nhận địa chỉ của họ ở Nhật Bản.

VPN là một giải pháp tiện lợi với chi phí thấp để xây dựng một hệ thống mạng riêng mà vẫn cung cấp được nhiều chức năng đến người dùng như truy cập từ xa, duyệt web ẩn danh, truy cập những web bị chặn giới hạn địa lí, … Tuy nhiên, VPN vẫn tồn tại những nhược điểm như gói dữ liệu có nguy cơ bị thất lạc, những người dùng có nguy cơ bị tấn công, …

* 1. **HỆ THỐNG PHÁT HIỆN XÂM NHẬP MẠNG (INSTRUCTION DETECT SYSTEM - IDS)**
     1. **Khái niệm**

Hệ thống phát hiện xâm nhập là một hệ thống phát hiện các xâm nhập trái phép, các cuộc tấn công vào hệ thống mạng và đưa ra cảnh báo. IDS có thể là thiết bị phần cứng hay phần mềm. Mục đích của nó là ngăn ngừa và phát hiện những hành động mang tính phá hoại sự bảo mật của hệ thống hoặc những hành vi như dò tìm, quét các cổng. Ngoài ra, IDS có khả năng phân biệt hành vi tấn công đến từ bên ngoài hay trong nội bộ.

Ngày nay có rất nhiều sản phẩm công nghệ ra đời với mục đích bảo vệ hệ thống mạng của người dùng. Tuy nhiên, chúng ta không nên nhầm lẫn một trong số chúng là một hệ thống phát hiện xâm nhập. Chẳng hạn, tường lửa hay một phần mềm chống vi-rút nào đó không phải là IDS [[8]](#_TÀI_LIỆU_THAM_1).

IDS là một trong những giải pháp hiệu quả và quan trọng trong bảo vệ hệ thống. Nhìn chung, nó sẽ cung cấp các chức năng như sau:

o Theo dõi, giám sát và phát hiện các hành vi bất thường, xâm nhập trái phép đối với hệ thống

o Đưa ra cảnh báo về tình trạng cho hệ thống và nhà quản trị kịp thời

o Xác định vị trí có vấn đề trong hệ thống

o Phân biệt nguồn gốc cuộc tấn công đến từ nội bộ hay bên ngoài

* + 1. **Phân loại**

Tùy theo chức năng và nhiệm vụ, IDS sẽ được chia thành nhiều loại khác nhau như sau [[8]](#_TÀI_LIỆU_THAM_1):

o *NIDS (Network Intrusion Detection Systems):* hệ thống phát hiện xâm nhập mạng sẽ tập hợp gói tin để phân tích sâu bên trong mà không làm thay đổi cấu trúc gói tin. NIDS có thể là phần mềm triển khai trên máy chủ (server) hoặc dạng thiết bị tích hợp. Nó hoạt động tốt trong việc kiểm tra và phát hiện các dạng tấn công trên mạng. Thông thường, NIDS sẽ được bố trí tại những điểm dễ bị tấn công trong hệ thống.

o *HIDS (Host Intrusion Detection Systems):* hệ thống phát hiện xâm nhập host (bất cứ một máy tính nào có kết nối tới một mạng máy tính và có địa chỉ xác định) theo dõi các hoạt động bất thường trên các host riêng biệt. Nó thường được được đặt trên các host quan trọng hay cần được theo dõi.

o *Signature-Based IDS:* là các IDS hoạt động dựa trên chữ ký. Nó giám sát các gói tin trên mạng và so sánh chúng với cơ sở dữ liệu chữ ký. Nhược điểm của các IDS này là có thể không phát hiện ra mối đe dọa mới, khi chữ ký để nhận biết nó chưa được IDS cung cấp hay cập nhật.

o *Misuse-based IDS:* hệ thống phát hiện xâm nhập dựa vào dấu hiệu. Những dấu hiệu đó có thể là thông tin về các kết nối nguy hiểm đã biết trước. Sau đó, nó so sánh thông tin của các gói tin đến với các dấu hiệu này để phát hiện ra các hoạt động đáng ngờ và đưa ra cảnh báo cho hệ thống. IDS này hiệu quả trong việc phát hiện các tấn công đã biết với tỷ lệ cảnh báo sai thấp. Tuy nhiên, việc chỉ phát hiện được những cuộc tấn công đã biết lại trở thành nhược điểm lớn của nó.

o *Anomaly-Based IDS:* IDS này sẽ phát hiện mối đe dọa dựa trên sự bất thường. Quá trình phát hiện bất thường được tiến hành qua hai giai đoạn: giai đoạn huấn luyện (pha huấn luyện) và gia đoạn phát hiện (pha phát hiện). Tại pha huấn luyện, nó xây dựng một hồ sơ về các hoạt động bình thường (thông số chuẩn). Sau đó, tại pha phát hiện, nó tiến hành so khớp gói tin với hồ sơ đã tạo ở pha huấn luyện để xác định dấu hiệu bất thường. Nó khá hiệu quả trong việc phát hiện các mối nguy hiểm không được biết trước.

o *Passive IDS:* IDS loại này rất thụ động, nó chỉ phát hiện và gửi cảnh báo đến nhà quản trị. Những hành động sau đó sẽ phụ thuộc vào người quản trị.

o *Reactive IDS:* IDS loại này không chỉ thực hiện các chức năng như IDS Passive mà còn thực hiện những hành động được thiết lập sẵn để phản ứng lại các mối đe dọa tức thì.

* + 1. **Ưu nhược điểm**

**Ưu điểm**

o Là một lựa chọn thích hợp để thu thập số liệu, giúp kiểm tra các sự cố xảy ra đối với hệ thống mạng với những bằng chứng thuyết phục nhất

o Là công cụ để chủ động giám sát và bảo vệ mạng của bạn khỏi hoạt động độc hại

o Đem đến cái nhìn bao quát và toàn diện về toàn bộ hệ thống mạng

**Nhược điểm**

o Dễ đưa ra các báo động sai nếu không được cấu hình hợp lý

o Tốn nhiều chi phí để triển khai, vận hành, phát triển và mở rộng

Mặc dù được xem là một biện pháp hữu hiệu nhưng IDS vẫn có khả năng bị tấn công bởi tấn công từ chối dịch vụ (Distributed Denial of Service - DoS) hay các tấn công đánh lừa nhằm khiến IDS đưa ra các cảnh báo sai.

## 

## **CHƯƠNG 2: TỔNG QUAN VỀ MACHINE LEARNING**

### **2.1 KHÁI QUÁT VỀ HỌC MÁY (MACHINE LEARNING)**

#### **2.1.1 Học máy là gì?**

Học máy (machine learning) là một lĩnh vực con của trí tuệ nhân tạo (artificial intelligence) sử dụng các thuật toán cho phép máy tính có thể học từ dữ liệu để thực hiện các công việc thay vì được lập trình một cách rõ ràng.

#### **2.1.2 Phân nhóm thuật toán**

Về cơ bản, các thuật toán học máy (machine learning) có thể phân thành hai nhóm [[9]](#_TÀI_LIỆU_THAM_1):

* Nhóm các thuật toán học máy theo phong cách học tập, hoặc
* Nhóm các thuật toán học máy dựa trên sự tương đồng về hình thức hoặc chức năng

##### *2.1.2.1 Thuật toán phân theo phong cách học tập*

*2.1.2.1.1 Học có giám sát (supervised learning)*

Shape, square

Description automatically generated

**Hình 2. 1 Học có giám sát (supervised learning)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.1 minh họa về thuật toán học có giám sát. Về cơ bản, trong thuật toán được giám sát này, dữ liệu đầu vào được gọi là dữ liệu huấn luyện và có nhãn. Dữ liệu đầu vào của quá trình bao gồm cả vector đầu vào chứa các thuộc tính của dữ liệu lẫn giá trị đầu ra mục tiêu (gọi là nhãn của dữ liệu). Mô hình học có giám sát cho phép dự đoán đầu ra của một dữ liệu mới dựa trên các cặp (đầu vào, đầu ra) đã biết trước thu được từ bộ dữ liệu huấn luyện. Bộ dữ liệu huấn luyện bao gồm các cặp (dữ liệu, nhãn). Trong đó, một mô hình được xây dựng thông qua một quá trình huấn luyện. Quá trình huấn luyện tiếp tục cho đến khi mô hình đạt được mức độ mong muốn.

• Các bài toán phổ biến là phân loại và hồi quy.

• Các thuật toán phổ biến bao gồm hồi quy logistic và mạng thần kinh

*2.1.2.1.2 Học không giám sát (unsupervised learning)*

A picture containing shape

Description automatically generated

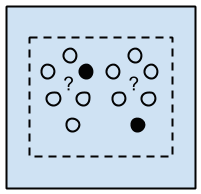
**Hình 2. 2 Học không giám sát (unsupervised learning) [9]**

Hình 2.2 mô tả về thuật toán học không giám sát. Học không giám sát là một lớp mô hình học sử dụng một thuật toán để mô tả, trích xuất ra các mối quan hệ tiềm ẩn trong dữ liệu. Khác với học có giám sát, học không giám sát chỉ thực thi trên dữ liệu đầu vào không cần các thuộc tính nhãn và không có kết quả đã biết.

Chúng ta phải xây dựng một mô hình bằng cách suy ra các cấu trúc tiềm ẩn có trong dữ liệu đầu vào. Điều này có thể là để trích xuất ra các quy tắc chung. Nó có thể thông qua một quá trình toán học để giảm sự dư thừa.

* Các bài toán phổ biến là phân cụm, giảm kích thước và học quy tắc kết hợp.
* Các thuật toán phổ biến bao gồm thuật toán Apriori và k-Means.

*2.1.2.1.3 Học bán giám sát (semi-supervised learning)*

******

**Hình 2. 3 Học bán giám sát (semi-supervised learning)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.3 mô tả thuật toán học bán giám sát. Dữ liệu đầu vào là một hỗn hợp của các đối tượng được dán nhãn và không nhãn.

Mục tiêu của mô hình học bán giám sát là tận dụng hiệu quả toàn bộ dữ liệu có được, không chỉ dữ liệu được gán nhãn mà cả dữ liệu chưa được gán nhãn. Thông thường điều này làm tăng tính chính xác của mô hình được xây dựng được. Vì vậy, mô hình học bán giám sát thường kết hợp cá các thuật toán không giám sát và có giám sát. Các mô hình học không giám sát như phân cụm hoặc xấp xỉ phân phối được sử dụng để tìm ra các mẫu và nhóm dữ liệu, và các nhóm dữ liệu này sau đó sẽ được sử dụng để gán nhãn dựa trên các thuật toán học có giám sát.

Cho trước một bài toán dự đoán, mô hình phải học các cấu trúc để tổ chức dữ liệu cũng như đưa ra dự đoán.

* Các bài toán phổ biến là phân loại và hồi quy.
* Các thuật toán phổ biến là co-training, self-training. Các mô hình này thường đưa ra các giả định về cách mô hình hóa dữ liệu chưa được gắn nhãn.

Học bán giám sát có hiệu quả trong các bài toán thực tế khi tài nguyên sử dụng để gán nhãn dữ liệu tấy tốn kém. Chẳng hạn như trong các bài toán thị giác máy tính (dữ liệu ảnh), xử lý ngôn ngữ tự nhiên (dữ liệu văn bản), nhận dạng giọng nói (dữ liệu âm thanh)

##### *2.1.2.2 Các thuật toán được nhóm theo sự tương đồng*

Các thuật toán học máy thường được nhóm theo một sự tương đồng về chức năng của chúng.

Ví dụ, các phương pháp dựa trên cây và các phương pháp lấy cảm hứng từ mạng thần kinh.

Đây là một nhóm phương pháp hữu ích, nhưng nó không hoàn hảo. Có một số thuật toán có thể dễ dàng phù hợp với nhiều bài toán, như hồi quy và phân loại.

*2.1.2.2.1 Thuật toán hồi quy (regression algorithms)*

Ảnh có chứa văn bản, thiết bị đo

Mô tả được tạo tự động

**Hình 2. 4 Thuật toán hồi quy (regression algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.4 mô tả thuật toán hồi quy. Thuật toán hồi quy liên quan đến việc mô hình hóa mối quan hệ giữa các biến. Những phương pháp này vốn có nguồn gốc từ thống kê nhưng dần được sử dụng phổ biến trong học máy.

Các thuật toán hồi quy phổ biến nhất trong học máy là:

* Ordinary Least Squares Regression (OLSR)
* Linear Regression
* Logistic Regression
* Stepwise Regression
* Multivariate Adaptive Regression Splines (MARS)
* Locally Estimated Scatterplot Smoothing (LOESS)

*2.1.2.2.2 Thuật toán instance-based*

A pair of scissors

Description automatically generated with low confidence

**Hình 2. 5 Thuật toán instance-based (instance-based algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.5 mô tả thuật toán instance-based. Mô hình này là một bài toán quyết định với các đối tượng (instance) trong dữ liệu huấn luyện.

Phương pháp như vậy xây dựng một cơ sở dữ liệu của các đối tượng dữ liệu (instance). Nó cần so sánh dữ liệu mới với cơ sở dữ liệu. Để so sánh, các độ đo tương tự được dùng để tìm ra kết quả phù hợp nhất và đưa ra dự đoán. Vì lý do này, các phương này cũng được gọi là phương pháp thắng-lấy-tất cả và học tập dựa trên bộ nhớ. Trọng tâm được đặt vào đại diện của các instance được lưu trữ. Vì vậy, các độ đo tương tự được sử dụng giữa các đối tượng (instance).

Các thuật toán instance-based phổ biến nhất trong học máy là:

* k-Nearest Neighbor (kNN)
* Learning Vector Quantization (LVQ)
* Self-Organizing Map (SOM)
* Locally Weighted Learning (LWL)

*2.1.2.2.3 Thuật toán regularization*

A picture containing sport, athletic game

Description automatically generated

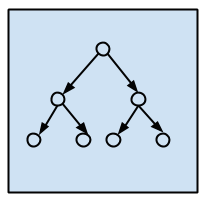
**Hình 2. 6 Thuật toán regularization (regularization algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.6 mô tả thuật toán regularization. Đây là một kỹ thuật mở rộng được thực hiện cho một phương pháp khác. Đó là xử phạt các mô hình dựa trên sự phức tạp của chúng. Việc ủng hộ các mô hình đơn giản hơn thường tốt hơn trong việc khái quát hóa.

Các thuật toán regularization phổ biến nhất trong học máy là:

* Ridge Regression
* Least Absolute Shrinkage and Selection Operator (LASSO)
* Elastic Net
* Least-Angle Regression (LARS)

*2.1.2.2.4 Thuật toán cây quyết định*



**Hình 2. 7 Thuật toán cây quyết định (decision tree algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

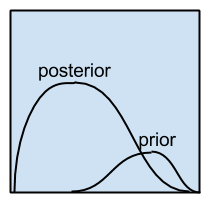
Hình 2.7 mô tả thuật toán cây quyết định. Phương pháp cây quyết định xây dựng một mô hình của các quyết định. Điều đó được thực hiện dựa trên giá trị thực tế của các thuộc tính trong dữ liệu.

Cây quyết định có thể được sử dụng cho bài toán phân loại và hồi quy. Cây quyết định thường nhanh và chính xác và là một trong những thuật toán được yêu thích trong học máy.

Các thuật toán cây quyết định phổ biến nhất trong học máy là:

* Classification and Regression Tree (CART)
* Iterative Dichotomiser 3 (ID3)
* C4.5 and C5.0 (different versions of a powerful approach)
* Chi-squared Automatic Interaction Detection (CHAID)
* Decision Stump
* Conditional Decision Trees

*2.1.2.2.5 Thuật toán Bayesian*



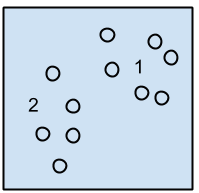
**Hình 2. 8 Thuật toán Bayesian (Bayesian Algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.8 mô tả thuật toán Bayesian. Những phương pháp này là những phương pháp áp dụng định lý Bayes (có thể cho phân loại và cả hồi quy).

Các thuật toán Bayesian phổ biến nhất trong học máy là:

* Naive Bayes
* Gaussian Naive Bayes
* Multinomial Naive Bayes
* Averaged One-Dependence Estimators (AODE)
* Bayesian Belief Network (BBN)
* Bayesian Network (BN)

*2.1.2.2.6 Thuật toán clustering*



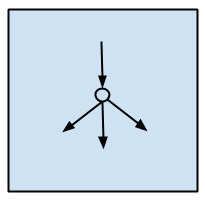
**Hình 2. 9 Thuật toán gom cụm (clustering algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.9 mô tả thuật toán gom cụm. Các phương pháp gom cụm có thể được phân theo nhiều nhóm. Hai nhóm phổ biến là nhóm các phương pháp dựa trên centroid và phân cấp. Tất cả các phương pháp đều liên quan đến việc sử dụng các cấu trúc vốn có trong dữ liệu.

Các thuật toán phân cụm phổ biến nhất trong học máy là:

* k-Means
* k-Medians
* Expectation Maximisation (EM)
* Hierarchical Clustering

*2.1.2.2.7 Thuật toán mạng nơ-ron nhân tạo*

****

**Hình 2. 10 Thuật toán mạng neural nhân tạo (artifical neural network algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.10 mô tả thuật toán mạng neural nhân tạo. Đây là một mô hình Toán học hay mô hình tính toán được xây dựng dựa trên các mạng neural sinh học. Nó gồm có một nhóm neural nhân tạo nối với nhau, và xử lý thông tin bằng cách truyền theo các kết nối và giá trị mới tại các nút.

Các thuật toán [mạng thần kinh nhân tạo](https://data-flair.training/blogs/artificial-neural-network/) phổ biến nhất là:

* Perceptron
* Back-Propagation
* Hopfield Network
* Radial Basis Function Network (RBFN)

*2.1.2.2.8 Thuật toán deep learning*

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Hình 2. 11 Thuật toán deep learning (deep learning algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.11 mô tả thuật toán học sâu (deep learning). Phương pháp học sâu là một bản cập nhật hiện đại cho mạng nơ ron nhân tạo. Chúng quan tâm đến việc xây dựng các mạng lưới thần kinh lớn hơn và phức tạp hơn nhiều.

Các thuật toán deep learning phổ biến nhất là:

* Deep Boltzmann Machine (DBM)
* Deep Belief Networks (DBN)
* Convolutional Neural Network (CNN)
* Stacked Auto-Encoders

*2.1.2.2.9 Thuật toán giảm chiều*

Ảnh có chứa quảng trường

Mô tả được tạo tự động

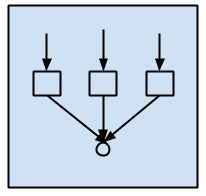
**Hình 2. 12 Thuật toán giảm chiều (dimensionality reduction algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.12 mô tả thuật toán giảm chiều. Giống như các phương pháp phân cụm, giảm kích thước tìm kiếm cấu trúc vốn có trong dữ liệu.

Nói chung, nó có thể hữu ích để trực quan hóa dữ liệu đa chiều. Ngoài ra, chúng ta có thể sử dụng nó trong một phương pháp học có giám sát. Có nhiều phương pháp giảm chiều như bên dưới:

* Principal Component Analysis (PCA)
* Principal Component Regression (PCR)
* Partial Least Squares Regression (PLSR)
* Sammon Mapping
* Multidimensional Scaling (MDS)
* Projection Pursuit
* Linear Discriminant Analysis (LDA)
* Mixture Discriminant Analysis (MDA)
* Quadratic Discriminant Analysis (QDA)
* Flexible Discriminant Analysis (FDA)

*2.1.2.2.10 Thuật toán tổng hợp*



**Hình 2. 13 Thuật toán tổng hợp (ensemble algorithms)** [**[9]**](#_TÀI_LIỆU_THAM_1)

Hình 2.13 mô tả thuật toán tổng hợp. Về cơ bản, các phương pháp này là các mô hình bao gồm các mô hình yếu hơn. Ngoài ra, khi chúng đã được huấn luyện, dự đoán của chúng (các mô hình yếu) được kết hợp theo một cách nào đó để đưa ra dự đoán cuối cùng. Đây là một lớp kỹ thuật rất mạnh và rất phổ biến. Các thuật toán ensemble phổ biến là:

* Boosting
* Bootstrapped Aggregation (Bagging)
* Stacked Generalization (blending)
* Gradient Boosting Machines (GBM)
* Gradient Boosted Regression Trees (GBRT)
* Random Forest

### **2.2 MỘT SỐ THUẬT TOÁN CỔ ĐIỂN**

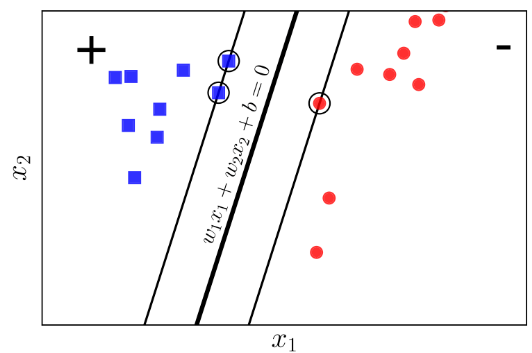
#### **2.2.1 Thuật toán Support Vector Machine (SVM)**

Support vetor machine (SVM) là thuật toán học máy có giám sát được sử dụng

Support vetor machine (SVM) là thuật toán học máy có giám sát được sử dụng cho bài toán phân lớp (classification) hoặc hồi quy (regression). Trong đó bài toán phân lớp được sử dụng nhiều nhất. Mục tiêu của nó là tìm ra hyperplane phân chia các lớp (class).

Giả sử rằng các cặp dữ liệu của training set là với vector thể hiện đầu vào của một điểm dữ liệu và là nhãn của điểm dữ liệu đó, là số chiều của dữ liệu và là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi = 1 (class 1) hoặc = -1 (class 2)

Hình 2.14 dưới đây mô tả ví dụ về trường hợp không gian hai chiều:



**Hình 2. 14 Phân tích bài toán SVM** [**[10]**](#_TÀI_LIỆU_THAM_1)

Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt hyperplane có dạng , tức là mặt phân chia giữa hai classes 1 và -1 (hình 2.14). Hơn nữa, class 1 nằm về phía dương, class -1 nằm về phía âm của mặt phân chia. Nếu ngược lại, ta chỉ cần đổi dấu của và .

Với cặp dữ liệu bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

|  |  |
| --- | --- |
|  | (2.1) |

Với mặt phần chia như trên, margin được tính là khoảng cách gần nhất từ một điểm tới mặt đó (bất kể điểm nào trong hai classes):

|  |  |
| --- | --- |
|  | (2.2) |

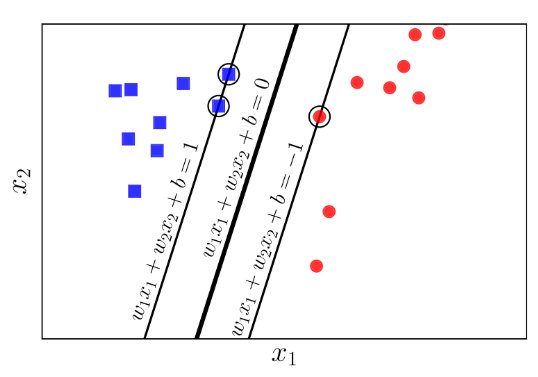
Bài toán tối ưu trong SVM chính là bài toán tìm  và sao cho margin này đạt giá trị lớn nhất:

|  |  |
| --- | --- |
|  | (2.3) |

Nếu ta thay vector hệ số bởi và bởi trong đó là một hẳng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi. Dựa trên tính chất này, ta có thể giả sử:

|  |  |
| --- | --- |
|  | (2.4) |

Với những điểm nằm gần mặt phân chia nhất như hình 2.15 dưới đây:



**Hình 2. 15 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn** [**[10]**](#_TÀI_LIỆU_THAM_1)

Như vậy, với mọi ta có:

|  |  |
| --- | --- |
|  | (2.5) |

Vậy bài toán tối ưu có thể đưa về bài toán tối ưu có ràng buộc sau đây:

|  |  |
| --- | --- |
|  | (2.6) |

Xác định class cho một điểm dữ liệu mới bằng cách: Sau khi tìm được mặt phân cách = 0, class của bất kỳ một điểm nào sẽ được xác định đơn giản bằng cách:

|  |  |
| --- | --- |
|  | (2.7) |

#### **2.2.2 Thuật toán naïve bayes**

Naive Bayes là một thuật toán phân lớp được mô hình hoá dựa trên định lý Bayes trong xác suất thống kê [[27]](#_Danh_mục_tham):

|  |  |
| --- | --- |
|  | (2.8) |

Trong đó:

: xác suất của mục tiêu y với điều kiện có đặc trưng X

: xác suất của X khi biết mục tiêu y

X là vector các đặc trưng, có thể viết dưới dạng:

Khi đó, đẳng thức Bayes trở thành:

|  |  |
| --- | --- |
| = | (2.9) |

Khi đó, việc tìm y để P(y|X) đạt cực đại trở thành:

|  |  |
| --- | --- |
|  | (2.10) |

Một số kiểu mô hình Naïve Bayes [[28]](#_Danh_mục_tham)

*+ Multinomial Naive Bayes*

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài chính là số từ trong từ điển. Giá trị của thành phần thứ là số lần từ thứ xuất hiện trong văn bản đó.

Khi đó, () tỉ lệ với tần suất từ thứ xuất hiện trong các văn bản của class . Giá trị này có thể được tính bằng cách:

|  |  |
| --- | --- |
|  | (2.11) |

Trong đó:

: Tổng số lần từ thứ i xuất hiện trong các văn bản của class c

: Tổng số từ xuất hiện trong class c

*+ Bernoulli Naive Bayes*

Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị nhị phân 0 hoặc 1.

*+ Gaussian Naive Bayes*

Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.

Với mỗi chiều dữ liệu và một class , tuân theo một phân phối có chuẩn kỳ vọng và phương sai

|  |  |
| --- | --- |
|  | (2.12) |

Trong đó, bộ tham số được xác định bằng Maximum Likelihood:

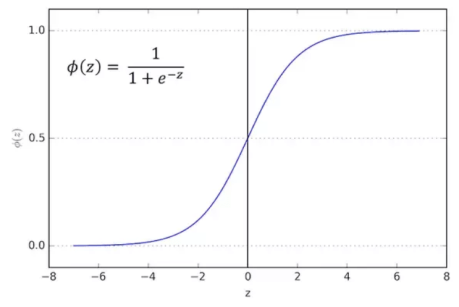
|  |  |
| --- | --- |
|  | (2.13) |

#### **2.2.3 Thuật toán Logistic Regression**

Thuật toán logistic regression là một thuật toán dựa vào thống kê đánh giá giữa các giá trị đầu vào (một hoặc một vài feature) và kết quả , là một giá trị đại diện cho một sự kiện có xảy ra hay không. Trong hồi qui tuyến tính, hàm hồi qui được dùng để dự báo mục tiêu liên tục [[29]](#_Danh_mục_tham):

|  |  |
| --- | --- |
|  | (2.14) |

Vì giá trị có thể vượt ngoài khoảng [0,1] nên người ta dùng hàm sigmoid có tác dụng chiếu giá trị dự báo lên không gian xác suất nằm trong khoảng [0,1] và đồng thời tạo ra tính phi tuyến cho phương trình hồi qui nhằm giúp nó có đường biên phân chia giữa hai nhóm tốt hơn.



**Hình 2. 18 Đồ thị hàm sigmoid** [**[30]**](#_Danh_mục_tham)

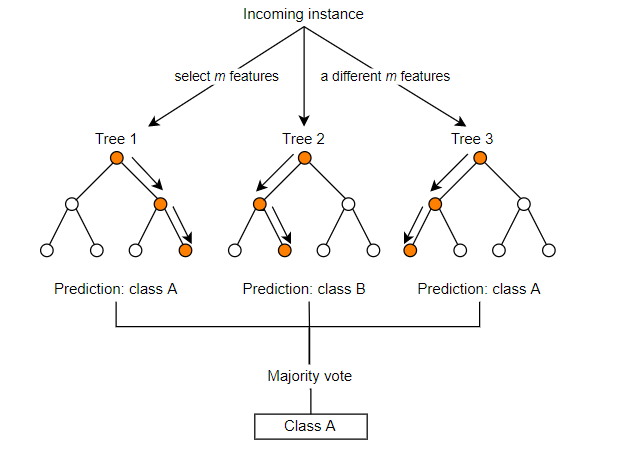
Hình 2.18 biểu diễn đồ thị hàm sigmoid có hình dạng là một đường cong chữ s và đơn điệu tăng. Xét khoảng giá trị trong hàm sigmoid, ta có:

|  |  |
| --- | --- |
|  |  |
|  |  |

#### **2.2.4 Thuật toán Random Forest**

Random forests là thuật toán học có giám sát. Nó có thể được sử dụng cho cả phân lớp và hồi quy. Thuật toán Random Forest mình sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree. Tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.

Ở bước huấn luyện, nhiều cây quyết định được xây dựng. Các cây quyết định này khác nhau. Sau đó ở bước dự đoán, với một dữ liệu mới, thì ở mỗi cây quyết định đều đưa ra dự đoán. Cuối cùng, kết quả được tổng hợp từ dự đoán của các cây quyết định.



**Hình 2. 19 Mô hình RF dang phân loại đơn giản với Ntree = 3** [**[31]**](#_TÀI_LIỆU_THAM_1)

*Xây dựng thuật toán random forest*

Giả sử bộ dữ liệu có n đối tượng (object) và mỗi đối tượng có d thuộc tính (feature).

Quá trình xây dựng từng cây quyết định như sau:

* Lấy mẫu ngẫu nhiên có hoàn lại (random sampling with replacement) n đối tượng từ bộ dữ liệu với kĩ thuật bootstrapping để tạo ra môt mẫu boostrap (boostrap sample). Khi dùng kĩ thuật này thì tập n dữ liệu mới có thể có những dữ liệu bị trùng nhau trong mẫu boostrap.
* Do quá trính xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest thường khác nhau.
* Thuật toán Random Forest sẽ bao gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau.
* Cuối cùng, kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ dự đoán của các cây quyết định.

Khi dùng thuật toán Random Forest, các siêu tham số như: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây. Ngoài ra, vẫn có các thuộc tính của thuật toán Decision Tree để xây dựng cây như độ sâu tối đa, số phần tử tối thiểu trong một node để có thể tách.

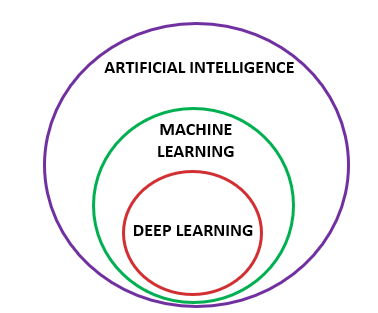
## 

## **CHƯƠNG 3: TỔNG QUAN VỀ HỌC SÂU (DEEP LEARNING)**

### **3.1 KHÁI QUÁT VỀ HỌC SÂU**

#### **3.1.1 Học sâu là gì?**

Học sâu (deep learning) được bắt nguồn từ thuật toán neural network vốn xuất phát chỉ là một ngành nhỏ của học máy. Học sâu là một lĩnh vực của học máy, là một trường con của lĩnh vực trí tuệ nhân tạo (AI) dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều biến đổi phi tuyến. Nói một cách đơn giản là chúng bắt chước hoạt động của bộ não con người trong việc xử lí dữ liệu và tạo ra các mẫu để sử dụng cho việc ra quyết định. Hình 3.1 mô tả mối liên quan của học sâu, học máy, và trí tuệ nhân tạo.



**Hình 3. 1 Mối liên quan của Deep Learning, Machine Learning, Artifical Intelligence** [**[11]**](#_TÀI_LIỆU_THAM_1)

Một số ứng dụng khác của học sâu

* Hệ thống gợi ý trên các nền tảng
* Nhận diện hình ảnh
* Phát hiện các loại bệnh hiếm gặp
* Ứng dụng xe tự động
* Tính năng dịch tự động
* Trợ lý ảo

Khi nào nên sử dụng học sâu

* Khi có rất nhiều dữ liệu và phần lớn dữ liệu không có cấu trúc.
* Các thuật toán học sâu có thể lấy dữ liệu lộn xộn và không có nhãn rộng rãi chẳng hạn như video, hình ảnh, bản ghi âm thanh, văn bản, ... và áp đặt đủ thứ tự cho dữ liệu đó để đưa ra dự đoán hữu ích. Thành phần cơ bản của học sâu là neural networks.
  1. **KHÁI QUÁT VỀ MẠNG NƠ-RON NHÂN TẠO**
     1. **Khái niệm**

**Mạng nơ-ron nhân tạo (artificial neural network - ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các nơ-ron được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).**

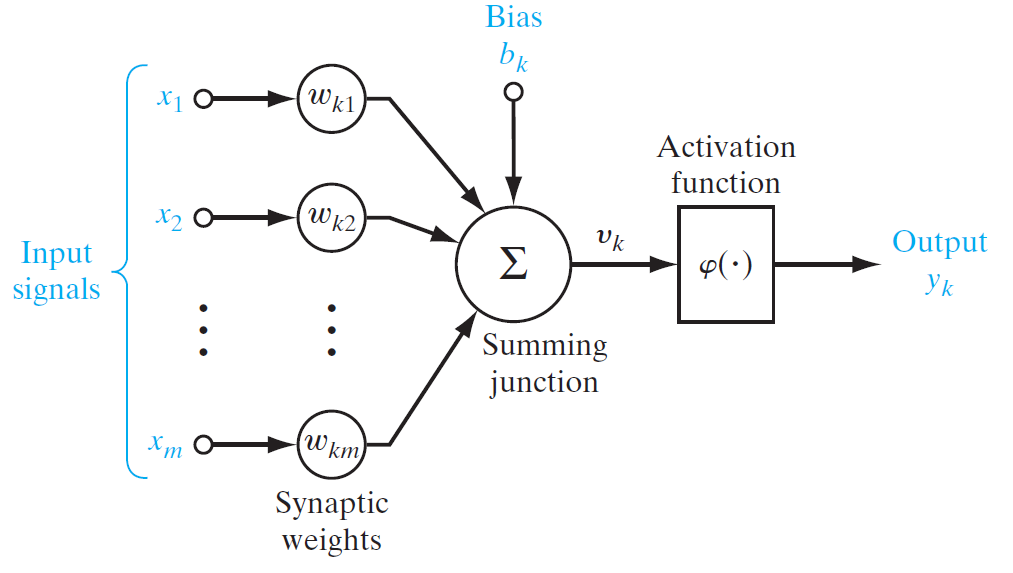
**Một mạng nơ ron sinh học được minh họa trong hình 3.2:**

Ảnh có chứa mũi tên

Mô tả được tạo tự động

**Hình 3. 2 Cấu trúc một mạng nơ-ron sinh học** [**[12]**](#_TÀI_LIỆU_THAM_1)

Tương tự, ta có các nơ-ron nhân tạo hay Perceptron là đơn vị cấu thành mạng nơ-ron nhân tạo. Nó được tạo nên từ một số lượng lớn các phần tử (nơ-ron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Để được coi là một ANN thì nó phải chứa cấu trúc đồ thị có hướng, có nhãn trong đó mỗi nút trong đồ thị thực hiện một số tính toán đơn giản. Một mạng nơ-ron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu, ...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất, học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơ-ron. Mô hình một nơ-ron nhân tạo được minh hoạ trong hình 3.3:



**Hình 3. 3 Mô hình của một nơ-ron nhân tạo được gán nhãn k** [**[13]**](#_TÀI_LIỆU_THAM_1)

* + 1. **Các thành phần cơ bản**

*+ Tập các đầu vào (inputs)*

Là các tín hiệu vào (input signals) của nơ-ron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.

*+ Tập các liên kết*

Mỗi liên kết được thể hiện bởi một trọng số liên kết – synaptic weight. Trọng số liên kết giữa tín hiệu vào thứ *k* với nơ-ron *j* thường được kí hiệu là . Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.

*+ Bộ tổng (summing function)*

Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.

*+ Ngưỡng (còn gọi là một độ lệch - bias)*

Ngưỡng này thường được đưa vào như một thành phần của hàm truyền (transfer function).

*+ Hàm truyền (transfer function)*

Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơ-ron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng.

*+ Đầu ra (output)*

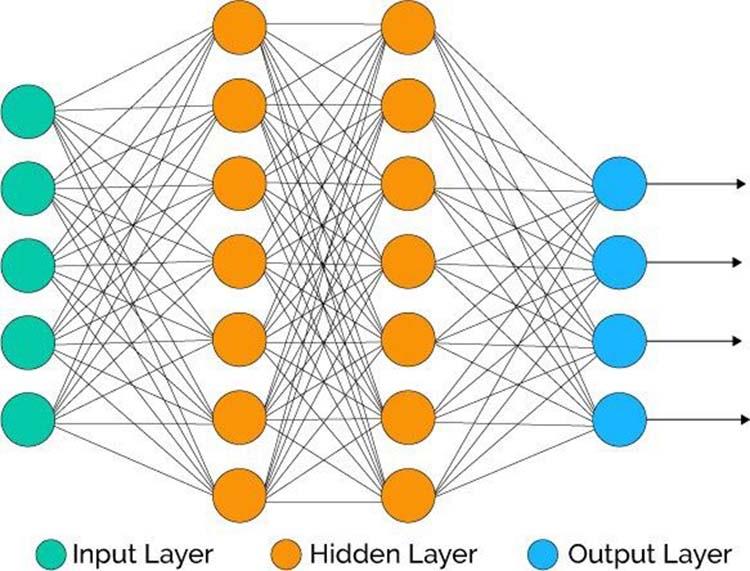
Là tín hiệu đầu ra của một nơ-ron, với mỗi nơ-ron sẽ có tối đa là một đầu ra.

* + 1. **Một số kiến trúc mạng nơ-ron nhân tạo**

Mạng nơ-ron là sự kết hợp của những tầng perceptron hay còn gọi là perceptron đa tầng. Mỗi một mạng nơ-ron thường bao gồm 3 kiểu tầng là:

* Tầng vào (input layer): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
* Tầng ra (output layer): Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.
* Tầng ẩn (hidden layer): Tầng này nằm giữa tầng vào và tầng ra nó thể hiện cho quá trình suy luận logic của mạng.

Lưu ý: Mỗi một mạng nơ-ron chỉ có duy nhất một tầng vào và một tầng ra nhưng lại có thể có rất nhiều tầng ẩn.



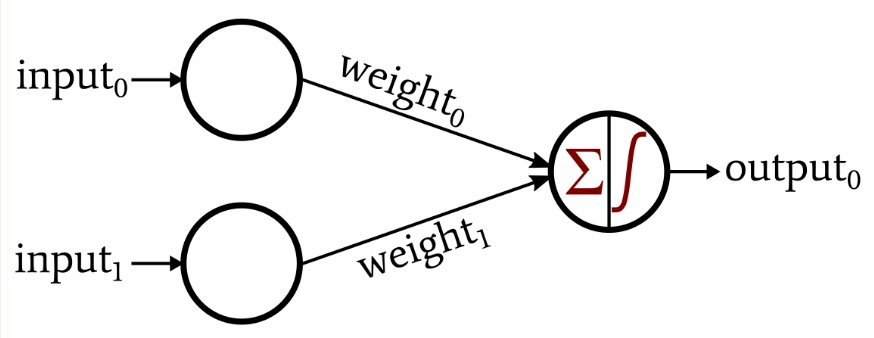
**Hình 3. 4 Kiến trúc mạng Neural Network** [**[14]**](#_TÀI_LIỆU_THAM_1)

Hình 3.4 mô tả kiến trúc mạng neural network. Các nơ-ron kết nối với nhau theo nhiều cách khác nhau nên tạo ra các kiến trúc mạng khác nhau. Sự khác nhau về kiến trúc và cách kết nối tạo ra khả năng ứng dụng khác nhau của các mạng. Ta sẽ tìm hiểu về một số kiến trúc mạng phổ biến.

1. *Perceptron*

Đây là mô hình nơ-ron cơ bản nhất. Các nơ-ron Perceptron sử dụng hàm kích hoạt là hàm ngưỡng – sigmoid. Các nơ-ron Perceptron nhận các đầu vào và thực hiện tính toán với trọng số để tạo ra đầu ra cho các bài toán phân loại đơn giản.

Hình 3.5 minh họa kiến trúc của một Perceptron:



**Hình 3. 5 Perceptron neural** [**[15]**](#_TÀI_LIỆU_THAM_1)

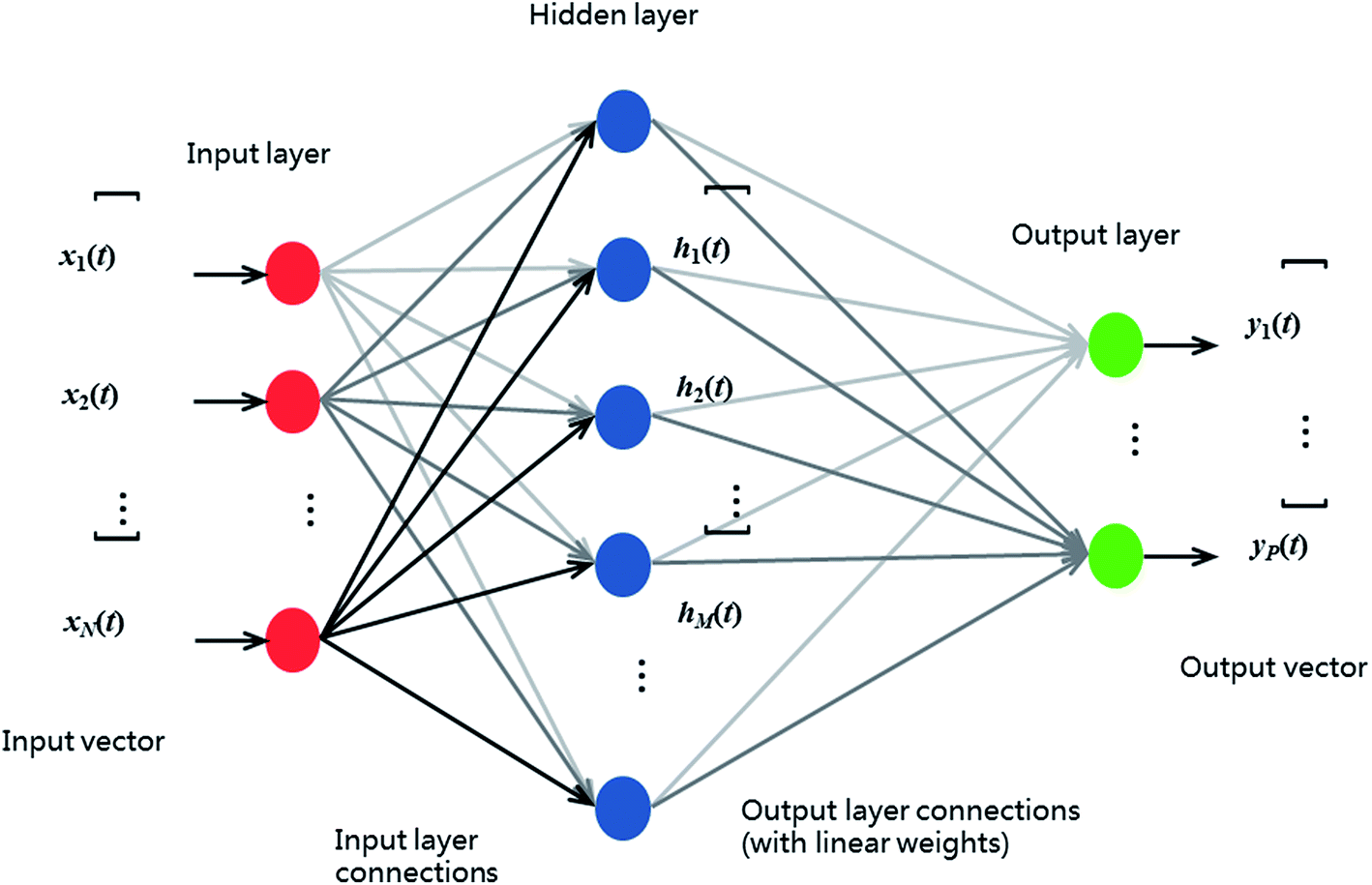
1. *Mạng nhiều lớp lan truyền thẳng (multil-layer feed-forward network)*

Mạng nhiều lớp lan truyền thẳng là mạng nơ-ron truyền thống có nhiều lớp ẩn và các kết nối theo một hướng từ đầu vào đến đầu ra. Các nơ-ron trong mạng lan truyền thẳng sử dụng hàm kích hoạt phi tuyến tính (tanh, ReLU, ... sẽ trình bày chi tiết ở các chương sau) thay vì hàm ngưỡng.

1. *Mạng nơ-ron hàm cơ sở xuyên tâm (radial basis function (RBF) neural network)*

Mạng nơ-ron hàm cơ sở xuyên tâm là trường hợp đặc biệt của mạng nơ-ron nhiều lớp cho bài toán phân loại phi tuyến tính. Mạng nơ-ron hàm cơ sở xuyên tâm chỉ sử dụng một lớp ẩn và sử dụng hàm kích hoạt RBF. Hình 3.6 mô tả kiến trúc của mạng RBF với một tầng ẩn

+ Mô hình mạng RBF:



**Hình 3. 6 Cấu trúc mạng nơ-ron hàm cơ sở xuyên tâm** [**[16]**](#_TÀI_LIỆU_THAM_1)

* 1. **MỘT SỐ THUẬT TOÁN VỀ HỌC SÂU**
     1. **Mạng neural hồi quy – Recurrent Neural Network (RNN)**
        1. *Khái niệm*

Mạng thần kinh hồi quy (hay còn gọi là mạng thần kinh/nơ-ron tái phát, recurrent neural network, viết tắt RNN) là một lớp của mạng thần kinh nhân tạo, với các nút kết nối thành đồ thị có hướng dọc theo một trình tự thời gian. Điều này cho phép mạng thể hiện hành vi động tạm thời. Có nguồn gốc từ mạng thần kinh truyền thẳng, RNN có thể dùng trạng thái trong (bộ nhớ) để xử lý các chuỗi đầu vào có độ dài thay đổi. Điều này làm cho RNN có thể áp dụng cho các tác vụ như xử lý ngôn ngữ tự nhiên hay nhận dạng tiếng nói.

Thông thường, người ta sử dụng thuật ngữ "mạng thần kinh hồi quy" không có tính hệ thống (có phần bừa bãi) nhằm để chỉ hai lớp mạng rộng với một cấu trúc chung giống nhau, cái đầu tiên là lớp mạng đáp ứng xung hữu hạn (finite impulse response) và cái thứ hai là lớp mạng đáp ứng xung vô hạn (infinite impulse response). Cả hai lớp mạng đều thể hiện hệ thống động lực theo thời gian. Mạng hồi quy xung hữu hạn là một đồ thị có hướng không có chu trình (directed acyclic graph) có thể bị mở ra và thay thế bằng một mạng thần kinh truyền thẳng chặt chẽ hơn, trong khi mạng hồi quy xung vô hạn là một đồ thị có hướng mà không thể mở ra (unrolled).

Cả hai mạng hồi quy xung hữu hạn và vô hạn có thể chứa các trạng thái lưu trữ bổ sung, và bộ nhớ có thể được kiểm soát trực tiếp bởi mạng thần kinh. Bộ nhớ cũng có thể được thay thế bằng một mạng hoặc đồ thị khác, nếu kết hợp với thời gian trễ hoặc có vòng lặp phản hồi. Các trạng thái được kiểm soát như vậy được gọi là trạng thái cổng (gated state) hoặc bộ nhớ cổng (gated memory) và là một phần của mạng bộ nhớ dài-ngắn hạn (LSTM) và bộ nhớ định kỳ được kiểm soát. Đây còn được gọi là "mạng thần kinh phản hồi" (feedback neural network, FNN) [[17]](#_TÀI_LIỆU_THAM_1).

* + - 1. *Phân loại bài toán RNN*

Các mô hình RNN hầu hết được sử dụng trong các lĩnh vực xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói. RNN được phân thành các loại như sau:

**Bảng 3. 1 Phân loại chi tiết các bài toán RNN** [**[18]**](#_TÀI_LIỆU_THAM_1)

|  |  |  |
| --- | --- | --- |
| Các loại RNN | Hình minh họa cụ thể | Ví dụ |
| Một – một  (One-to-one) | Bài toán có 1 input và 1 output | Mạng nơ-ron truyền thống kiến trúc 1-1 |
| Một – nhiều  (One-to-many)  , | Bài toán có 1 input và nhiều output | Bài toán caption cho ảnh, đầu vào là một ảnh nhưng đầu ra là mô tả nội dung của ảnh, dưới dạng một câu |
| Nhiều – một  (Many-to-one)  , | Bài toán có nhiều input và 1 output | Bài toán phân loại hành động trong video, đầu vào là các hình ảnh được tách từ video , đầu ra là hành động trong video |
| Nhiều – nhiều  (Many-to-one) | Bài toán có nhiều input và nhiều output với input = output | Nhận dạng thực thể có tên (named entity recognition) |
| Nhiều – nhiều  (Many-to-one) | Bài toán có nhiều input và nhiều output với input output | Bài toán dịch ngôn ngữ khi đầu vào là một câu gồm: “I love VietNam” và đầu ra cũng là một câu gồm nhiều chữ “Tôi yêu Việt Nam” |

* + - 1. *Mô hình*

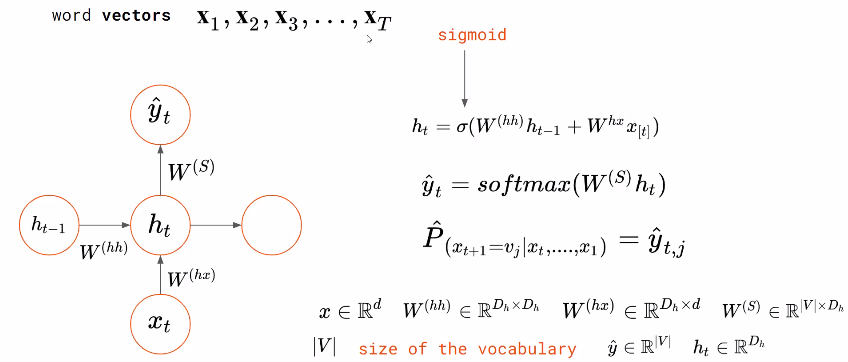
RNN về cơ bản là một hộp đen (bên trái của hình 3.7), nơi nó có “trạng thái bên trong” được cập nhật khi một chuỗi được xử lý. Tại mỗi bước thời gian, chúng ta đưa một vectơ đầu vào vô RNN, nơi nó sửa đổi trạng thái đó như một hàm của những gì nó nhận được. Khi điều chỉnh trọng số, RNN sẽ hiển thị các hành vi khác nhau về cách trạng thái của nó phát triển khi nhận được các đầu vào này. Chúng ta cũng quan tâm đến việc tạo ra một đầu ra dựa trên trạng thái RNN, vì vậy chúng ta có thể tạo ra các vectơ đầu ra này trên đầu RNN (như được mô tả trong hình 3.7).

RNN tại mỗi bước thời gian có hai đầu vào - một khung đầu vào (xi) và đại diện trước đây của những gì có vẻ như cho đến nay (tức là lịch sử) - để tạo ra một đầu ra yi và cập nhật lịch sử của nó, lịch sử này sẽ được truyền đi theo thời gian. Tất cả các khối RNN trong hình 3.7 (bên phải) là cùng một khối chia sẻ cùng một tham số, nhưng có các đầu vào và lịch sử khác nhau ở mỗi bước thời gian.

Ảnh có chứa văn bản, đồng hồ, ký hiệu

Mô tả được tạo tự động

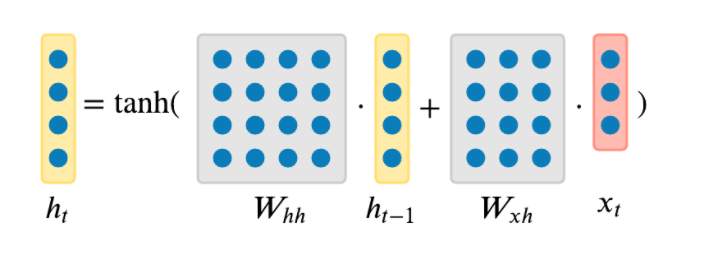
**Hình 3. 7 RNN rút gọn (trái) và RNN triển khai cụ thể (phải)** [**[19]**](#_TÀI_LIỆU_THAM_1)



**Hình 3. 8 Mô hình RNN đơn giản** [**[20]**](#_TÀI_LIỆU_THAM_1)

Ở dạng RNN đơn giản nhất (hình 3.8), mà chúng ta gọi là RNN Vanilla, mạng chỉ là một trạng thái ẩn duy nhất ***h,*** nơi sử dụng công thức lặp lại về cơ bản cho biết cách cập nhật trạng thái ẩn ***h*** như một chức năng của trạng thái ẩn trước đó và đầu vào hiện tại Qua đó, có ma trận trọng số và nơi chúng sẽ chiếu cả trạng thái ẩn từ bước thời gian trước và đầu vào hiện tại và sau đó sẽ được tổng hợp và thu gọn lại với hàm ***tanh*** chức năng cập nhật trạng thái ẩn tại thời điểm ***t***. Sự lặp lại này cho biết ***h*** sẽ thay đổi như thế nào dưới dạng một hàm trong lịch sử của nó và cũng là đầu vào hiện tại tại bước thời gian này:

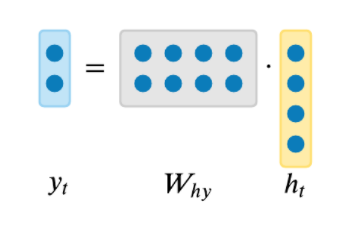
|  |  |
| --- | --- |
|  | (3.1) |



**Hình 3. 9 Công thức minh họa trạng thái ẩn ht** [**[19]**](#_Danh_mục_tham)

Chúng ta có thể dự đoán dựa trên đầu bằng cách chỉ sử dụng một phép chiếu ma trận khác trên đầu trạng thái ẩn. Đây là trường hợp hoàn chỉnh đơn giản nhất mà có thể kết nối mạng nơ-ron:

|  |  |
| --- | --- |
|  | (3.2) |



**Hình 3. 10 Công thức minh họa output yt** [**[19]**](#_Danh_mục_tham)

* + - 1. *Hàm loss*

Hàm mất mát (loss function) của cả mô hình bằng tổng mất mát của mỗi output, ở đây ta dùng cross entropy loss

|  |  |
| --- | --- |
|  | (3.3) |

Trong đó: ***T*** là độ dài câu/văn bản.

Backpropagation Through Time (BPTT): thuật toán lan truyền ngược là một kĩ thuật thường được sử dụng trong trong quá trình huấn luyện DNNs (deep neaural networks). Ý tưởng chung của thuật toán là sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng tham số (parameter/weight) của mạng. Gradient, sau đó, sẽ được sử dụng để cập nhật các tham số đó.

Quá trình trên sẽ được lặp lại cho tới khi các tham số (parameter/weight) của mạng hội tụ. Thông thường chúng ta sẽ có một siêu tham số (hyperparameter) định nghĩa cho số lượng vòng lặp để thực hiện quá trình trên. Một trong các siêu tham số đó thường được gọi là số epoch (hay số lần mà training set được duyệt qua một lần và weights được cập nhật). Nếu số lượng vòng lặp quá nhỏ, DNN có thể sẽ không cho ra kết quả tốt, và ngược lại thì thời gian huấn luyện sẽ quá dài nếu số lượng vòng lặp quá lớn. Ở đây ta có một sự đánh đổi (tradeoff) giữa độ chính xác và thời gian huấn luyện.

* + - 1. *Nhược điểm*

+ Mất nhiều thời gian để huấn luyện

+ Vanishing gradient và exploding gradient khi dùng BPTT:

* Nếu ở một thời điểm nào đó, gradient tiến gần về 0, thì các lần cập nhật trọng số (được thực hiện dựa trên gradient) sẽ không làm thay đổi nhiều trọng số của các layer ở tầng thấp hơn, khiến chúng không thể hội tụ và DNN sẽ không thu được kết quả tốt. Hiện tượng này được gọi là ***vanishing gradients***.
* Trong trường hợp khác, gradient có thể có giá trị lớn hơn trong quá trình backpropagation, làm một số layers có giá trị cập nhật cho weights quá lớn, khiến chúng phân kỳ (phân rã), dẫn đến DNN cũng không có kết quả như mong muốn. Hiện tượng này được gọi là ***exploding gradients***, và thường gặp khi sử dụng Recurrent Neural Networks (RNNs).
  + - 1. *Giải pháp*

Để giải quyết vấn đề về triệt tiêu/bùng nổ đạo hàm (vanishing gradient và exploding gradient) khi dùng BPTT, ta có thể có nhiều giải pháp như sau:

* Khởi tạo tham số ***W*** hợp lý sẽ giúp giảm được hiệu ứng mất mát đạo hàm,
* Sử dụng hàm kích hoạt ***ReLU*** thay cho hàm kích hoạt ***tanh*** và ***sigmoid***. Đạo hàm ***ReLU*** sẽ là một số hoặc là 0 hoặc là 1, nên có ta có thể kiểm soát được vấn đề triệt tiêu/bùng nổ đạo hàm.

Sử dụng kiến trúc nhớ dài-ngắn hạn (***LSTM*** - Long Short-Term Memory) hoặc Gated Recurrent Unit (***GRU***).

* + 1. **Bộ nhớ ngắn dài hạn – Long Short Term Memory (LSTM)**
       1. *Giới thiệu*

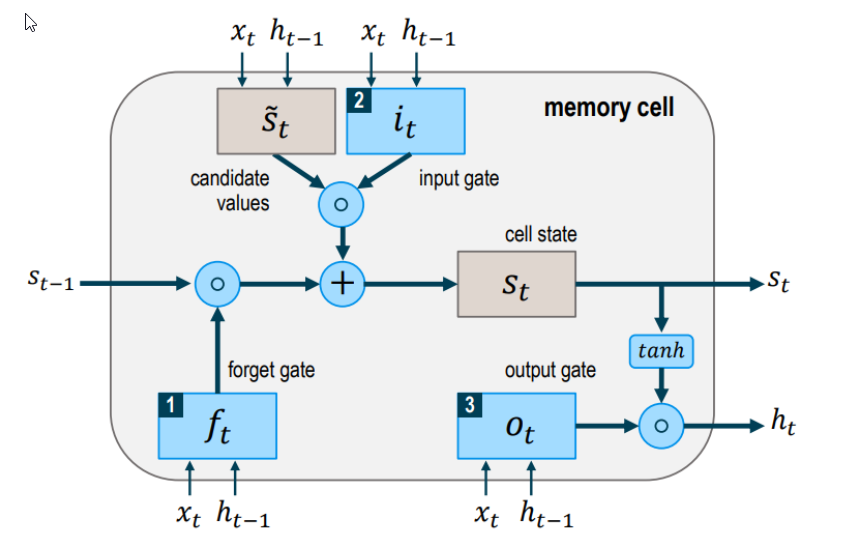
LSTM được đề xuất vào năm 1997 bởi Sepp Hochreiter và Jurgen Schmidhuber. Nó ra đời nhằm giải quyết vấn đề *long-term dependencies* trong RNN.

Thực tế, RNN không có khả năng ghi nhớ dữ liệu từ các bước thời gian quá xa (chỉ nhớ được các state gần). Do đó, các phần tử đầu tiên của chuỗi đầu vào không ảnh hưởng nhiều đến kết quả tính toán cho các bước phía sau. Mà điều này là một vấn đề.

RNN thuần có thể coi là một trường hợp đặc biệt của LSTM [[21]](#_Danh_mục_tham).

* + - 1. *Mô hình*

Một mạng LSTM có thể bao gồm nhiều tế bào LSTM (LSTM memory cell) liên kết với nhau. Chính LSTM memory cell làm nên điều đặc biệt với LSTM. Hình 3.11 dưới đây biểu diễn kiến trúc bên trong của một tế bào LSTM:



**Hình 3. 11 Mô hình biểu diễn kiến trúc bên trong của một tế bào LSTM** [**[21]**](#_Danh_mục_tham)

Mạng LSTM có thể bao gồm nhiều tế bào LSTM liên kết với nhau.

Trong đó:

+ : trạng thái bên trong tế bào (cell internal state) tại thời điểm t.

+: những thông tin tiềm năng (candidate values) có thể được thêm vào cell internal state tại thời điểm t.

+ , , : lần lượt là cổng quên, cổng sàng lọc thông tin đầu vào và đầu ra

+ : thời gian hiện tại

+ : giá trị đầu vào tại thời điểm hiện tại

+ : giá trị đầu ra của memory cell từ bước thời gian trước đó

Tùy vào mỗi cổng mà có chức năng khác nhau:

+ Cổng quên (forget gate - ): có nhiệm vụ loại bỏ những thông tin không cần thiết trạng thái bên trong tế bào

+ Cổng vào (input gate - ): có nhiệm vụ chọn lọc những thông tin nào cần thiết thêm vào trạng thái bên trong tế bào

+ Cổng ra (output - ): có nhiệm vụ xác định những thông tin nào từ trạng thái bên trong của tế bào được sử dụng

Trong quá trình lan truyền xuôi, trạng thái bên trong tế bào và giá trị đầu ra được tính như sau:

Đầu tiên, tế bào LSTM quyết định những thông tin nào cần được loại bỏ ở bước thời gian trước đó . Sau đó giá trị được tính dựa trên giá trị đầu vào hiện tại , giá trị đầu ra từ tế bào LSTM trước đó và bias của cổng quên.

|  |  |
| --- | --- |
|  | (3.4) |

Tiếp theo, LSTM quyết định những thông tin nào cần được thêm vào . Gồm 2 quá trình tính toán đối với và . Với và biểu diễn như sau:

|  |  |
| --- | --- |
|  | (3.5) |
|  | (3.6) |

Sau đó giá trị mới của được tính dựa trên kết quả tính toán thu được từ các bước trước với phép nhân hadamard theo từng phần tử (hadamard product) được ký hiệu bằng :

|  |  |
| --- | --- |
|  | (3.7) |

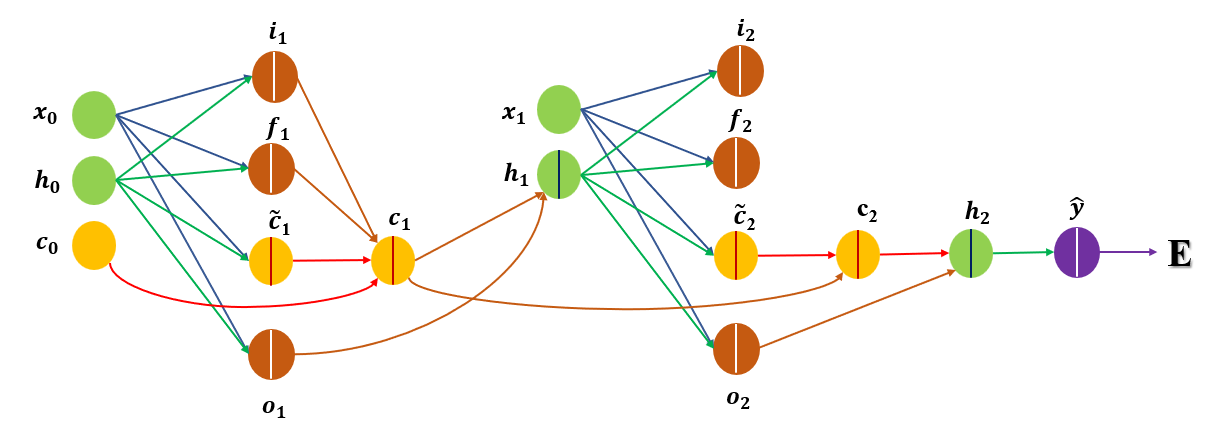
Cuối cùng, giá trị đầu ra của tế bào LSTM được tính dựa theo 2 phương trình:

|  |  |
| --- | --- |
|  | (3.8) |
|  | (3.9) |

* + - 1. *Lan truyền xuôi và lan truyền ngược*

Chúng tôi sẽ dùng một mạng LSTM gồm hai ô (tế bào LSTM) để nói về quá trình lan truyền xuôi (forward pass) và lan truyền ngược (backward pass) của LSTM, hình 3.12 đã minh họa điều này. Trong đó, lan truyền xuôi có trách nhiệm tính toán và lưu trữ các biến trung gian (bao gồm cả đầu ra) cho mạng nơron theo thứ tự từ lớp đầu vào đến lớp đầu ra. Lan truyền ngược được dùng như một thuật toán học tập để tính toán độ dốc gradient (gradient descent) liên quan đến trọng số trong các mô hình mạng nơ-ron.

Để tiện gợi nhớ trong lúc minh họa, trạng thái bên trong tế bào (cell internal state) thay vì được kí hiệu sẽ được ghi là , sẽ được thay bằng trong phần này. Sự thay đổi kí hiệu này sẽ không làm thay đổi bất kì ý nghĩa về mặt nội dung.



**Hình 3. 12 Mô tả trạng thái LSTM 2 ô**

Trong đó: Vòng tròn màu xanh lá tương ứng với đầu vào của giá trị. Trong mỗi ô tại thời điểm hiện tại, đầu vào sẽ gồm 2 phần tương ứng lần lượt là giá trị tại thời điểm và giá trị bước thời gian trước đó .

Vòng tròn màu vàng biểu thị ô nhớ. Tại vị trí đầu tiên, ô nhớ sẽ được gán giá trị bằng 0.

Vòng tròn màu cam là các cổng kiểm soát luồng thông tin. Các cổng ở đây bao gồm cổng đầu vào (input) , cổng quên (forget) và cổng đầu ra (output).

Những vòng tròn có đường cắt bao gồm 2 nhiệm vụ. Đầu tiên về phía bên trái hình cắt sẽ nhận một tổ hợp tuyến tính làm đầu vào sau đó đầu ra sẽ kết hợp với phía bên phải hình cắt là một chức năng kích hoạt. Tùy vào các cổng thì sẽ có các hàm kích hoạt được sử dụng khác nhau.

* Lan truyền xuôi:

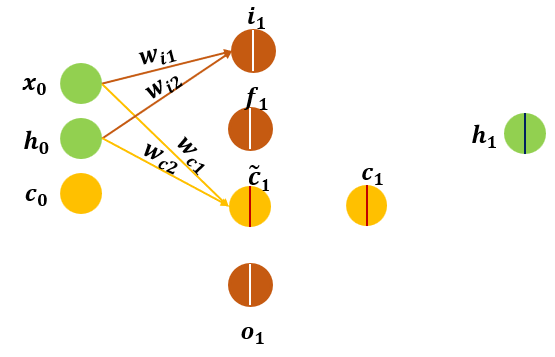
Giả sử chúng tôi có giá trị đầu vào tại 2 thời điểm tương ứng với một chuỗi đầu vào có giá trị vô hướng tại thời điểm ô thứ nhất. Sau đó chúng tôi khởi tạo các trọng số và các độ lệch lần lượt theo hình 3.13 như sau:

Ảnh có chứa bàn

Mô tả được tạo tự động

**Hình 3. 13 Khởi tạo các trọng số và độ lệch**

Hình 3.14 mô tả chi tiết cổng đầu vào của một LSTM tại ô đầu tiên:



**Hình 3. 14 Mô hình đầu vào của LSTM tại ô đầu tiên**

Chúng tôi dùng kí hiệu “net” để tính giá trị ròng hoặc ban đầu (net input) cho các cổng. Tại vị trí đầu vào :

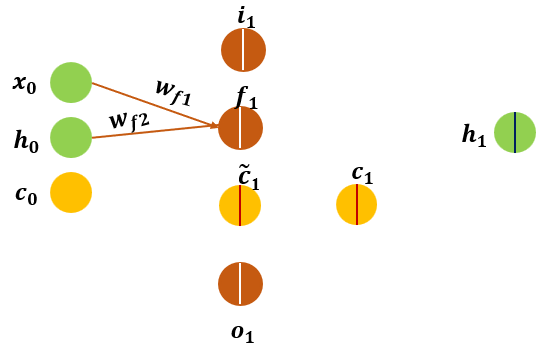
|  |
| --- |
| + |
|  |

* Từ kết quả được tính trên. Giả sử, ta cho giá trị của hàm . Qua đó cho phép thông tin đi từ vào ô nhớ

Tại vị trí ô nhớ của cổng đầu vào :

|  |
| --- |
| + |
|  |

Hình 3.15 mô tả chi tiết cổng quên của LSTM tại ô đầu tiên:



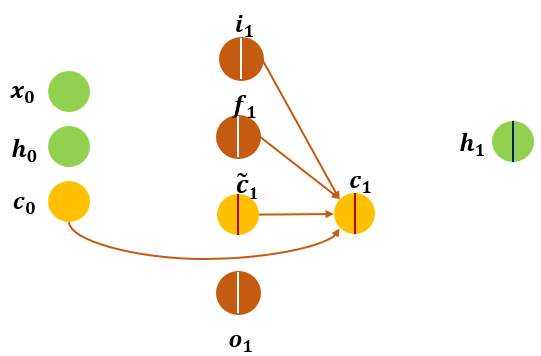
**Hình 3. 15 Mô tả cổng quên LSTM tại ô đầu tiên**

Tại vị trí cổng quên :

|  |
| --- |
| + |
|  |

Ở tại cổng quên có nhiệm vụ loại bỏ những thông tin không cần thiết trạng thái bên trong tế bào. Với giá trị bằng 0 sẽ quyết định loại bỏ và ngược lại. Vì vậy chúng tôi giả sử giá trị . Qua đó thông tin sẽ được giữ lại.

Sau khi có được thông tin từ cổng đầu vào, ô nhớ và cổng quên thì chúng tôi kết hợp thông tin lại với nhau để tính giá trị của ô nhớ được mô tả qua hình 3.16 như sau:

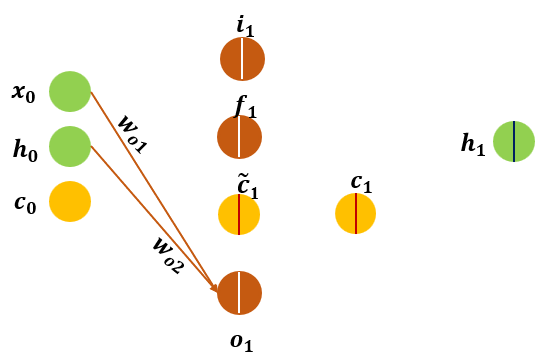


**Hình 3. 16 Mô tả thông tin ô nhớ được tổng hợp từ ,,**

Tại vị trí ô nhớ **:**

|  |
| --- |
| + |

Sau khi cập nhật giá trị của ô nhớ. Chúng tôi sẽ tiếp tục xét đến giấ trị của đầu ra được biểu diễn qua hình 3.17 như sau:



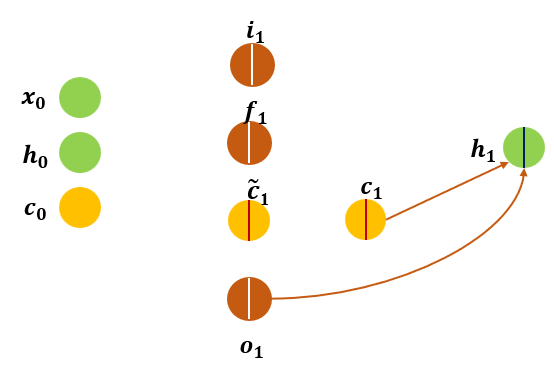
**Hình 3. 17 Mô tả thông tin đầu ra**

Tại vị trí đầu ra :

|  |
| --- |
| + |
|  |

Sau khi kết quả được tính xong, chúng tôi sẽ xét giá trị xem đầu ra có thỏa mãn yêu cầu đối với hàm sigmoid hay không. Để phù hợp cho bài toán chúng tôi đang triển khai vì vậy giả sử .

Tiếp theo đó, chúng tôi sẽ cập nhật giá trị trạng thái ẩn . Ngoài ra còn được hiểu là giá trị lịch sử của trạng thái trước đó được mô tả qua hình 3.18 như sau:



**Hình 3. 18 Mô tả thông tin trạng thái ẩn**

Tại vị trí đầu ra :

|  |
| --- |
|  |

Sau khi đã thực hiện các bước tính toán ở trạng thái ô đầu tiên LSTM. Tiếp theo, chúng tôi sẽ bắt đầu thực hiện tính toán LSTM ô thứ 2. Giả sử các giá trị đầu vào , trọng số không thay đổi và giá trị vô hướng đích . Tương tự như các bước trên, các công thức ở ô nhớ này sẽ được tính như sau.

Tại vị trí đầu vào :

|  |
| --- |
| + |
|  |

Tại vị trí ô nhớ của cổng đầu vào :

|  |
| --- |
| + |
|  |

Tại vị trí cổng quên :

|  |
| --- |
| + |
|  |

Tại vị trí ô nhớ **:**

|  |
| --- |
| + |

Tại vị trí đầu ra :

|  |
| --- |
| + |
|  |

Tại vị trí đầu ra :

|  |
| --- |
|  |

Tại vị trí :

+

Sau đó, chúng tôi đi tính lỗi bình phương trung bình (MSE) được kí hiệu là E. Giả sử giá trị thực thì E được tính như sau:

* Lan truyền ngược:

Lúc này, chúng tôi sẽ đi ngược lại quá trình tình toán bên trên. Đầu tiên, chúng tôi đi tính độ dốc của lỗi với output:

Đối với độ dốc của lỗi với ô nhớ thứ hai , công thức sẽ là:

Tiếp đó, chúng tôi đi tính giá trị ở cổng input và cổng forget. Với cổng input, công thức là:

Với cổng forget, công thức là:

Để tính cho giá trị đề xuất , ta có công thức:

Trạng thái của ô trước đó nữa sẽ được tính như sau:

Sau đó, chúng tôi tính các giá trị “net” ứng với từng cổng lần lượt như sau.

Tại cổng input:

Tại cổng forget:

Tại cổng output:

## 

## **CHƯƠNG 4: XÂY DỰNG MÔ HÌNH**

1. **MÔI TRƯỜNG CÀI ĐẶT VÀ CÁC THƯ VIỆN LIÊN QUAN**
   1. **Môi trường cài đặt**

Toàn bộ bài toán được thực hiện trên Google Colaboratory (Google Colab hay Colab) là một sản phẩm của Google Research. Nó cho phép người dùng viết và thực thi các đoạn code Python một cách đơn giản trong trình duyện (Chrome, Microsoft Edge,…). Colab là một sự lựa chọn thích hợp với data analysis, machine learning và giáo dục. Khi sử dụng Colab, ta không cần cấu hình thêm điều gì cho máy mà còn được sử dụng GPU (Graphics Processing Unit) và TPU (Tensor Processing Unit) miễn phí từ Colab. GPU là một bộ xử lý bổ sung để nâng cao giao diện đồ họa và chạy các tác vụ, thuật tuấn cao cấp với hàng nghìn tỷ phép toán. Còn TPU là một mạch tích hợp dành riêng cho ứng dụng, để tăng tốc các tính toán và thuật toán AI. Thông thường, Colab cung cấp GPU loại Nvidia K80s.

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Hình 4. 1 Cấu trúc phần cứng Colab cung cấp** [**[22]**](#_Danh_mục_tham)

Trong hình 4.1 là cấu trúc phần cứng mà Colab cung cấp cho người dùng sử dụng. Vì là hỗ trợ miễn phí nên Colab có giới hạn thời gian sử dụng, tối đa là 12 giờ. Mỗi lần mở lại trình sử dụng, Colab yêu cầu người dùng thực thi toàn bộ code từ đầu.

Chúng tôi lựa chọn Colab để thực hiện bài toán vì đây là một nền tảng miễn phí được cung cấp bởi Google và là một xu hướng tốt trong lĩnh vực nghiên cứu học sâu. Hơn hết, nó cung cấp các tài nguyên hiệu quả (GPU, TPU) nhằm nâng cao hiệu suất huấn luyện, thời gian thực hiện các mô hình, thuật toán phức tạp mặc dù máy thật của ta không quá mạnh. Ngoài ra, cách sử dụng của nó rất đơn giản, các thư viện cần thiết (Tensorflow, Sklearn, …) chỉ cần gọi là có thể sử dụng mà không phải qua bước cài đặt.

* 1. **Một số thư viện sử dụng**

*4.1.2.1 Thư viện Tensorflow*

Thư viện Tensorflow là thư viện mã nguồn mở cho machine learning nổi tiếng thế giới, được phát triển trên ngôn ngữ Python bởi các nhà nghiên cứu từ Google. Nó xuất hiện lần đầu tiền vào năm 2015 và được cải tiến để trở thành phiên bản ổn định hơn vào năm 2017. Tensorflow cho phép người dùng tạo các biểu đồ luồng dữ liệu (dataflow graph), cấu trúc mô tả cách dữ liệu di chuyển trong graph hay các node (series nodes).

Ảnh có chứa văn bản, thiết bị, thiết bị đo

Mô tả được tạo tự động

**Hình 4. 2 Minh họa graph trong Tensorflow** [**[23]**](#_Danh_mục_tham)

Hình mô tả một graph trong Tensorflow. Graph bao gồm cạnh (edge) và các node. Trong đó, node đại diện cho hoạt động toán học và các cạnh chứa tensor. Tensor là một mảng có n chiều (n-dimensional array). Một tensor có thể được bắt nguồn từ dữ liệu input hay kết quả của một tính toán từ các node. Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau qua các node. Một số loại tensor như:

* Tensor 0 chiều (scalar): 1, 2, -9, …
* Tensor 1 chiều (vector):
* Tensor 2 chiều (matrix):
* Tensor từ 3 chiều trở lên gọi chung là tensor.

Tensorflow có nhiều đặc điểm nổi bật để trở thành lựa chọn ưu tiên của người lập trình về machine learning như sau:

* Những bước lặt vặt như triển khai các thuật toán, hay tìm ra cách hợp lý để chuyển output của một chức năng sang input của một chức năng khác hoặc ngược lại để được Tensorflow xử lí. Nhờ đó, người lập trình có thể tập trung vào phần logic của ứng dụng đang thực hiện.
* Tensorflow được trang bị nhiều tính năng hữu dụng. Nếu người dùng chưa cảm thấy hài lòng về cách thức thực hiện cũng như kết quả mà Tensorflow mang đến, họ có thể trực tiếp chỉnh sửa các operation trong biểu đồ hướng xử lý dữ liệu nhờ tính năng eager execution. Nếu người lập trình muốn quan sát một cách trực tiếp quy trình làm việc của Tensorflow để phát hiện ra những sai sót thì có thể dùng tính năng TensorBoard.
* Tensorflow được phát triển bởi Google, vì thế nó liên tục được cải tiến và cập nhật sao cho phù hợp với xu hướng nghiên cứu hiện đại.

Vào năm 2019, Google đã phát hành phiên bản mới của thư viện học sâu Tensorflow (Tensorflow 2) tích hợp trực tiếp API Keras và quảng bá giao diện này làm giao diện mặc định hoặc giao diện tiêu chuẩn để phát triển học sâu trên platform. Sự tích hợp này thường được gọi là giao diện tf.keras (“tf” là viết tắt của Tensorflow). Ta gọi như thế là để phân biệt nó với dự án mã nguồn mở Keras độc lập. Nhờ đó, thay vì phải sử dụng hai thư viện là Tensorflow và Keras độc lập thì giờ, ta chỉ cần dùng tf.keras. Thực chất, ưu điểm của Keras độc lập vẫn được duy trì và cải tiến ở tf.keras. Nó ra đời nhằm tập trung vào việc cho phép các thử nghiệm tiến hành nhanh hơn để thu về hiệu quả nhanh nhất.

Keras nói chung chấp nhận ba loại đầu vào là mảng numpy, đối tượng Tensorflow Dataset và trình tạo Python.

*4.1.2.2 Thư viện scikit – learn*

Scikit-learn (sklearn) là thư viện mã nguồn mở dành cho học máy được viết trên Python. Nó cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering và dimensionality reduction. Sklearn được thiết kế để xử lý các thư viện số và khoa học của Python như NumPy và SciPy.

Scikit-learn ban đầu được đề xuất bởi David Cournapeau trong một dự án Summer Code của Google vào năm 2007. Later Matthieu Brucher tham gia dự án trên và bắt đầu sử dụng nó làm một phần luận văn tiến sĩ của ông ấy. Vào năm 2010, viện nghiên cứu khoa học và công nghệ kỹ thuật số quốc gia của Mỹ (National Institute for Research in Digital Science and Technology - INRIA) bắt đầu tài trợ và phiên bản đầu tiên được xuất bản (v0.1 beta) vào cuối tháng 1 năm 2010 [[24]](#_Danh_mục_tham).

Một số ưu điểm của thư viện scikit-learn:

* Thư viện scitkit-learn cung cấp rất nhiều thuật toán được tạo sẵn để thực hiện cả học máy có giám sát và không giám sát.
* Đối với các thuật toán khác nhau, scitkit-learn đảm bảo cấu trúc mã cho đào tạo và dự đoán mô hình đều giống nhau. Ví dụ: Sau khi lựa chọn mô hình, bước tiếp theo là làm cho mô hình phù hợp với một số dữ liệu đào tạo với phương thức fit(). Sau đó, ta dùng **train\_test\_split()** để chia tập dữ liệu thành dữ liệu thử nghiệm (test) và huấn luyện (train). Cuối cùng, ta gọi predict() để đưa ra dự đoán.
* Scikit-learn có các phương thức tích hợp để thực hiện các bước tiền xử lý.
* Về ước tính hiệu suất mô hình, scikit-learn cung cấp phương thức score(), nó trả về một chỉ số mặc định có liên quan nhất đến nhiệm vụ học máy mà đang được thực hiện.

Scikit-learn được sử dụng bởi những gã khổng lồ công nghệ như Inria, Mendeley, Wisdom.io, Evernote, Telecom Paris Tech và Aweber.

1. **MÔ TẢ DỮ LIỆU**

Để phục vụ cho hoạt động nghiên cứu đề tài này, nhiều bộ dữ liệu đã được thu thập và KDD 1999 được xem là sự lựa chọn phổ biến nhất. Nó được tạo cho thử thách KDD Cup vào năm 1999. Trong đó, KDD Cup là cuộc thi khai phá dữ liệu và khám phá kiến thức được tổ chức hàng năm bởi ACM Special Interest Group về khám phá kiến thức và khai thác dữ liệu (tổ chức chuyên nghiệp hàng đầu của những nhà nghiên cứu về khai phá dữ liệu) [[25]](#_Danh_mục_tham).

KDD 1999 gồm các tập dữ liệu huấn luyện và kiểm tra. Tập dữ liệu huấn luyện có 4.898.431 bản ghi, mỗi bản ghi có 41 thuộc tính (loại giao thức, dịch vụ và cờ) và được dán nhãn là bình thường hoặc một cuộc tấn công một cách chính xác với một kiểu tấn công cụ thể. Thông tin chi tiết 41 thuộc tính sẽ được chúng tôi minh họa trong bảng 4.1 sau.

**Bảng 4. 1 Thông tin các thuộc tính của bộ dữ liệu**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** |
| 1 | duration | Chiều dài (số giây) của kết nối |
| 2 | protocol\_type | Loại giao thức. Ví dụ tcp, udp,... |
| 3 | service | Dịch vụ mạng trên các điểm đến. Ví dụ http, telnet, ... |
| 4 | src\_bytes | Số lượng dữ liệu byte từ nguồn tới đích |
| 5 | dst\_bytes | Số lượng dữ liệu byte từ đích tới nguồn |
| 6 | flag | Tình trạng bình thường hoặc lỗi kết nối |
| 7 | land | Bằng 1 nếu kết nối là từ/đến host tương tự/port; bằng 0 nếu ngược lại |
| 8 | wrong\_fragment | Số lượng wrong fragments |
| 9 | urgent | Số lượng gói tin khẩn cấp |
| 10 | hot | Số lượng chỉ số nóng |
| 11 | num\_failed\_logins | Số lần đăng nhập thất bại |
| 12 | logged\_in | Bằng 1 nếu đăng nhập thành công; 0 nếu ngược lại |
| 13 | num\_compromised | Số lượng điều kiện thỏa hiệp |
| 14 | root\_shell | Bằng 1 nếu thu được root shell; 0 nếu ngược lại |
| 15 | su\_attempted | Bằng 1 nếu cố gắng thực hiện lệnh ''su root''; 0 nếu ngược lại |
| 16 | num\_root | Số lần truy cập quyền “root” |
| 17 | num\_file\_creations | Số hoạt động tạo tập tin |
| 18 | num\_shells | Số lượng shell prompts |
| 19 | num\_access\_files | Số hoạt động trên các tập tin kiểm soát truy cập |
| 20 | num\_outbound\_cmds | Số lượng lệnh outbound trong một phiên ftp |
| 21 | is\_host\_login | Bằng 1 nếu đăng nhập thuộc về danh sách ''máy chủ'' đã biết; 0 nếu ngược lại |
| 22 | is\_guest\_login | Bằng 1 nếu đăng nhập là một tài khoản khách; 0 nếu ngược lại |
| 23 | count | Số lượng kết nối đến các máy chủ tương tự giống như các kết nối hiện hành trong 2 giây đã qua |
| 24 | serror\_rate | Số phần trăm kết nối có lỗi “SYN” |
| 25 | rerror\_rate | Số phần trăm kết nối có lỗi “REJ” |
| 26 | same\_srv\_rate | Số phần trăm các kết nối đến những dịch vụ tương tự |
| 27 | diff\_srv\_rate | Phần trăm kết nối với các dịch vụ khác nhau |
| 28 | srv\_count | Số lượng kết nối đến các dịch vụ tương tự giống nhau các kết nối hiện hành trong 2 giây đã qua |
| 29 | srv\_serror\_rate | Phần trăm kết nối có lỗi “SYN” từ các dịch vụ |
| 30 | srv\_rerror\_rate | Phần trăm kết nối có lỗi “REJ” từ các dịch vụ |
| 31 | srv\_diff\_host\_rate | Tỉ lệ phần trăm kết nối đến máy chủ khác nhau từ dịch vụ |
| 32 | dst\_host\_count | Đếm các kết nối có cùng một đích đến |
| 33 | dst\_host\_srv\_count | Số kết nối có cùng host đích và sử dụng các dịch vụ tương tự |
| 34 | dst\_host\_same\_srv\_rate | Phần trăm các kết nối có cùng host đích và sử dụng các dịch vụ tương tự |
| 35 | dst\_host\_diff\_srv\_rate | Phần trăm các dịch vụ khác nhau trên các host hiện hành |
| 36 | dst\_host\_same\_src\_port\_rate | Phần trăm các kết nối đến các host hiện thời có cùng cổng src |
| 37 | dst\_host\_srv\_diff\_host\_rate | Phần trăm các kết nối đến các dịch vụ tương tự đến từ các host khác nhau |
| 38 | dst\_host\_serror\_rate | Phần trăm các kết nối đến các host hiện thời có một lỗi S0 |
| 39 | dst\_host\_srv\_serror\_rate | Phần trăm các kết nối đến các host hiện hành và dịch vụ quy định rằng có một lỗi S0 |
| 40 | dst\_host\_rerror\_rate | Phần trăm các kết nối đến các host hiện thời có mỗi lỗi RST |
| 41 | dst\_host\_srv\_rerror\_rate | Phần trăm các kết nối đến các máy chủ hiện hành và dịch vụ quy định rằng có lỗi RST |

Ví dụ về một bản ghi trong KDD 1999:

0, tcp, ftp\_data, SF, 334, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 2, 20, 1.00,0.00, 1.00, 0.20, 0.00, 0.00, 0.00, 0.00, warezclient

Thông tin cuối cùng trong mỗi bản ghi chính là nhãn của kết nối. Trong ví dụ trên, nhãn cuối cùng của bản ghi là “warezclient” tương ứng với kiểu tấn công warezclient, một dạng tấn công R2L.

Tuy nhiên, một vài vấn đề nghiêm trọng gây ảnh hưởng đến kết quả nghiên cứu được tìm thấy trong bộ dữ liệu KDD 1999 vào năm 2009. Cụ thể, một số lượng lớn các bản ghi dư thừa (78% trong dữ liệu đào tạo và 75% trong dữ liệu kiểm tra) được tìm thấy gây ra sự sai lệch. Ngoài ra, khi chọn ngẫu nhiên các tập hợp con của dữ liệu đào tạo và kiểm tra, kết quả thu về thường có thể đạt được độ chính xác phi thực tế rất cao. Do vậy, NSL-KDD được tạo ra. Nó bao gồm các bản ghi đã chọn của tập dữ liệu KDD 1999 hoàn chỉnh và không gặp phải những thiếu sót nói trên.

Bộ dữ liệu NSL-KDD vẫn gồm 41 thuộc tính chưa kể nhãn của kết nối (kết nối bình thường hoặc tên một loại tấn công cụ thể), phần dữ liệu cho huấn luyện gồm 125.973 bản ghi và tập dữ liệu kiểm tra gồm 22.544 bản ghi, ít hơn nhiều so với tập dữ liệu KDD99 [[26]](#_Danh_mục_tham). Các tác giả cho rằng kích thước của tập dữ liệu NSL-KDD là hợp lý, có thể được sử dụng như tập dữ liệu hoàn chỉnh mà không cần phải lấy mẫu ngẫu nhiên. Điều này cho phép xem xét một cách nhất quán và có thể so sánh các công trình nghiên cứu khác nhau.

Trong bộ dữ liệu này, một bản ghi sẽ được phân loại theo trạng thái bình thường “Normal” (không bị tấn công) và cụ thể kiểu tấn công nếu có. Ngoài 22 loại kiểu tấn công xuất hiện trong phần dữ liệu huấn luyện, phần dữ liệu kiểm tra sẽ có thêm 17 kiểu tấn công nữa. Các kiểu tấn công này gồm các phân loại nhỏ hơn của Probe, Dos, U2R và R2L.

* 1. **TIỀN XỬ LÝ**

Trong số các thuộc tính của bộ dữ liệu NSL-KDD, có bốn thuộc tính có kiểu dữ liệu ở dạng phi số gồm thuộc tính protocol\_type (loại giao thức), thuộc tính service (dịch vụ mạng trên các điểm đến), thuộc tính flag (tình trạng bình thường hoặc lỗi kết nối) và nhãn kết nối (kết nối bình thường hoặc tên một loại tấn công cụ thể). Để phục vụ cho bài toán phân loại, chúng tôi sẽ tiến hành chuyển đổi kiểu dữ liệu phi số về dạng số bằng kỹ thuật mã hóa one-hot (one-hot encoding).

Trong cách mã hóa này, một “từ điển” cần được xây dựng chứa tất cả các giá trị khả dĩ của từng dữ liệu hạng mục. Sau đó mỗi giá trị hạng mục sẽ được mã hóa bằng một vector nhị phân với toàn bộ các phần tử bằng 0 trừ một phần tử bằng 1 tương ứng với vị trí của giá trị hạng mục đó trong từ điển (one-hot vector).

Ví dụ, nếu ta có dữ liệu một cột là “An” , “Bình”, “Yên” thì ta thực hiện các bước sau:

1. Xây dựng từ điển. Trong trường hợp này, từ điển có kích thước là 3 gồm: {“An” , “Bình”, “Yên”}.
2. Xác định chỉ số (vị trí trong từ điển) phù hợp: “An”: 0 , “Bình”: 1, “Yên”: 2
3. Mã hóa giá trị được minh họa trong bảng 4.2.

**Bảng 4. 2 Minh họa cho kỹ thuật mã hóa one-hot**

|  |  |  |
| --- | --- | --- |
| **An** | **Bình** | **Yên** |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Dạng one-hot vector của các từ sẽ được biểu diễn cụ thể trong bảng 4.3.

**Bảng 4. 3 Minh họa cho kỹ thuật mã hóa one-hot (tiếp theo)**

|  |  |
| --- | --- |
| **Từ** | **one-hot vector** |
| An | [1,0,0] |
| Bình | [0,1,0] |
| Yên | [0,0,1] |

Số chiều của vector này đúng bằng số từ trong từ điển. Diễn giải theo một cách khác, mỗi giá trị nhị phân trong vector này thể hiện việc giá trị hạng mục đang xét “có phải là” giá trị tương ứng trong từ điển không. Với các giá trị mới không nằm trong từ điển (out-of-vocabolary hay OOV), ta có thể mã hóa chúng thành [0, 0, 0] theo nghĩa chúng không phải là bất cứ một giá trị nào trong từ điển.

Vận dụng kỹ thuật này, sau khi biến đổi chúng tôi thu được một kết quả được minh họa như hình 4.2 trên năm đối tượng đầu tiên. Trong đó phần bên trái là trước khi biến đổi, phần bên phải là sau khi áp dụng kỹ thuật này.



**Hình 4. 3 Minh họa sử dụng kỹ thuật mã hóa one-hot trong bài toán**

Đối với nhãn kết nối, ngoài nhãn “Normal” thể hiện một kết nối bình thường, còn lại sẽ là tên một loại tấn công thuộc bốn nhóm Dos, Probe, R2L và U2R. Vì thế, chúng tôi sẽ tiến hành đánh số cho chúng như trong bảng 4.4.

**Bảng 4. 4 Minh họa cách biến đổi cho thuộc tính nhãn kết nối**

|  |  |  |
| --- | --- | --- |
| **Loại nhãn** | **Các phân loại** | **Đánh số** |
| Normal | normal | 0 |
| Dos | neptune, back, land, pod, smurf, teardrop, mailbomb, apache2,processtable, udpstorm, worm | 1 |
| Probe | ipsweep, nmap, portsweep, satan, mscan, saint | 2 |
| R2L | ftp\_write, guess\_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, httptunnel | 3 |
| U2R | buffer\_overflow, loadmodule, perl, rootkit, ps, sqlattack, xterm | 4 |

Sau đó, một thuộc tính khác được thêm vào nhằm xác nhận kết nói đó có hay không bị tấn công. Những nhãn “Normal”, thể hiện một kết nối bình thường, sẽ giữ nguyên đánh số “0”. Ngược lại, nhãn “1” thể hiện một kết nối bị tấn công.

**4.3.2 Lựa chọn thuộc tính**

Sau khi chuẩn hóa kiểu dữ liệu như trên, số thuộc tính sẽ được tăng lên từ 41 thuộc tính thành 118 thuộc tính, chưa kể thuộc tính nhãn kết nối. Do vậy, chúng tôi sẽ thực hiện lựa chọn thuộc tính (feature selection) trên 41 thuộc tính ban đầu. Mục đích của nó là giải quyết một số vấn đề về dữ liệu như các thuộc tính bị nhiễu, giảm số chiều dữ liệu, tăng tốc độ xử lí của thuật toán cũng như tăng độ chính xác của thuật toán.

Trong bài toán này, chúng tôi đã sử dụng phương pháp Correlation-based Feature Selection (CFS). CFS là một thuật toán lọc đơn giản đánh giá và xếp hạng các tập hợp con tính năng hơn là các tính năng riêng lẻ, dựa trên hai loại mối tương quan. Một là mối tương giữa quan đặc trưng và thuộc tính phân loại (một lớp cụ thể) và (feature - classification) và mối tương quan giữa hai đặc trưng (feature - feature) . Hai khái niệm này dựa trên giả thuyết sau: “Tập hợp con đặc trưng tốt chứa các đặc trưng có tương quan cao với việc phân loại, nhưng không tương quan với nhau”. CFS có thể được tính như sau (Ghiselli 1964):

|  |  |
| --- | --- |
|  | (4.1) |

Trong đó :

: mức độ liên quan của một tập hợp con thuộc tính S có chứa các thuộc tính k.

: giá trị trung bình của mối tương quan giữa thuộc tính () với biến lớp.

: giá trị trung bình của mối tương quan thuộc tính với thuộc tính.

: số thuộc tính của tập hợp con.

Việc triển khai CFS được sử dụng trong các thí nghiệm được mô tả trong dữ liệu này tìm kiếm theo phương pháp heuristic: tốt nhất trước (best first search - BFS). BFS có thể bắt đầu với hoặc không có đặc trưng hoặc tất cả các đặc trưng. Cách đầu, việc tìm kiếm lần lượt thêm các đặc trưng vào tập con; cách sau, việc tìm kiếm lần lượt xóa từng đặc trưng. Để ngăn chặn BFS khám phá toàn bộ không gian tìm kiếm tập con đặc trưng, một tiêu chí dừng được áp dụng. Việc tìm kiếm sẽ chấm dứt nếu năm tập con được mở rộng hoàn toàn liên tiếp cho thấy không có cải thiện so với tập con tốt nhất hiện tại.

Hall (2000) đã đề xuất một cách tiếp cận tìm kiếm đầu tiên tốt nhất. Cụ thể, việc tìm kiếm bắt đầu với một tập con trống và đánh giá cho mỗi đặc trưng giá trị của việc được thêm vào tập hợp trống. Đối với bước này, mối tương quan feature-feature có thể bị bỏ qua, vì mẫu số của phương trình trên đơn giản hóa thành 1, do

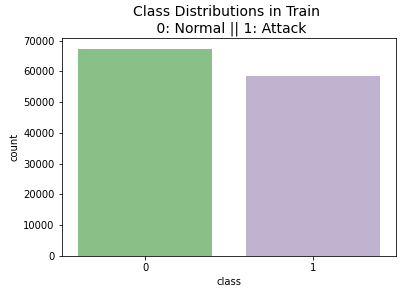
|  |  |
| --- | --- |
|  | (4.2) |

Chúng tôi sử dụng WEKA (Waikato Environment for Knowledge Analysis) để lựa chọn thuộc tính bằng CFS. Weka là một phần mềm mã nguồn mở viết trên ngôn ngữ Java và được phát triển tại đại học Waikato, New Zealand. Nó được biết đến như một công cụ đắc lực trong khai phá dữ liệu.

Trong phần lựa chọn thuộc tính của WEKA, chúng tôi đã sử dụng CFS và phương pháp tìm kiếm Best First. Kết quả, 19 thuộc tính được lựa chọn chưa kể nhãn kết nối. Cụ thể, 19 thuộc tính này lần lượt là thuộc tính thứ 2 (protocol\_type), thứ 3 (service), thứ 4 (src\_bytes), thứ 5 (dst\_bytes), thứ 6 (flag), thứ 7 (land), thứ 8 (wrong\_fragment), thứ 10 (hot), thứ 12 (logged\_in), thứ 23 (count), thứ 25 (rerror\_rate), thứ 29 (srv\_serror\_rate), thứ 30 (srv\_rerror\_rate), thứ 35 (dst\_host\_diff\_srv\_rate), thứ 36 (dst\_host\_same\_src\_port\_rate), thứ 37 (dst\_host\_srv\_diff\_host\_rate), thứ 38 (dst\_host\_serror\_rate), thứ 39 (dst\_host\_srv\_serror\_rate) và thứ 40 (dst\_host\_rerror\_rate).

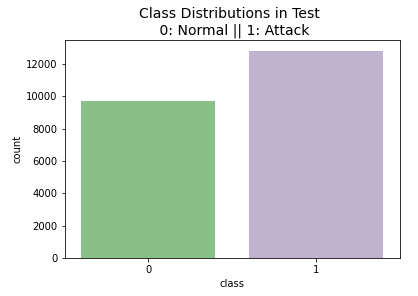
* 1. **THỰC NGHIỆM**

Lúc này, nhãn kết nối chỉ gồm “Normal” - kết nối bình thường, mang giá trị “0” và “Attack” - kết nối bị tấn công, mang giá trị “1”. Chúng tôi đã minh họa sự phân bố nhãn trên tập dữ liệu ở phần huấn luyện và thử nghiệm như bên dưới.



**Hình 4. 4 Minh họa sự phân bố nhãn kết nối trên dữ liệu huấn luyện**

Hình 4.4 minh họa sự phân bố nhãn kết nối trên dữ liệu huấn luyện. Trong đó, số đối tượng được gắn nhãn “Normal” là 67343 và nhãn “Attack” là 58630.



**Hình 4. 5 Minh họa sự phân bố nhãn kết nối trên dữ liệu thử nghiệm**

Hình 4.5 minh họa sự phân bố nhãn kết nối trên dữ liệu thử nghiệm. Trong đó, số đối tượng được gắn nhãn “Normal” là 9711 và nhãn “Attack” là 12833.

* + 1. **Phân loại không lựa chọn thuộc tính**

Trước tiên, chúng tôi thực hiện phân loại bài toán mà không thực hiện lựa chọn thuộc tính. Đây là một bài toán phân loại nhị phân, chúng tôi sẽ tiến hành thực nghiệm trên Logistic Regression và mô hình LSTM. Độ đo được sử dụng là accuracy (độ chính xác) và F1-score.

Accuracy là tỷ lệ các trường hợp được dự báo đúng trên tổng số các trường hợp. F1-score là trung bình điều hòa (harmonic mean) giữa recall (độ bao phủ) và precision (độ chuẩn xác). Trong đó, recall là tỉ lệ các dự đoán đúng mà đúng trong thực tế trên tổng số đối tượng dữ liệu đúng thật sự trong thực tế. Precision là tỉ lệ các dự đoán đúng mà đúng trong thực tế trên tổng số dự đoán đúng (bao gồm dự đoán đúng nhưng sai trong thực tế). Do vậy, giá trị F1-score càng cao thì kết quả phân loại càng tốt.

Lúc này, dữ liệu đầu vào sẽ là thông tin của các kết nối gồm 41 thuộc tính sau khi đã tiền xử lí mà không thực hiện bước lựa chọn thuộc tính. Dữ liệu đầu ra chính là nhãn kết nối, chứa hai giá trị “0” và “1”. Số lượng đối tượng kết nối ở hai lớp đều ở mức cân bằng nên chúng tôi có thể sử dụng accuracy.

Đối với Logistic Regression, chúng tôi đã đặt siêu tham số “C = 0.1”, giá trị mặc định của nó là 1. Nó là mức phạt đối với mô hình, giá trị “C” cao có nghĩa là có thể đặt sự tin tưởng thật cao cho dữ liệu huấn luyện, trong khi giá trị thấp cho biết dữ liệu này có thể không đại diện đầy đủ cho dữ liệu thế giới thực.

Trong khi đó, cấu trúc mạng LSTM gồm hai tầng mạng được minh họa trong hình 4.6. Một tầng LSTM có 80 nút và một tầng chứa hàm kích hoạt “tanh” nhằm điều chỉnh lại giá trị đầu ra trong khoảng (-1;1). Tổng số tham số của chúng là 26321, trong đó cả 26321 tham số đều là tham số cần huấn luyện. Mặt khác, LSTM sử dụng hàm tối ưu RMSprop và hàm mất mát mean squared error (MSE).

Ảnh có chứa bàn

Mô tả được tạo tự động

**Hình 4. 6 Cấu trúc mạng LSTM**

Có thể hiểu, RMSprop duy trì tốc độ học cho mỗi tham số. Tuy nhiên, chúng được điều chỉnh dựa trên mức độ trung bình của các cường độ gần của các gradient đối với trọng số (weight) (ví dụ đơn giản là tốc độ thay đổi của nó). Điều này có ý nghĩa trong việc giải quyết các vấn đề trực tiếp và không có tính ổn định. Hàm mất mát mean squared error (MSE) tính sự khác biệt bình phương trung bình giữa các giá trị ước tính và giá trị thực.

Tiếp theo, chúng tôi sử dụng phương thức fit() của Tensorflow để thực hiện huấn luyện mô hình phiên bản LSTM. Dữ liệu tập “train” được đưa vào như input của phương thức này. Dữ liệu đầu vào sẽ được chia thành các phần nhỏ hơn gọi là batch, mặc định giá trị này là 32. Tiếp sau đó là tham số epochs, nó biểu thị số lần lặp quá trình huấn luyện của mô hình trên tất cả các batch. Hình 4.7 minh họa quá trình huấn luyện của LSTM. Mỗi hàng lần lượt biểu thị lần epochs, thời gian thực hiện, giá trị mất mát và độ chính xác trên phần huấn luyện, giá trị mất mát (kí hiệu là “loss” theo hình). Tuy quá trình huấn luyện được thực hiện trên tất cả các batch mỗi epoch nhưng kết quả chỉ biểu thị batch cuối cùng ứng với mỗi epoch. Trong LSTM này, chúng tôi sử dụng giá trị mặc định cho số lượng batch là 32 và epochs là 10. Khi thử tăng giá trị của epochs lên 12, chúng tôi thấy độ chính xác có phần sụt giảm nên giữ nguyên là 10.

Ảnh có chứa bàn

Mô tả được tạo tự động

**Hình 4. 7 Quá trình huấn luyện LSTM**

Kết quả thực nghiệm được mô tả trong bảng 4.5 lần lượt là accuracy, F1-score và ma trận nhầm lẫn (confusion matrix). Mỗi ma trận nhầm lẫn đều gồm 4 thống số lần lượt là true positive (TP), false positive (FP), true negative (TN) và false negative (FN). Trong đó, FP và FN là các giá dự đoán sai, TP và TN là các giá dự đoán đúng.

**Bảng 4. 5 Minh họa kết quả cho phân loại không lựa chọn thuộc tính**

|  |  |  |  |
| --- | --- | --- | --- |
| Thông số  Thuật toán | Accuracy | F1-score | Ma trận nhầm lẫn |
| Logistic Regression | 96.98% | 97.29% |  |
| LSTM | 78.93% | 78.54% |  |

Quan sát bảng 4.5, kết quả dự đoán theo Logistic Regression có 9666 trường hợp TN, 12198 trường hợp TP, 635 trường hợp FN và 45 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp sử dụng Logistic Regression lần lượt là 95.05% và 99.63%.

Mặt khác, kết quả dự đoán theo LSTM có 9100 trường hợp TN, 8694 trường hợp TP, 4139 trường hợp FN và 611 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp sử dụng LSTM lần lượt là 67.75% và 93.43%. LSTM đã đưa ra dự đoán sai khá nhiều so với giá trị thật sự của nhãn kết nối.

Có thể thấy, khi thực hiện phân loại mà không tiến hành lựa chọn thuộc tính thì kết quả thu về từ LSTM không cao mà thậm chí còn thấp hơn Logistic Regression. Với kết quả này, chúng tôi tiếp tục thực nghiệm bài toán phân loại nhưng có tiến hành lựa chọn thuộc tính nhằm cải thiện độ chính xác của các thuật toán, đặc biệt là LSTM.

* + 1. **Phân loại có lựa chọn thuộc tính**

Trong trường hợp tiến hành lựa chọn thuộc tính bằng CFS kết hợp với phương pháp tìm kiếm Best First, chúng tôi đã tiến hành thực nghiệm trên một số thuật toán phân loại cổ điển gồm Random Forest, Logistic Regression, Bernoulli Naive Bayes và Support Vector Machine (SVM). Đồng thời, chúng tôi cũng tiến hành trên mô hình LSTM. Độ đo được sử dụng là accuracy (độ chính xác) và F1-score.

Lúc này, dữ liệu đầu vào sẽ là thông tin của các kết nối gồm 19 thuộc tính sau khi đã tiền xử lí và thực hiện lựa chọn thuộc tính. Dữ liệu đầu ra chính là nhãn kết nối, chứa hai giá trị “0” và “1”. Số lượng đối tượng kết nối ở hai lớp đều ở mức cân bằng nên chúng tôi có thể sử dụng accuracy.

**Bảng 4. 6 Mô tả kết quả trên một số thuật toán khác**

|  |  |  |  |
| --- | --- | --- | --- |
| Thông số  Thuật toán | Accuracy | F1-score | Ma trận nhầm lẫn |
| Random Forest | 98.24% | 98.43% |  |
| Logistic Regression | 98.07% | 98.28% |  |
| Bernoulli Naive Bayes | 96.13% | 96.71% |  |
| SVM | 97.46% | 97.75% |  |

Chúng tôi đã ghi lại nhanh kết quả phân loại trên bốn thuật toán trong bảng 4.6. Theo quan sát, Random Forest đã cho về kết quả cao nhất trong số các thuật toán cổ điển được sử dụng với accuracy đạt 98.24% và F1-score đạt 98.43%. Thuật toán này gồm hai siêu tham số quan trọng cần được thiết lặp gồm số lượng cây và độ sâu lớn nhất. Cụ thể, chúng tôi đặt giá trị cho hai siêu tham số này lần lượt là 70 và 35. Kết quả dự đoán theo Random Forest có 9710 trường hợp TN, 12437 trường hợp TP, 396 trường hợp FN và 0 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 96.91% và 100%.

Logistic Regression đã cho về kết quả với accuracy đạt 98.07% và F1-score đạt 98.28%. Chúng tôi đã đặt siêu tham số “C = 0.1”, giá trị mặc định của nó là 1. Kết quả dự đoán theo Logistic Regression có 9672 trường hợp TN, 12436 trường hợp TP, 397 trường hợp FN và 39 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 96.91% và 99.69%.

Bernoulli Naive Bayes đã cho về kết quả với accuracy đạt 96.13% và F1-score đạt 96.71%. Chúng tôi đã đặt siêu tham số “alpha = 0.0005”, nó đại diện cho tham số làm mịn phụ gia. Kết quả dự đoán theo Bernoulli Naive Bayes có 8843 trường hợp TN, 12829 trường hợp TP, 4 trường hợp FN và 868 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 99.97% và 93.66%.

SVM đã cho về kết quả với accuracy đạt 97.46% và F1-score đạt 97.75%. Kết quả dự đoán theo SVM có 9525 trường hợp TN, 12446 trường hợp TP, 387 trường hợp FN và 186 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 96.98% và 98.53%.

Tiếp theo, chúng tôi tiến hành phân loại trên mô hình mạng nơ-ron LSTM với năm phiên bản nhỏ. Kết quả phân loại được minh họa trong bảng 4.7. Trong đó, tên phiên bản chỉ có tác dụng phân biệt các phiên bản với nhau.

**Bảng 4. 7 Mô tả kết quả trên LSTM**

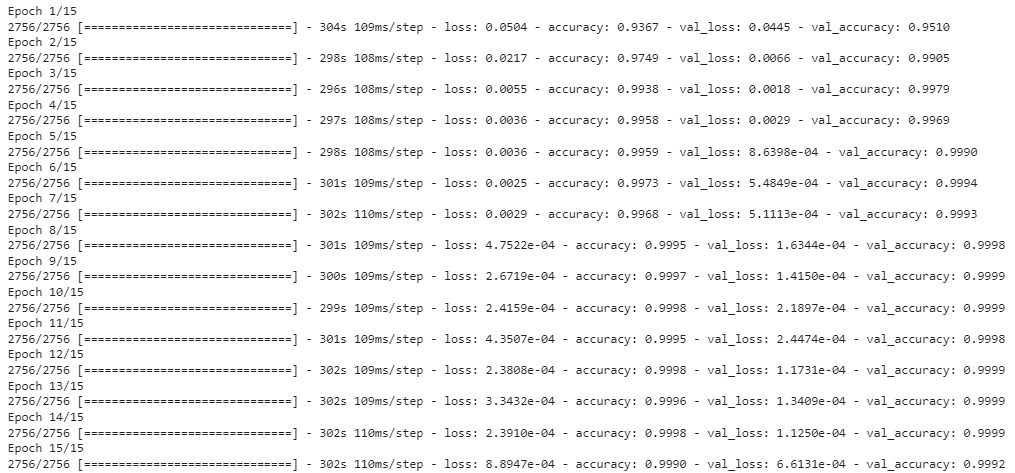
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên phiên bản | Accuracy | F1-score | Hàm tối ưu | Hàm mất mát | Ma trận nhầm lẫn |
| LSTM 1 | 94.77% | 95.30% | RMSprop | MSE |  |
| LSTM 2 | 99.40% | 99.47% | Adam | Binary crossentropy |  |
| LSTM 3 | 96.67% | 96.69% | Adam | Binary crossentropy |  |
| LSTM 4 | 93.96% | 94.40% | Adam | Binary crossentropy |  |
| LSTM 5 | 95.98% | 96.36% | RMSprop | MSE |  |

Ngoài RMSprop, chúng tôi sử dụng hàm tối ưu Adam trong trường hợp thực nghiệm này. Adam là sự kết hợp những lợi thế của hai thuật toán tối ưu AdaGrad và RMSprop nhằm giảm dần độ dốc ngẫu nhiên. Trong đó, hàm tối ưu AdaGrad duy trì tốc độ học trên mỗi tham số nhằm cải thiện hiệu suất đối với các vấn đề với dốc thưa thớt (sparse gradients). Thay vì điều chỉnh các tham số learning rate dựa trên thời điểm trung bình đầu tiên (giá trị trung bình) như RMSprop, Adam cũng sử dụng giá trị trung bình của thời điểm thứ hai của các gradient (Phương sai không tập trung).

Bên cạnh đó, hàm mất mát binary crossentropy cũng được sử dụng. Đây là hàm mất mát được sử dụng mặc định cho bài toán phân lớp nhị phân. Nó tính toán độ chênh lệch giữa hai phân phối xác suất của dự đoán và của nhãn thật.

Hai phiên bản đầu (LSTM 1 và LSTM 2) đều có chung một kiến trúc giống với phiên bản LSTM được sử dụng trong phần phân loại không lựa chọn thuộc tính đã được minh họa trong hình 4.6. Mặt khác, LSTM 1 sử dụng hàm mất mát mean squared error (MSE) và LSTM 2 sử dụng binary crossentropy vì số lớp bài toán chỉ gồm “1” cho kết nối bình thường và “0” cho kết nối bị tấn công.

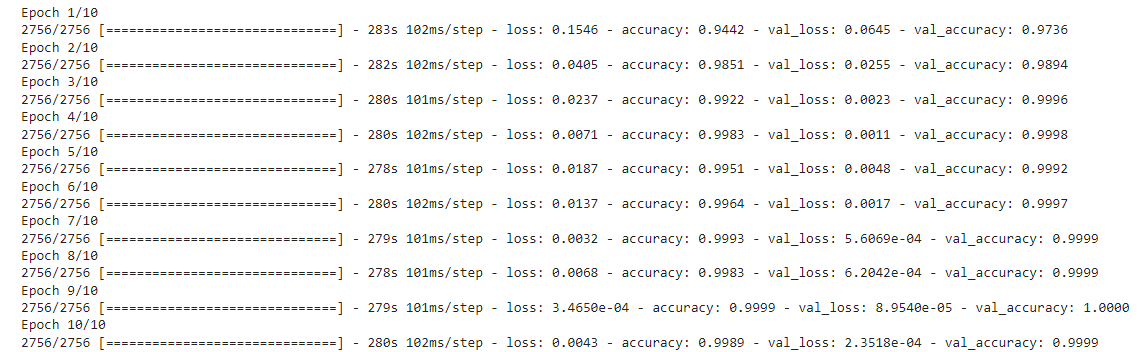
Tiếp theo, chúng tôi sử dụng phương thức fit() của Tensorflow để thực hiện huấn luyện mô hình phiên bản LSTM 1. Dữ liệu tập “train” được đưa vào như input của phương thức này. Dữ liệu đầu vào sẽ được chia thành các phần nhỏ hơn gọi là batch, mặc định giá trị này là 32. Tiếp sau đó là tham số epochs, nó biểu thị số lần lặp quá trình huấn luyện của mô hình trên tất cả các batch. Hình 4.8 đã minh họa quá trình huấn luyện của LSTM 1. Mỗi hàng lần lượt biểu thị lần epochs, thời gian thực hiện, giá trị mất mát và độ chính xác trên phần huấn luyện, giá trị mất mát (kí hiệu là “loss” theo hình). Tuy quá trình huấn luyện được thực hiện trên tất cả các batch mỗi epochs nhưng kết quả chỉ biểu thị lần epochs cuối cùng. Chúng tôi sử dụng giá trị cho số lượng batch là 32, epochs là 15 và validation\_split là 0.3 (70% tập “train” cho phần huấn luyện và 30% tập “train” cho validation) cho LSTM 1. Trong đó, ngoài các thông số cho phần “train”, mỗi hàng sẽ biểu thị thêm giá trị mất mát và độ chính xác trên phần validation (kí hiệu là “val”).



**Hình 4. 8 Quá trình huấn luyện LSTM 1**

Kết quả dự đoán theo LSTM 1 có 9388 trường hợp TN, 12976 trường hợp TP, 857 trường hợp FN và 323 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 93.32% và 97.37%.

Ở phiên bản LSTM 2, chúng tôi còn cấu hình thêm tham số validation\_split, nó sẽ chia tập dữ liệu đầu vào (tập “train”) thành hai phần nhỏ theo giá trị được chỉ định. Trong đó, một phần dùng để huấn luyện tìm ra các tham số và một phần khác dùng để kiểm tra độ chính xác của mô hình khi sử dụng các tham số huấn luyện đó, gọi là tập validation. Trường hợp này, chúng tôi sử dụng giá trị mặc định cho số lượng batch là 32, epochs là 10 và validation\_split là 0.3. Hình 4.9 đã minh họa quá trình huấn luyện của LSTM 2.



**Hình 4. 9 Quá trình huấn luyện LSTM 2**

Kết quả dự đoán theo LSTM 2 có 9618 trường hợp TN, 12728 trường hợp TP, 30 trường hợp FN và 105 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 99.18% và 99.76%.

Ba phiên bản sau (LSTM 3, LSTM 4 và LSTM 5) sử dụng thêm kỹ thuật dropout. Trong kiến trúc mạng nơ-ron, kỹ thuật dropout là việc chúng ta sẽ bỏ qua một vài unit trong suốt quá trình huấn luyện trong mô hình, những nút bị bỏ qua được lựa chọn ngẫu nhiên. Ở đây, chúng ta hiểu “bỏ qua - ignoring” là nút đó sẽ không tham gia và đóng góp vào quá trình huấn luyện. Tác dụng của nó là chống over-fitting (mô hình cho độ chính xác cao với dự liệu thử nghiệm nhưng không tốt trên dữ liệu khác).

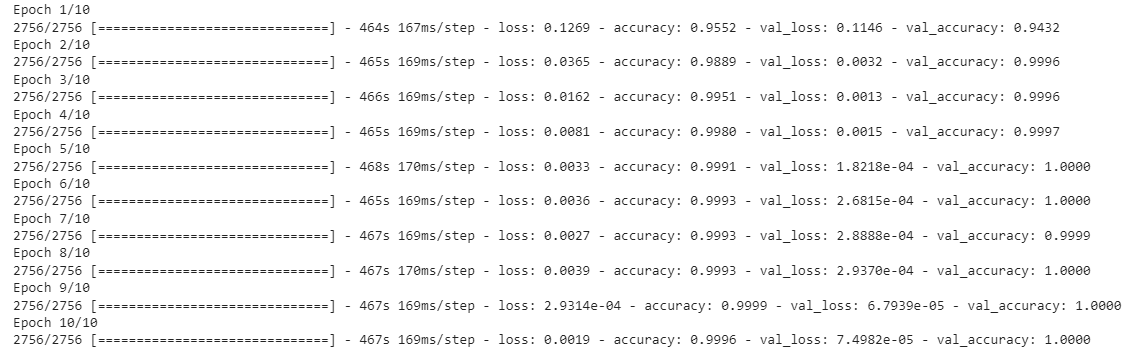
Cả phiên bản LSTM 3, LSTM 4 và LSTM 5 đều có cùng kiến trúc mạng được minh họa trong hình 4.10. Sau mỗi tầng mạng chính là một tầng chứa dropout. Tổng số tham số của chúng là 77841, trong đó cả 77841 tham số đều là tham số cần huấn luyện.

Ảnh có chứa bàn

Mô tả được tạo tự động

**Hình 4. 10 Cấu trúc LSTM 3, LSTM 4 và LSTM 5**

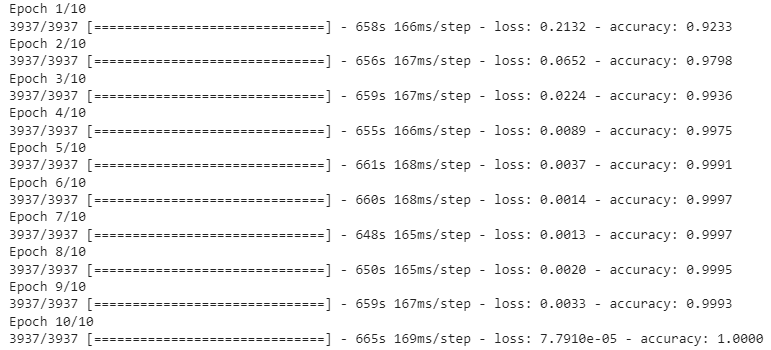
Tuy nhiên, các phiên bản LSTM 3, LSTM 4 và LSTM 5 vẫn tồn tại nhiều sự khác biệt so với nhau. Đầu tiên, LSTM 3 sử dụng giá trị mặc định cho số lượng batch là 32, epochs là 10 và validation\_split là 0.3. Chúng tôi đã minh họa quá trình huấn luyện của LSTM 3 trong hình 4.11.



**Hình 4. 11 Quá trình huấn luyện LSTM 3**

Kết quả dự đoán theo LSTM 3 có 9710 trường hợp TN, 12084 trường hợp TP, 749 trường hợp FN và 1 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 94.16% và 99.99%.

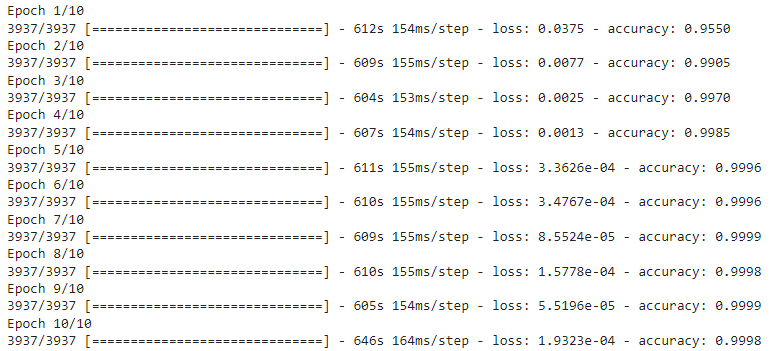
Kế đến trong LSTM 4, chúng tôi sử dụng giá trị mặc định cho số lượng batch là 32, epochs là 10 và không dùng validation\_split. Hình 4.12 đã minh họa quá trình huấn luyện của LSTM 4.



**Hình 4. 12 Quá trình huấn luyện LSTM 4**

Kết quả dự đoán theo LSTM 4 có 9750 trường hợp TN, 11478 trường hợp TP, 1355 trường hợp FN và 6 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 89.44% và 99.95%.

Phiên bản LSTM 5 là sự kết hợp của LSTM 1 và kỹ thuật dropout. Nó cùng sử dụng loại hàm mất mát và tối ưu giống LSTM 1. Nhưng vì kết hợp với dropout nên nó có cùng kiến trúc với LSTM 3 và LSTM 4. Trong LSTM 5, chúng tôi sử dụng giá trị mặc định cho số lượng batch là 32, epochs là 10 và không dùng validation\_split. Hình 4.13 đã minh họa quá trình huấn luyện của LSTM 5.



**Hình 4. 13 Quá trình huấn luyện LSTM 5**

Kết quả dự đoán theo LSTM 5 có 9657 trường hợp TN, 11981 trường hợp TP, 852 trường hợp FN và 54 trường hợp FP. Giá trị recall và precision được tính trong trưởng hợp này lần lượt là 93.36% và 99.55%.

Tổng thời gian thực hiện quá trình huấn luyện của mỗi mô hình LSTM thường dao động từ 55 phút đến 65 phút. Quá trình dự đoán trên tập thử nghiệm mất khoảng 5 phút. Quan sát thực nghiệm trong trường hợp phân loại nhị phân, chúng tôi thấy LSTM 2 cho kết quả cao nhất.

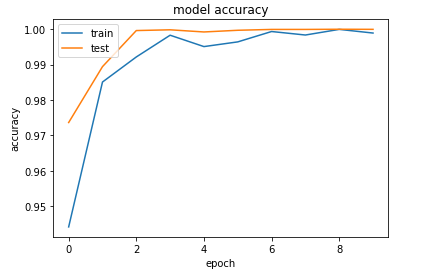
* 1. **KẾT QUẢ VÀ ĐÁNH GIÁ**

So sánh trước và sau khi tiến hành lựa chọn thuộc tính, kết quả phân loại đã thay đổi khá nhiều, nhất là với mô hình LSTM. Độ chính xác tăng từ 78.93% lên khoảng 98%, tăng khoảng 24%. So với mức tăng nhiều của LSTM thì Logistic Regression chỉ tăng thêm khoảng 1% (từ 96.98% tăng lên 98.07%). Điều này đã cho chứng tỏ được sự quan trọng của quá trình tiền xử lý trong lĩnh học máy, học sâu đặc biệt là đối với thực nghiệm trên các mô hình phức tạp như mạng nơ-ron.

**Bảng 4. 8 Tổng hợp kết quả thực nghiệm**

|  |  |  |
| --- | --- | --- |
| Thông số  Thuật toán | Accuracy | F1-score |
| Logistic Regression  (không lựa chọn thuộc tính) | 96.98% | 97.29% |
| LSTM  (không lựa chọn thuộc tính) | 78.93% | 78.54% |
| Random Forest | 98.24% | 98.43% |
| Logistic Regression | 98.07% | 98.28% |
| Bernoulli Naive Bayes | 96.13% | 96.71% |
| SVM | 97.46% | 97.75% |
| LSTM 1 | 94.77% | 95.30% |
| LSTM 2 | 99.40% | 99.47% |
| LSTM 3 | 96.67% | 96.69% |
| LSTM 4 | 93.96% | 94.40% |
| LSTM 5 | 95.98% | 96.36% |

Quan sát kết quả thực nghiệm tổng hợp được minh họa trong bảng 4.8, chúng tôi thấy phiên bản LSTM 2 mang về kết quả cao nhất. Cụ thể, chúng tôi đã quan sát thêm kết quả độ chính xác trong quá trình huấn luyện ở hình 4.14.



**Hình 4. 14 Biểu đồ thể hiện độ chính xác trong huấn luyện với LSTM 2**

LSTM 2 sử dụng tham số “validation\_split”, nên phần dữ liệu huấn luyện được chia thành phần “train” và “validation”. Trong hình 4.14, đường màu xanh thể hiện độ chính xác của phần “train” và đường màu cam thể hiện độ chính xác của phần “validation” trong 10 epoch. Rõ ràng, giá trị độ chính xác trên cả hai phần dữ liệu này không lệch nhau là mấy, độ lệch lớn nhất rơi vào epoch đầu tiên vào khoảng 0.03%. Khi dự đoán trên dữ liệu thử nghiệm, độ chính xác đạt 99.4%. Cả phần huấn luyện lẫn thử nhiệm đều có kết quả cao và không lệch nhau quá nhiều.

Tuy LSTM (LSTM 2) đã cho kết quả cao nhất nhưng chúng tôi thấy mức chênh lệch giữa kết quả khi sử dụng các thuật toán cổ điển và các mô hình LSTM là không nhiều. Có thể nói, kết quả của cả hai đều rất cao khi độ chính xác và F1-score đều xấp xỉ 100%. Chúng tôi nghĩ rằng, một trong những lí do khiến kết quả tốt như vậy là do bộ dữ liệu NSL-KDD được các chuyên gia đánh giá là rất tốt sau khi được cải tiến từ KDD 1999. Mặt khác, việc chúng tôi xử lí bài toán bằng cách đưa nó về dạng phân loại nhị phân có thể là một lí do khác. Trong dạng phân loại nhị phân, điều chúng tôi quan tâm là đối tượng kết nối đang trong trạng thái bình thường hay bị tấn công. Khi ấy, việc phân bố nhãn kết nối khá cân bằng đã tạo một điều kiện thuận lợi để xử lí bài toán.

Tuy kết quả khi thực nghiệm trên LSTM chưa thể đáp ứng được kì vọng ban đầu, nhưng chỉ xét về nhiệm vụ xử lí phân loại cho bài toán thì nó vẫn hoạt động tốt.

# 

# **PHẦN 3: KẾT LUẬN**

1. **ĐÓNG GÓP**

* **Về mặt lý thuyết**
* Tìm kiểu một số kĩ thuật ứng dụng trong quá trình tiền xử lí dữ liệu như mã hóa one-hot, sử dụng CFS trong lựa chọn thuộc tính, …
* Tìm hiểu WEKA trong lựa chọn thuộc tính.
* Có thêm nhiều kiến thức về môi trường Colab.
* Tìm hiểu về các mô hình mạng nơ-ron trong học sâu (RNN, LSTM).
* **Về mặt ứng dụng**
* Sử dụng đơn giản WEKA trong lựa chọn thuộc tính.
* Sử dụng Colab để thực hiện các thử nghiệm.
* Xây dựng mô hình mạng nơ-ron LSTM trong phát hiện xâm nhập mạng.

1. **HẠN CHẾ**

Quan sát kết quả thực nghiệm, so với mô hình LSTM chúng tôi thấy rằng chỉ với các thuật toán cổ điển đã mang lại một kết quả rất cao trong bài toán phân loại này.

Mặt khác, dữ liệu được sử dụng không quá lớn nên chưa thể đại diện cho toàn bộ dữ liệu thực tế, cũng như khó có thể đưa ra giải thích cho các vấn đề thực tiễn.

1. **HƯỚNG PHÁT TRIỂN**

Thay vì quan tâm đối tượng kết nối đang trong trạng thái bình thường hay bị tấn công, chúng tôi muốn biết nếu bị tấn công thì chính xác nó thuộc loại nào (Dos, Probe, R2L và U2R). Tức là từ bài toán phân loại nhị phân, chúng tôi sẽ giải quyết bài toán theo hướng phân loại đa nhãn. Tuy nhiên, sự mất cân bằng giữa các nhãn sẽ là một vấn đề rắc rối và mất nhiều thời gian để xử lí. Mặt khác, chúng tôi cũng mong đợi được sử dụng mạng nơ-ron GRU (Gated Recurrent Unit), một biến thể của LSTM trong giải quyết bài toán và đưa ra các so sánh với LSTM.

# **DANH MỤC THAM KHẢO**

|  |  |
| --- | --- |
| [1] | B. T. t. iSpace, “Tấn công mạng là gì?,” [Trực tuyến]. Available: https://ispace.edu.vn/tong-quan-ve-tan-cong-mang/. [Đã truy cập 04 2022]. |
| [2] | Công ty Luật Dương Gia, “Tấn công mạng là gì? Quy định phòng, chống tấn công mạng?,” 15 5 2022. [Trực tuyến]. Available: https://luatduonggia.vn/tan-cong-mang-la-gi-quy-dinh-phong-chong-tan-cong-mang/. [Đã truy cập 20 5 2022]. |
| [3] | H. T. M. Ngọc, “Tấn công phát tán Malware là hình thức tấn công gì?,” 27 04 2021. [Trực tuyến]. Available: https://nhanhoa.com/tin-tuc/tan-cong-phat-tan-malware-la-hinh-thuc-tan-cong-gi.html. [Đã truy cập 04 2022]. |
| [4] | vietnix, “DDoS là gì?,” [Trực tuyến]. Available: https://vietnix.vn/ddos-la-gi/. [Đã truy cập 04 2022]. |
| [5] | Q. Tỉnh, “Tấn công từ chối dịch vụ DoS và DDoS là gì? Tác hại của chúng ra sao?,” [Trực tuyến]. Available: https://quantrimang.com/tim-hieu-ve-tan-cong-tu-choi-dich-vu-dos-34926. [Đã truy cập 4 2022]. |
| [6] | hostingviet, “Tường Lửa (Firewall) Là Gì? Chức Năng Và Cách Thiết Lập Firewall,” [Trực tuyến]. Available: https://hostingviet.vn/tuong-lua-firewall-la-gi. [Đã truy cập 04 2022]. |
| [7] | monamedia, “Mạng riêng ảo VPN là gì?,” [Trực tuyến]. Available: https://monamedia.co/vpn-la-gi-tong-quan-ve-mang-rieng-ao-vpn/. [Đã truy cập 04 2022]. |
| [8] | quantrimang, “Hệ thống phát hiện xâm nhập (IDS) (Phần 1),” [Trực tuyến]. Available: https://quantrimang.com/he-thong-phat-hien-xam-pham-ids-phan-1-37334. [Đã truy cập 04 2022]. |
| [9] | J. Brownlee, “A Tour of Machine Learning Algorithms,” machinelearningmastery, 12 8 2019. [Trực tuyến]. Available: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/. [Đã truy cập 05 2022]. |
| [10] | H. C. Trung, “Giới thiệu về Support Vector Machine (SVM),” 20 8 2020. [Trực tuyến]. Available: https://viblo.asia/p/gioi-thieu-ve-support-vector-machine-svm-6J3ZgPVElmB. [Đã truy cập 05 2022]. |
| [11] | quantrimang, “Machine learning là gì? Deep learning là gì? Sự khác biệt giữa AI, machine learning và deep learning,” [Trực tuyến]. Available: https://quantrimang.com/su-khac-biet-giua-ai-hoc-may-va-hoc-sau-157948. [Đã truy cập 05 2022]. |
| [12] | N. T. Long, “Mạng nơ-ron nhân tạo (Artificial Neural Network),” 5 8 2019. [Trực tuyến]. Available: https://nguyentruonglong.net/ly-thuyet-ve-mang-no-ron-nhan-tao-artificial-neural-network-ann.html. [Đã truy cập 5 2022]. |
| [13] | ITNavi, “Tổng quan về Neural Network(mạng Nơ Ron nhân tạo) là gì?,” 13 5 2021. [Trực tuyến]. Available: https://itnavi.com.vn/blog/neural-network-la-gi/. [Đã truy cập 05 2022]. |
| [14] | R. Keim, “How to Use a Simple Perceptron Neural Network Example to Classify Data,” 17 11 2019. [Trực tuyến]. Available: https://www.allaboutcircuits.com/technical-articles/how-to-perform-classification-using-a-neural-network-a-simple-perceptron-example/. [Đã truy cập 05 2022]. |
| [15] | Y. W. Q. L. T. W. C. L. Z. ,. S. W. Xiujuan Wang, “RSC Advances,” *Prediction of the stress relaxation property of diene rubber composites by artificial neural network approaches,* 2015. |
| [16] | Wikipedia, “Recurrent neural network,” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Recurrent\_neural\_network. [Đã truy cập 05 2022]. |
| [17] | S. A. Afshine Amidi, “Mạng neural hồi quy cheatsheet,” stanford.edu, [Trực tuyến]. Available: https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-recurrent-neural-networks. [Đã truy cập 05 2022]. |
| [18] | cs231n, “Introduction to RNN,” [Trực tuyến]. Available: https://cs231n.github.io/rnn/. [Đã truy cập 05 2022]. |
| [19] | ProtonX, “Youtube,” 24 7 2020. [Trực tuyến]. Available: https://www.youtube.com/watch?v=t0EoeTYU-fc. [Đã truy cập 05 2022]. |
| [20] | N. T. Long, “Giải thích chi tiết về mạng Long Short-Term Memory (LSTM),” 18 10 2018. [Trực tuyến]. Available: https://nguyentruonglong.net/giai-thich-chi-tiet-ve-mang-long-short-term-memory-lstm.html. [Đã truy cập 5 2022]. |
| [21] | codelearn, “Google Colab Là Gì Và Dùng Để Làm Gì?,” 06 1 2020. [Trực tuyến]. Available: https://codelearn.io/sharing/google-colab-la-gi. [Đã truy cập 06 2022]. |
| [22] | N. V. Hiếu, “Bài 1 – Tổng quan về thư viện Tensorflow,” [Trực tuyến]. Available: https://nguyenvanhieu.vn/thu-vien-tensorflow/#thu-vien-tensorflow-la-gi. [Đã truy cập 06 2022]. |
| [23] | codelearn, “Thư Viện Scikit-learn Trong Python Là Gì?,” 06 02 2020. [Trực tuyến]. Available: https://codelearn.io/sharing/scikit-learn-trong-python-la-gi. [Đã truy cập 6 2022]. |
| [24] | M. I. a. E. G. M. I. Anna L. Buczak, “A Survey of Data Mining and Machine Learning,” *A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,* tập II, số 1553-877X, 2016. |
| [25] | UNB, “NSL-KDD dataset,” [Trực tuyến]. Available: https://www.unb.ca/cic/datasets/nsl.html. [Đã truy cập 06 2022]. |
| [26] | T. D. Tan, “Viblo - Mô hình phân lớp Naive Bayes,” 22 6 2019. [Trực tuyến]. Available: https://viblo.asia/p/mo-hinh-phan-lop-naive-bayes-vyDZO0A7lwj. [Đã truy cập 5 2022]. |
| [27] | Funda, “Bài 32: Naive Bayes Classifier,” 8 8 2017. [Trực tuyến]. Available: https://machinelearningcoban.com/2017/08/08/nbc/. [Đã truy cập 5 2022]. |
| [28] | phamdinhkhanh, “Hồi qui Logistic,” [Trực tuyến]. Available: https://phamdinhkhanh.github.io/deepai-book/ch\_ml/classification.html#xac-suat-cua-logistic-va-phan-phoi-bernoulli. [Đã truy cập 5 2022]. |
| [29] | nttuan8, 2 3 2019. [Trực tuyến]. Available: https://nttuan8.com/bai-2-logistic-regression/. [Đã truy cập 5 2022]. |
| [30] | trituenhantao, “Thuật ngữ về Deep Learning,” 6 7 2019. [Trực tuyến]. Available: https://trituenhantao.io/kien-thuc/thuat-ngu-ve-deep-learning-cho-linh-moi-phan-1/. [Đã truy cập 5 2022]. |
| [31] | T. Wood, “What is a Random Forest?,” [Trực tuyến]. Available: https://deepai.org/machine-learning-glossary-and-terms/random-forest. [Đã truy cập 06 2022]. |