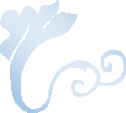
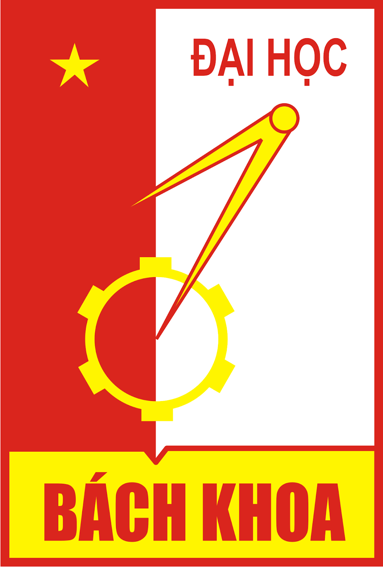


# HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY





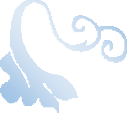
Báo cáo cuối kỳ

**IT5021 – Nghiên cứu đồ án tốt nghiệp 1**



|  |  |  |
| --- | --- | --- |
| **Thông tin sinh viên** | | |
| **Họ và tên** | **Lớp** | **MSSV** |
| Mai Minh Quân | Việt Nhật 03 | 20225661 |

**Giáo viên hướng dẫn:** Tiến sỹ Nguyễn Đức Toàn



Contents

[HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY 1](#_Toc199064851)

[I. Giới thiệu và Tổng quan về Machine Learning 5](#_Toc199064852)

[1.Tìm hiểu về Machine Learning 5](#_Toc199064853)

[2. Xác định một bài toán sử dụng Machine Learning 5](#_Toc199064854)

[3. Các kiểu học máy 5](#_Toc199064855)

[4. Kết hợp các kiểu học máy. 7](#_Toc199064856)

[5. Huấn luyện mô hình 7](#_Toc199064857)

[6. Các mô hình học máy 10](#_Toc199064858)

[II. Các Thuật toán Machine Learning Cơ bản 10](#_Toc199064859)

[1. K-Nearest Neighbors (KNN) 10](#_Toc199064860)

[2. Linear Regression 12](#_Toc199064861)

[3. Logistic Regression 16](#_Toc199064862)

[4. Naive Bayes 19](#_Toc199064863)

[5. Support Vector Machine 20](#_Toc199064864)

[III. Mạng Neural (Neural Networks) 24](#_Toc199064865)

[1.Tổng quan về Neural Network 24](#_Toc199064866)

[1.1 Giới thiệu về Neural Network 24](#_Toc199064867)

[1.2 Kiến trúc của mạng Neuron 25](#_Toc199064868)

[1.3 Activation Function 27](#_Toc199064869)

[1.4 Quá trình Feedforward 28](#_Toc199064870)

[1.5 Thuật toán Backpropagation 28](#_Toc199064871)

[2. Gradient Descent và Backpropagation 29](#_Toc199064872)

[2.1 Gradient Descent 29](#_Toc199064873)

[2.2 Backpropagation 31](#_Toc199064874)

[3. Hiện tượng Overfitting và cách khắc phục (L1, L2 Regularization) 33](#_Toc199064875)

[3.1 Hiện tượng Overfitting 33](#_Toc199064876)

[3.2 L2 Regulation 34](#_Toc199064877)

[3.3 L1 Regulation 37](#_Toc199064878)

[4. Các hàm kích hoạt (Activation Functions) 38](#_Toc199064879)

[4.1 Hàm kích hoạt (Activation Functions) 38](#_Toc199064880)

[4.2 Hàm kích hoạt Sigmoid (Sigmoid Activation Function) 39](#_Toc199064881)

[4.3 Hàm kích hoạt Tanh (Hyperbolic Tangent) 43](#_Toc199064882)

[4.4 Hàm kích hoạt ReLU (Rectified Linear Unit) 47](#_Toc199064883)

[4.5 Hàm kích hoạt Leaky ReLU 50](#_Toc199064884)

[4.6 Hàm kích hoạt Parametric ReLU (PReLU) 53](#_Toc199064885)

[4.7 Hàm kích hoạt ELU (Exponential Linear Unit) 54](#_Toc199064886)

[4.8 Hàm kích Hoạt Swish 56](#_Toc199064887)

[4.9 Hàm kích hoạt Softmax 58](#_Toc199064888)

[IV. Tiền xử lý dữ liệu (Data Preprocessing) 62](#_Toc199064889)

[1. Tổng quan về tiền xử lý dữ liệu 62](#_Toc199064890)

[1.1 Giới thiệu 62](#_Toc199064891)

[1.2 Các yếu tố ảnh hưởng đến chất lượng dữ liệu 62](#_Toc199064892)

[1.3 Các kỹ thuật tiền xử lý dữ liệu chính 63](#_Toc199064893)

[2. Data cleaning 65](#_Toc199064894)

[2.1 Xử lý giá trị thiếu (Missing Values) 66](#_Toc199064895)

[2.2 Xử lý dữ liệu nhiễu (Noisy Data) 68](#_Toc199064896)

[2.3 Làm sạch dữ liệu như một quy trình trong khai thác dữ liệu (Data Cleaning as a Process) 71](#_Toc199064897)

[3. Data integration 74](#_Toc199064898)

[3.1 Tổng quan 74](#_Toc199064899)

[3.2 Vấn đề nhận diện thực thể (The Entity Identification Problem) 74](#_Toc199064900)

[3.3 Redundancy and Correlation Analysis 75](#_Toc199064901)

[3.4 Hiệp phương sai (Covariance of Numeric Data) 80](#_Toc199064902)

[3.5 Trùng lặp dữ liệu (Tuple Duplication) 83](#_Toc199064903)

[3.6 Phát hiện và giải quyết xung đột giá trị dữ liệu (Detection and Resolution of Data Value Conflicts) 83](#_Toc199064904)

[V. Kiến trúc Transformer và Cơ chế Attention 85](#_Toc199064905)

[1. Cơ chế Attention 85](#_Toc199064906)

[1.1 Background 85](#_Toc199064907)

[1.2 Các Loại Attention Dựa Trên Hàm Điểm Số (Score-Based Attention) 87](#_Toc199064908)

[1.3. Mô Hình Attention Trong Sequence-to-Sequence 88](#_Toc199064909)

[2. Transformer 89](#_Toc199064910)

[2.1. Biểu diễn nguồn và đích (Source and Target Representation) 90](#_Toc199064911)

[2.2 Lớp Attention (Attention Layers) 91](#_Toc199064912)

[2.3 Residuals và Layer Normalization 95](#_Toc199064913)

[2.4 Positionwise Feed-forward Networks 96](#_Toc199064914)

[2.5 Bộ mã hóa (Encoder) 96](#_Toc199064915)

[2.6 Bộ giải mã (Decoder) 97](#_Toc199064916)

[VI. Federated Learning 97](#_Toc199064917)

[1. Tổng quan 97](#_Toc199064918)

[1.1 Giới thiệu 97](#_Toc199064919)

[1.2 Ví dụ chi tiết 98](#_Toc199064920)

[1.3 Ưu điểm của Federated Learning 98](#_Toc199064921)

[1.4 Nhược điểm của Federated Learning 98](#_Toc199064922)

[1.5 Các bước chính trong Federated Learning 99](#_Toc199064923)

[2. Phân loại 99](#_Toc199064924)

[2.1 Horizontal Federated Learning 99](#_Toc199064925)

[2.2 Vertical Federated Learning 100](#_Toc199064926)

[2.3 Federated Transfer Learning 101](#_Toc199064927)

[3. Các thành phần của Federated Learning 102](#_Toc199064928)

[3.1 Máy khách (Client) 102](#_Toc199064929)

[3.2. Máy chủ trung tâm (Central Server) 102](#_Toc199064930)

[3.3 Thu nhận mô hình toàn cục (Fetching Global Model) 103](#_Toc199064931)

[3.4 Huấn luyện mô hình cục bộ (Local Model Training) 103](#_Toc199064932)

[3.5 Tổng hợp mô hình (Model Aggregation) 103](#_Toc199064933)

[4. Kỹ thuật tổng hợp mô hình (Model Aggregation) 104](#_Toc199064934)

[4.1 All Model Averaging (AMA) - Trung bình cộng tất cả mô hình 106](#_Toc199064935)

[4.2 One Model Selection (OMS) - Lựa chọn một mô hình 110](#_Toc199064936)

[4.3 Best Model Average (BMA) - Trung bình cộng các mô hình tốt nhất 113](#_Toc199064937)

# **I. Giới thiệu và Tổng quan về Machine Learning**

## 1.Tìm hiểu về Machine Learning

Machine Learning là quá trình cho phép học các quy tắc (pattern) từ dữ liệu được cung cấp.

* Các quy tắc (pattern) được máy học có thể được sử dụng để đưa ra một số ước tính/dự đoán.

A diagram of data mining

Description automatically generated

Nguồn tham khảo: Baig M. Optimizing AI and Machine Learning Solutions. Your ultimate guide...2024

## 2. Xác định một bài toán sử dụng Machine Learning

Điều quan trọng cần chú ý là không phải tất cả các vấn đề đều cần được giải quyết bằng các bài toán Machine Learning.

Machine Learning đi kèm với những yêu cầu và thách thức riêng của nó.

* Một khối lượng lớn dữ liệu huấn luyện mà máy cần học để tạo ra mô hình.
* Tốn nhiều thời gian.

Điều kiện lý tưởng cho một giải pháp Machine Learning là một vấn đề mà bạn có đủ dữ liệu, đủ thời gian và yêu cầu độ chính xác cao hơn.

Nguồn tham khảo: Baig M. Optimizing AI and Machine Learning Solutions. Your ultimate guide...2024

## 3. Các kiểu học máy

Chúng ta có ba kiểu học máy chính.

A diagram of a diagram of a learning process

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| Các kiểu học máy | Học có giám sát  (Supervised Learning) | Học không giám sát  (Unsupervised Learning) | Học tăng cường  (Reinforcement Learning) |
| Định nghĩa | Máy học từ dữ liệu được cung cấp có nhãn. | Máy học từ dữ liệu được cung cấp không có nhãn. | Một tác nhân tương tác với môi trường bằng cách thực hiện các hành động và học hỏi từ các lỗi và phần thưởng. |
| Các loại bài toán | Phân loại (Classification), Hồi quy (Regression) | Phân cụm (Clustering), Giảm chiều dữ liệu (Dimensionality reduction) | Chính sách/hành động tối ưu (Optimal policy/actions) |
| Huấn luyện | Có sự giám sát bên ngoài | Không có sự giám sát | Không có sự giám sát |
| Cách tiếp cận | Giảm thiểu lỗi trên tập dữ liệu huấn luyện. | Nắm bắt các mẫu/cấu trúc trong dữ liệu. | Phương pháp thử và sai nhằm tối đa hóa kết quả. |

Ngoài ra, có nhiều quy tắc kinh nghiệm để lựa chọn cách định dạng cụ thể cho bài toán học máy đối với một nhiệm vụ/tình huống nhất định:

* Học có giám sát: Nếu giá trị mục tiêu có sẵn.
* Phân loại: Nếu giá trị mục tiêu là một danh mục (category).
* Hồi quy: Nếu giá trị mục tiêu là một đại lượng (quantity).
* Học không giám sát: Nếu nhãn mục tiêu không có sẵn.
* Phân cụm: Để nhóm các thực thể.
* Giảm chiều dữ liệu: Tinh lọc thông tin thành một tập hợp các đặc trưng (feature) nhỏ hơn.
* Học tăng cường: Để xác định hành động tốt nhất trong một tình huống.

Nguồn tham khảo: Baig M. Optimizing AI and Machine Learning Solutions. Your ultimate guide...2024

## 4. Kết hợp các kiểu học máy.

Nhược điểm của các phương pháp riêng lẻ:

* Phương pháp học có giám sát thuần túy gặp khó khăn với vấn đề mất cân bằng lớp nghiêm trọng.
* Phương pháp học không giám sát ít bị ảnh hưởng bởi vấn đề này nhưng học không giám sát có thể tạo ra các cụm nghi ngờ là điểm ngoại lai.
* Do đó, cả hai phương pháp đều có những nhược điểm riêng.

Giải pháp kết hợp:

* Để khắc phục những nhược điểm trên, chúng ta có thể kết hợp hai phương pháp.
* Cụ thể, kết quả gán cụm từ phương pháp học không giám sát được sử dụng làm biến dự báo trong dữ liệu huấn luyện cho mô hình phân loại (học có giám sát).
* Đặc trưng mới "Cụm” (Cluster) này cung cấp thông tin hữu ích, giúp mô hình học có giám sát hoạt động tốt hơn, ngay cả khi dữ liệu bị mất cân bằng lớp.
* Việc thêm đặc trưng này giúp giảm thiểu vấn đề các đặc trưng bị chi phối bởi thông tin từ lớp chiếm đa số.

Nguồn tham khảo: Baig M. Optimizing AI and Machine Learning Solutions. Your ultimate guide...2024

## 5. Huấn luyện mô hình

* Tổng quan về huấn luyện mô hình học có giám sát:

Dữ liệu huấn luyện được cung cấp cho mô hình. Mô hình sẽ tạo ra các dự đoán. Các dự đoán này được so sánh với các giá trị y thực tế từ tập dữ liệu nhãn. Mục tiêu của học có giám sát là so sánh kết quả đầu ra của mô hình với giá trị thực tế, từ đó điều chỉnh các tham số. Quá trình lặp lại bao gồm việc phân tích sai lệch và điều chỉnh mô hình, được gọi là huấn luyện mô hình.

A black square with white text

AI-generated content may be incorrect.

* Để đánh giá hiệu quả của mô hình, chúng ta chia toàn bộ tập dữ liệu thành ba phần:
* Tập dữ liệu huấn luyện (Training data set)
* Tập dữ liệu này được sử dụng để huấn luyện mô hình.
* Mô hình sẽ tạo ra các dự đoán dựa trên dữ liệu huấn luyện.
* Chúng ta tính toán "loss" giữa các dự đoán và giá trị thực tế.
* "Loss" là một đại lượng số học thể hiện sự khác biệt giữa dự đoán và thực tế.
* Dựa trên "loss", chúng ta điều chỉnh các tham số của mô hình. Quá trình này được gọi là "huấn luyện".

A black square with white text

AI-generated content may be incorrect.

* Tập dữ liệu kiểm chứng (Validation data set)
* Tập dữ liệu này được sử dụng để kiểm tra hiệu suất của mô hình sau quá trình huấn luyện.
* Sau mỗi lần huấn luyện, chúng ta đưa tập dữ liệu kiểm chứng vào mô hình để tính toán "loss".
* "Loss" trên tập dữ liệu kiểm chứng giúp chúng ta đánh giá khả năng tổng quát hóa của mô hình đối với dữ liệu mới.
* Điểm quan trọng: "Loss" từ tập dữ liệu kiểm chứng không được sử dụng để điều chỉnh trực tiếp các tham số của mô hình. Điều này khác với quá trình huấn luyện.

A black square with white text

AI-generated content may be incorrect.

* Tập dữ liệu kiểm thử (Testing data set)
* Sau khi hoàn thành quá trình huấn luyện và lựa chọn mô hình tốt nhất chúng ta sử dụng tập dữ liệu kiểm thử để đánh giá hiệu suất cuối cùng.
* Tập dữ liệu kiểm thử giúp xác định khả năng tổng quát hóa thực sự của mô hình đã chọn.
* Tập dữ liệu kiểm thử là bước kiểm tra cuối cùng, để xác định xem mô hình có hoạt động tốt trên dữ liệu mới hay không.

A black square with white text

AI-generated content may be incorrect.

Nguồn tham khảo: Machine Learning for Everybody – Full Course – freeCodeCamp.org

## 6. Các mô hình học máy

Một vài mô hình học máy cơ bản

* K-Nearest Neighbors
* Naive Bayes
* Linear Regression
* Logistic Regression
* Support Vector Machine
* Neural Network

Nguồn tham khảo: Machine Learning for Everybody – Full Course – freeCodeCamp.org

# **II. Các Thuật toán Machine Learning Cơ bản**

## 1. K-Nearest Neighbors (KNN)

Nguồn tham khảo**:**

* KNN Algorithm In Machine Learning - Simplilearn
* K-nearest neighbors - StatQuest

a) Thuật toán KNN

KNN - K Nearest Neighbors, là một trong những thuật toán Học máy có giám sát đơn giản nhất.

Nó phân loại một điểm dữ liệu dựa trên cách các láng giềng của nó được phân loại.

K trong KNN là một tham số đề cập đến số lượng láng giềng gần nhất được đưa vào quá trình đánh giá để phân loại.

A diagram of a diagram of a wine diagram

AI-generated content may be incorrect.

b) Cách lựa chọn giá trị "K"

Không có cách thức nào để xác định giá trị tốt nhất cho 'K', vì vậy có thể phải thử nghiệm một vài giá trị trước khi chọn một giá trị phù hợp

* Các giá trị K nhỏ (như K=1 hoặc K=2) có thể bị nhiễu và chịu ảnh hưởng của các điểm ngoại lai.
* Các giá trị K lớn giảm tác động của nhiễu và ngoại lai, nhưng nếu "K" quá lớn, các danh mục (category) có ít mẫu có thể bị áp đảo bởi các danh mục có nhiều mẫu hơn.

Các gợi ý để lựa chọn giá trị "K":

* Căn bậc hai của "n" với n là tổng số điểm dữ liệu trong tập dữ liệu.
* Chọn "K" là số lẻ giúp tránh nhầm lẫn khi phân loại giữa hai lớp dữ liệu.

c) Khi nào chúng ta sử dụng thuật toán KNN

Thuật toán KNN được ứng dụng trong các trường hợp sau:

* Dữ liệu được gán nhãn.
* Dữ liệu không có nhiễu.
* Tập dữ liệu có kích thước nhỏ

Lý do là vì KNN không học một hàm phân biệt từ tập dữ liệu huấn luyện.

d) Các bước của thuật toán K-Nearest Neighbors (KNN)

Thuật toán KNN bao gồm 5 bước sau:

* Giả sử D là tập hợp các điểm dữ liệu đã được gán nhãn và A là điểm dữ liệu chưa được gán nhãn cần phân loại.
* Tính toán khoảng cách từ điểm dữ liệu mới A đến tất cả các điểm dữ liệu đã được phân loại trong D.
* Chọn K (một tham số được xác định trước) điểm dữ liệu có khoảng cách nhỏ nhất.
* Kiểm tra nhãn lớp của K láng giềng gần nhất và đếm số lần xuất hiện của mỗi lớp.
* Gán lớp xuất hiện nhiều nhất trong số K láng giềng.

## 2. Linear Regression

Nguồn tham khảo**:**

* Linear Regression in Python - Machine Learning From Scratch - Patrick Loeber
* Linear Regression – StatQuest

a) Giới thiệu về Linear Regression

Linear Regression là một thuật toán học máy có giám sát được sử dụng để dự đoán các giá trị liên tục. Khác với phân loại, nơi mà mục tiêu là dự đoán các nhãn lớp rời rạc, Linear Regression hướng đến việc tìm ra mối quan hệ tuyến tính giữa các biến đầu vào và biến đầu ra.

Ví dụ:

* Cho một tập dữ liệu gồm các điểm (x, y) trên mặt phẳng, mục tiêu là tìm một đường thẳng (hàm tuyến tính) xấp xỉ tốt nhất các điểm này.
* Hàm tuyến tính được biểu diễn dưới dạng
* : Giá trị dự đoán.
* : Trọng số (độ dốc của đường thẳng).
* : Biến đầu vào.
* : Bias (giao điểm của đường thẳng với trục y).

A diagram of a graph

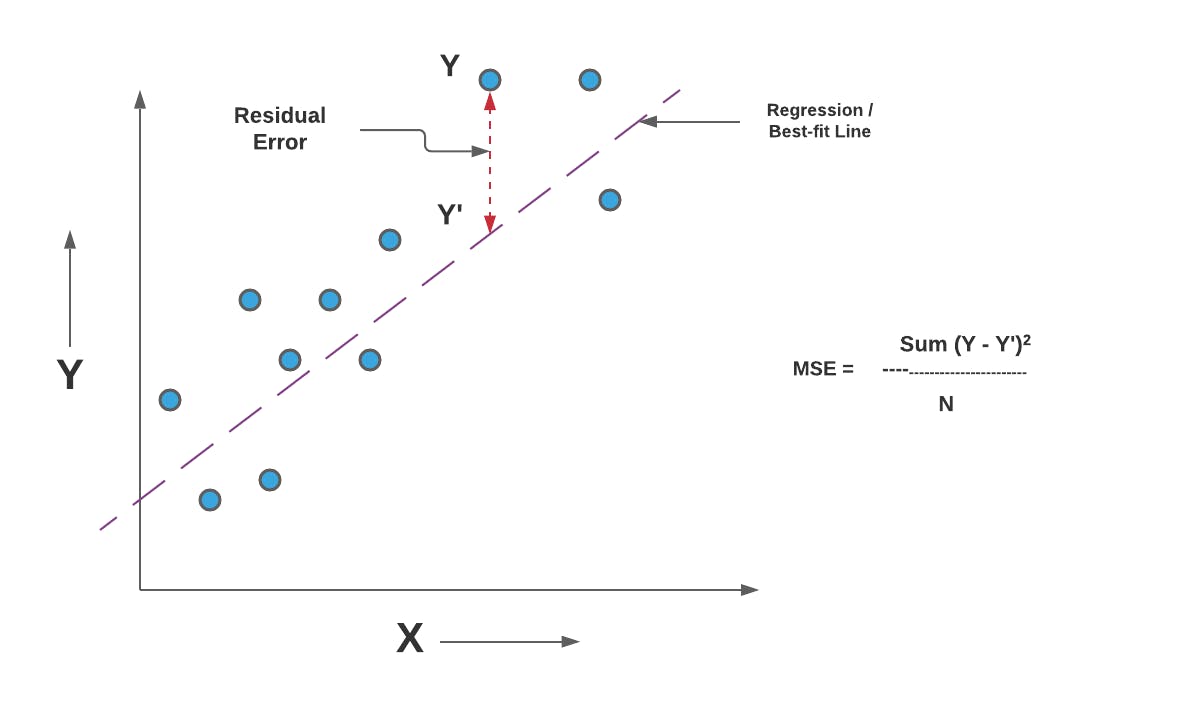
AI-generated content may be incorrect.

b) Hàm chi phí (Cost fuction)

Mục tiêu của hồi quy tuyến tính là tìm ra các giá trị tối ưu cho và sao cho đường thẳng xấp xỉ dữ liệu một cách chính xác nhất. Để đánh giá mức độ chính xác, chúng ta sử dụng hàm chi phí, thường là Mean Square Error (MSE):

* : Số lượng mẫu dữ liệu.
* : Giá trị thực tế.
* : Giá trị dự đoán.

Mục tiêu là giảm thiểu MSE, tức là tìm và sao cho sai số giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất.



c) Gradient Descend

Để tìm giá trị nhỏ nhất của hàm chi phí, chúng ta sử dụng phương pháp gradient descent. Đây là một phương pháp lặp để tối ưu hóa các tham số của mô hình.

Tính toán Gradient: Tính đạo hàm của hàm chi phí theo () và theo ().

Cập nhật Tham số:

* Cập nhật và theo hướng ngược lại của gradient để giảm thiểu chi phí.
* Công thức cập nhật:
* : Tốc độ học (learning rate), kiểm soát kích thước bước cập nhật.

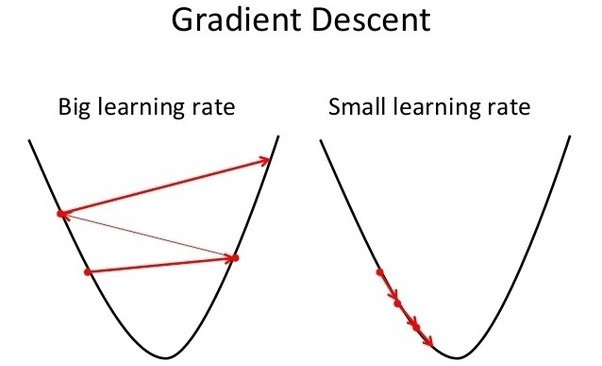
A diagram of a weight loss

AI-generated content may be incorrect.

d) Learning rate

Tốc độ học là một tham số quan trọng ảnh hưởng đến hiệu suất của mô hình.

* Tốc độ học nhỏ: Quá trình huấn luyện diễn ra chậm, nhưng có thể tìm được giá trị tối ưu.
* Tốc độ học lớn: Quá trình huấn luyện diễn ra nhanh, nhưng có thể không ổn định, dẫn đến việc bỏ qua giá trị tối ưu.
* Cần lựa chọn tốc độ học phù hợp để cân bằng giữa tốc độ huấn luyện và độ chính xác của mô hình.



## 3. Logistic Regression

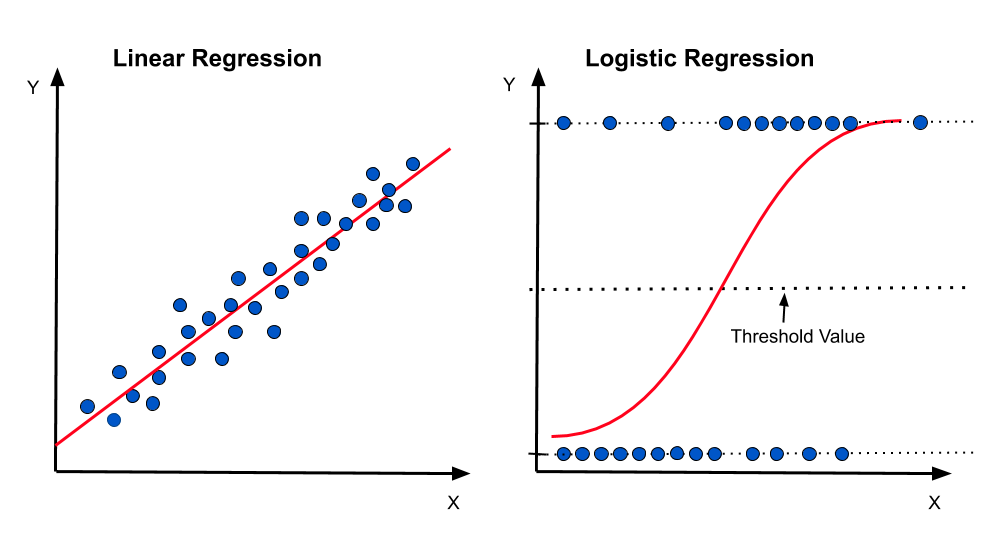
Nguồn tham khảo**:**

* Logistic Regression in Python - Machine Learning From Scratch - Patrick Loeber
* Logistic Regression – StatQuest

a) Giới thiệu về Logistic Regression

Hồi quy logistic là một thuật toán học máy có giám sát được sử dụng cho các bài toán phân loại nhị phân. Mặc dù có tên gọi là "hồi quy", thuật toán này thực chất được dùng để dự đoán xác suất một mẫu dữ liệu thuộc về một trong hai lớp.

* : Giá trị dự đoán.
* : Trọng số (độ dốc của đường thẳng).
* : Biến đầu vào.
* : Bias.

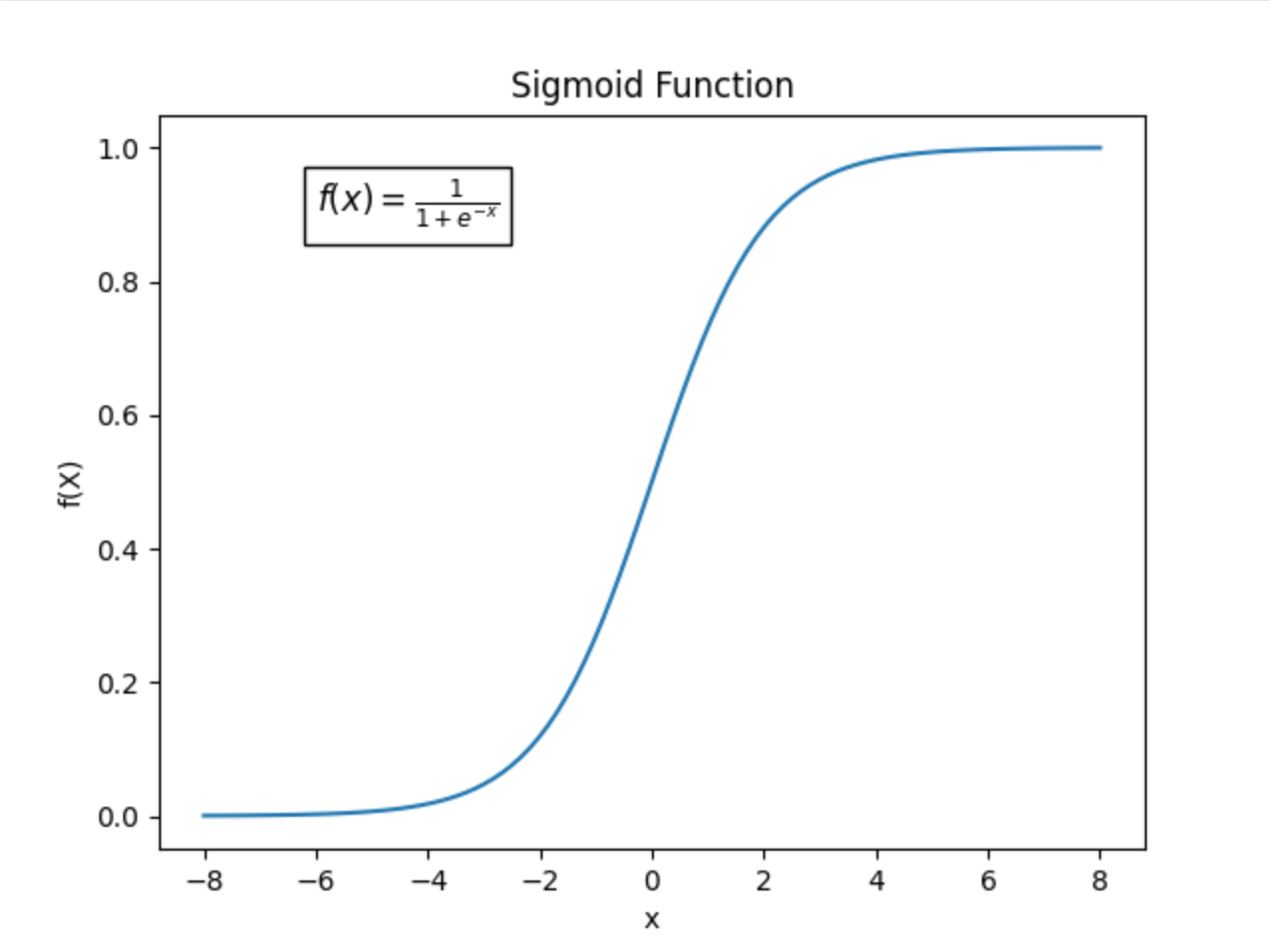


b) Hàm Sigmoid

Khác với hồi quy tuyến tính, dự đoán các giá trị liên tục, hồi quy logistic dự đoán xác suất, nằm trong khoảng từ 0 đến 1.

Để chuyển đổi kết quả đầu ra của mô hình tuyến tính thành xác suất, chúng ta sử dụng hàm sigmoid:

Hàm sigmoid tạo ra một đường cong hình chữ S, giới hạn kết quả đầu ra trong khoảng từ 0 đến 1, biểu diễn xác suất.



c) Loss function

Để tìm các giá trị tối ưu cho và , chúng ta sử dụng hàm chi phí cross-entropy.

Hàm chi phí cross-entropy đo lường sự khác biệt giữa xác suất dự đoán và nhãn lớp thực tế.

Mục tiêu là giảm thiểu hàm chi phí, tìm W và B sao cho xác suất dự đoán gần với nhãn lớp thực tế nhất.

Phương pháp gradient descent được sử dụng để tối ưu hóa các tham số:

* Tính toán gradient (đạo hàm) của hàm chi phí theo và .
* Cập nhật và theo hướng ngược lại của gradient để giảm thiểu chi phí.
* Tốc độ học (learning rate) là tham số quan trọng, kiểm soát kích thước bước cập nhật.
* Quy tắc cập nhật

: Tốc độ học (learning rate)

## 4. Naive Bayes

Nguồn tham khảo**:**

* Naive Bayes in Python - Machine Learning From Scratch

a) Giới thiệu về Naive Bayes

Naive Bayes là một thuật toán học máy có giám sát được sử dụng cho các bài toán phân loại. Thuật toán này dựa trên Định lý Bayes, một nguyên tắc xác suất cơ bản mô tả mối quan hệ giữa các xác suất có điều kiện.

Định lý Bayes:

Trong ngữ cảnh phân loại:

: Lớp (class).

: Véc-tơ đặc trưng (feature vector).

b) Giả định của Naive Bayes

Tên gọi "Naive" xuất phát từ giả định rằng tất cả các đặc trưng đầu vào đều độc lập với nhau. Điều này có nghĩa là sự xuất hiện của một đặc trưng không ảnh hưởng đến sự xuất hiện của bất kỳ đặc trưng nào khác.

* Ví dụ: Trong bài toán dự đoán xem một người có đi chạy bộ hay không, các đặc trưng "trời nắng" và "sức khỏe tốt" được giả định là độc lập.
* Trong thực tế, giả định này thường không đúng, nhưng Naive Bayes vẫn hoạt động hiệu quả trong nhiều trường hợp.

c) Công thức và giải thích

* : Xác suất hậu nghiệm (posterior probability), xác suất một mẫu dữ liệu thuộc lớp khi biết đặc trưng .
* : Xác suất điều kiện lớp (class conditional probability), xác suất xuất hiện đặc trưng khi biết lớp Y.
* : Xác suất tiên nghiệm của lớp (prior probability of ), xác suất xuất hiện lớp trong tập dữ liệu.
* : Xác suất tiên nghiệm của đặc trưng (prior probability of ).

Mục tiêu của Naive Bayes là tìm lớp Y có xác suất hậu nghiệm lớn nhất.

Vì P(X) là hằng số đối với tất cả các lớp, ta có thể bỏ qua nó trong quá trình so sánh.

Để tránh vấn đề tràn số do nhân nhiều xác suất nhỏ, ta thường sử dụng logarit.

d) Tính toán xác suất

Xác suất tiên nghiệm được tính bằng tần suất xuất hiện của lớp trong tập dữ liệu huấn luyện.

Xác suất điều kiện lớp thường được mô hình hóa bằng phân phối Gaussian (phân phối chuẩn).

* : Phương sai của đặc trưng trong lớp .
* : Kì vọng của đặc trưng trong lớp .

## 5. Support Vector Machine

Nguồn tham khảo**:**

* Support Vector Machine in Python - Machine Learning From Scratch

a) Giới thiệu về Support Vector Machine (SVM)

Support Vector Machine (SVM) là một thuật toán học máy có giám sát mạnh mẽ, được sử dụng rộng rãi cho cả các bài toán phân loại và hồi quy. Trong lĩnh vực phân loại, SVM đặc biệt hiệu quả trong việc tìm ra ranh giới quyết định tối ưu để phân tách các lớp dữ liệu.

b) Ranh giới quyết định tuyến tính và Hyperplane

SVM hoạt động dựa trên ý tưởng sử dụng một mô hình tuyến tính để xác định ranh giới quyết định. Ranh giới quyết định này, trong không gian n chiều (n là số chiều của dữ liệu), được gọi là hyperplane.

c) Hyperplane tối ưu và Margin

Mục tiêu của SVM là tìm ra hyperplane "tốt nhất" để phân tách các lớp dữ liệu. "Tốt nhất" ở đây được định nghĩa là hyperplane tạo ra sự phân tách lớn nhất giữa các lớp, hay nói cách khác, có "margin" lớn nhất.

* Margin**:** Khoảng cách giữa hyperplane và các điểm dữ liệu gần nhất của mỗi lớp. Các điểm dữ liệu này được gọi là "vector hỗ trợ" (support vectors).
* SVM tìm cách tối đa hóa margin, tức là tìm hyperplane sao cho khoảng cách đến các vector hỗ trợ của mỗi lớp là lớn nhất.

A diagram of a mathematical equation

AI-generated content may be incorrect.

d) Biểu diễn toán học của Hyperplane và Margin

* Phương trình Hyperplane:
* : Véc-tơ trọng số, xác định hướng của hyperplane.
* : Véc-tơ đặc trưng của một điểm dữ liệu.
* : Bias, xác định vị trí của hyperplane.
* Điều kiện phân loại:
  + cho các điểm thuộc lớp .
  + cho các điểm thuộc lớp .
* Kết hợp điều kiện:
  + : Nhãn lớp, nhận giá trị hoặc .

e) Hàm chi phí (Cost Function)

Để tìm ra các giá trị tối ưu cho và , SVM sử dụng hàm chi phí và phương pháp tối ưu hóa.

* Hàm Chi phí Hinge Loss:
  + Hinge Loss đo lường "mức độ vi phạm" của một điểm dữ liệu so với điều kiện phân loại.
  + Hinge Loss bằng 0 nếu điểm dữ liệu được phân loại đúng và nằm ngoài margin.
* Tối ưu hóa Margin: Để tối đa hóa margin, SVM cần cực tiểu hóa độ lớn của véc-tơ trọng số ().
* Hàm Chi phí Tổng quát:
  + : Tham số điều chỉnh, cân bằng giữa mục tiêu phân loại đúng và tối đa hóa margin.

f) Phương pháp Gradient Descent

SVM sử dụng phương pháp gradient descent để tìm ra các giá trị của và cực tiểu hóa hàm chi phí.

* Đạo hàm của Hàm Chi phí:
  + Trường hợp 1: (Hinge Loss = 0)
  + Trường hợp 2:
* Cập nhật Tham số:
  + : Tốc độ học (learning rate), kiểm soát kích thước bước cập nhật.

# **III. Mạng Neural (Neural Networks)**

## 1.Tổng quan về Neural Network

Source:

* Machine Learning cơ bản – Vũ Hữu Tiệp
* Deep Learning cơ bản – Nguyễn Thanh Tuấn

### 1.1 Giới thiệu về Neural Network

Neural Network (NN) là một mô hình tính toán lấy cảm hứng từ cách bộ não con người hoạt động. NN bao gồm nhiều neuron nhân tạo kết nối với nhau theo một cấu trúc phân tầng và được sử dụng để nhận diện mẫu, phân loại dữ liệu, dự đoán xu hướng, và nhiều ứng dụng khác.

Mạng nơ-ron có khả năng học từ dữ liệu bằng cách điều chỉnh các trọng số và bias của các kết nối giữa các neuron để tối ưu hóa đầu ra của mô hình.

### 1.2 Kiến trúc của mạng Neuron

1.2.1 Mô hình tổng quát

A diagram of a network

AI-generated content may be incorrect.

* Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Các hình tròn được gọi là node.
* Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer - 1 (không tính input layer).
* Ví dụ như ở hình trên có 1 input layer, 2 hidden layer và 1 output layer. Số lượng layer của mô hình là 3 layer.
* Mỗi node trong hidden layer và output layer:
* Liên kết với tất cả các node ở layer trước đó với các hệ số riêng.
* Mỗi node có 1 hệ số bias riêng.
* Diễn ra 2 bước: tính tổng linear và áp dụng activation function.

Một mạng nơ-ron điển hình bao gồm các lớp sau:

* Lớp đầu vào: Chứa các nơ-ron đầu vào, nhận dữ liệu đầu vào từ thế giới bên ngoài.
* Lớp ẩn: Một hoặc nhiều lớp nằm giữa lớp đầu vào và lớp đầu ra. Các nơ-ron trong lớp ẩn thực hiện các phép tính trung gian để trích xuất các đặc trưng phức tạp từ dữ liệu đầu vào.
* Lớp đầu ra: Chứa các nơ-ron đầu ra, tạo ra kết quả cuối cùng của mạng.

Mạng nơ-ron trong đó thông tin chỉ truyền theo một hướng từ lớp đầu vào đến lớp đầu ra được gọi là feedforward neural networks.

1.2.2 Công thức toán học

Mỗi nơ-ron thực hiện hai bước chính:

1. Tính tổng có trọng số của đầu vào

Trong đó:

* là ma trận trọng số giữa lớp và lớp .
* là đầu ra của lớp trước
* là bias của lớp hiện tại

- Bias

* Bias là một hệ số trong Neural Network, thường được ký hiệu là b, có vai trò tương tự như hệ số tung độ gốc trong phương trình đường thẳng.
* Bias cho phép đường thẳng (hoặc siêu phẳng trong không gian nhiều chiều) dịch chuyển, giúp mô hình biểu diễn các mối quan hệ phức tạp hơn trong dữ liệu. Nếu không có bias, đường thẳng sẽ luôn đi qua gốc tọa độ, làm giảm khả năng tổng quát hóa của mô hình.

- Hệ số Weight

* Weight, thường ký hiệu là w hoặc W, là các hệ số nhân với các giá trị đầu vào để xác định mức độ ảnh hưởng của từng đầu vào đến đầu ra của một nơ-ron.
* Mỗi liên kết giữa các nơ-ron trong mạng Neural Network có một weight tương ứng. Các weights thường được tổ chức thành các ma trận.

1. Áp dụng hàm kích hoạt

Trong đó là activation function, giúp mô hình học các đặc trưng phi tuyến

### 1.3 Activation Function

1.3.1 Tổng quan về Activation Function

Activation function là một hàm toán học được áp dụng lên đầu ra của một nơ-ron sau khi đã tính tổng tuyến tính (tổng của các đầu vào nhân với weights và cộng với bias).

Activation function tạo ra tính phi tuyến cho mô hình Neural Network, cho phép nó học được các mối quan hệ phức tạp trong dữ liệu. Nếu không có activation function phi tuyến, mạng Neural Network sẽ chỉ có thể biểu diễn các hàm tuyến tính, bất kể có bao nhiêu lớp.

1.3.2 Các loại Activation Function phổ biến

**-** Hàm Sigmoid: (đầu ra từ 0 đến 1)

**A graph of a function

AI-generated content may be incorrect.**

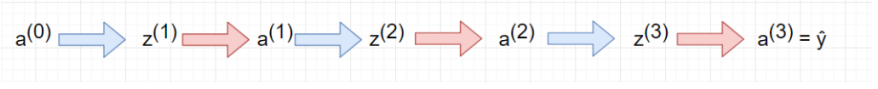
* Hàm Tanh: (đầu ra từ -1 đến 1, có tính zero-centered)

A graph of a function

AI-generated content may be incorrect.

* Hàm ReLU (Rectified Linear Unit): (đơn giản, tính toán nhanh, giúp huấn luyện mạng sâu hiệu quả)

### 1.4 Quá trình Feedforward



Feedforward là quá trình truyền dữ liệu từ đầu vào đến đầu ra thông qua các lớp nơ-ron.

* Nhận dữ liệu đầu vào từ input layer.
* Tính toán từng tầng theo công thức:
* Xuất kết quả dự đoán từ output layer.

### 1.5 Thuật toán Backpropagation

A math equations on a grid

AI-generated content may be incorrect.

- Backpropagation là thuật toán giúp tối ưu trọng số bằng cách cập nhật dựa trên đạo hàm của Loss Function.

- Backpropagation là thuật toán lan truyền ngược để cập nhật trọng số bằng cách tính toán gradient của Loss Function với từng trọng số.

- Các bước trong Backpropagation

* Feedforward: Tính đầu ra của mỗi nơ-ron.
* Tính lỗi ở đầu ra: So sánh với nhãn thực tế bằng Loss Function.
* Tính Gradient của Loss Function: Dùng đạo hàm chuỗi để lan truyền lỗi ngược về các tầng trước.
* Cập nhật trọng số bằng Gradient Descent.

## 2. Gradient Descent và Backpropagation

### 2.1 Gradient Descent

Mục tiêu của việc học trong mạng nơ-ron là tìm ra các giá trị tối ưu cho trọng số và bias sao cho mạng có thể thực hiện tốt một nhiệm vụ cụ thể (ví dụ: phân loại hình ảnh). Để đạt được điều này, chúng ta sử dụng một hàm chi phí để đo lường hiệu suất của mạng. Hàm chi phí thường được định nghĩa sao cho nó có giá trị nhỏ khi mạng hoạt động tốt và giá trị lớn khi mạng hoạt động kém.

1. Hàm Chi phí

Một hàm chi phí phổ biến là hàm chi phí quadratic (hay còn gọi là mean squared error - MSE):

Trong đó:

* là tập hợp tất cả các trọng số trong mạng.
* là tập hợp tất cả các bias trong mạng.
* là số lượng mẫu huấn luyện.
* là một mẫu huấn luyện đầu vào.
* là đầu ra mong muốn cho đầu vào x.
* là đầu ra thực tế của mạng cho đầu vào x.

Mục tiêu là tìm các giá trị của và để giảm thiểu hàm chi phí .

1. Thuật toán Gradient Descent

Gradient descent là một thuật toán tối ưu hóa được sử dụng để tìm cực tiểu của một hàm số. Trong bối cảnh mạng nơ-ron, chúng ta sử dụng gradient descent để tìm các giá trị của trọng số và bias giúp giảm thiểu hàm chi phí.

Ý tưởng cơ bản của gradient descent là di chuyển các tham số (trọng số và bias) theo hướng ngược lại với gradient của hàm chi phí. Gradient cho biết hướng mà hàm chi phí tăng nhanh nhất. Bằng cách di chuyển theo hướng ngược lại, chúng ta hy vọng sẽ tiến gần hơn đến điểm cực tiểu của hàm chi phí.

Quy tắc cập nhật cho trọng số () và bias () trong thuật toán gradient descent là:

Trong đó là tốc độ học (learning rate), một tham số dương nhỏ quyết định kích thước của mỗi bước di chuyển.

1. Stochastic Gradient Descent

Khi tập dữ liệu huấn luyện rất lớn, việc tính toán gradient của hàm chi phí trên toàn bộ dữ liệu có thể tốn rất nhiều thời gian. Stochastic gradient descent (SGD) là một biến thể của gradient descent giúp tăng tốc độ học bằng cách ước tính gradient dựa trên một mẫu nhỏ ngẫu nhiên của dữ liệu huấn luyện, được gọi là mini-batch.

Thay vì tính toán gradient trên toàn bộ mẫu huấn luyện, SGD chỉ tính toán gradient trên một mini-batch gồm mẫu (). Điều này giúp giảm đáng kể thời gian tính toán cho mỗi bước cập nhật.

Để kết nối quá trình học trong mạng nơ-ron với phương pháp giảm dần gradient, giả sử ​ và ​ lần lượt là các trọng số và bias trong mạng nơ-ron. Phương pháp stochastic gradient descent hoạt động bằng cách chọn ra một mini-batch ngẫu nhiên từ các đầu vào huấn luyện, sau đó tiến hành huấn luyện với các dữ liệu này.

Trong đó, tổng số là trên tất cả các ví dụ huấn luyện ​ trong mini-batch hiện tại. Sau khi hoàn thành việc huấn luyện với một mini-batch, một mini-batch khác sẽ được chọn ngẫu nhiên và tiếp tục huấn luyện. Quá trình này sẽ được lặp lại cho đến khi tất cả các ví dụ huấn luyện đã được sử dụng, lúc đó ta hoàn thành một "epoch" huấn luyện. Sau mỗi epoch, quá trình huấn luyện sẽ bắt đầu lại với một bộ dữ liệu huấn luyện mới.

### 2.2 Backpropagation

Backpropagation là một thuật toán hiệu quả để tính toán gradient của hàm chi phí đối với tất cả các trọng số và bias trong mạng nơ-ron. Đây là một thành phần cốt lõi của hầu hết các thuật toán học sâu.

* 1. Ký hiệu Ma trận

Để mô tả thuật toán backpropagation một cách hiệu quả, chúng ta sử dụng ký hiệu ma trận cho trọng số, bias và activation của các nơ-ron trong mạng.

* : Trọng số kết nối từ nơ-ron thứ ở lớp đến nơ-ron thứ ở lớp .
* : Bias của nơ-ron thứ ở lớp .
* : Activation của nơ-ron thứ ở lớp .
  1. Công thức Activation dưới dạng Ma trận

Activation của các nơ-ron ở lớp l có thể được tính toán từ các activations ở lớp l-1 bằng công thức ma trận:

Trong đó:

* , và là các vectơ cột chứa activations và biases của các lớp tương ứng.
* là ma trận trọng số kết nối lớp với lớp .
* là hàm sigmoid được áp dụng theo phần tử cho vectơ đầu vào.

Đại lượng được gọi là weighted input.

* 1. Lỗi của Nơ-ron

Trong thuật toán backpropagation, chúng ta định nghĩa lỗi () của một nơ-ron ở lớp có công thức như sau:

* 1. Bốn Phương trình cơ bản của Backpropagation

Thuật toán backpropagation dựa trên bốn phương trình cơ bản sau:

* Lỗi ở lớp đầu ra:

Trong đó là vectơ các đạo hàm riêng của hàm chi phí đối với các activations ở lớp đầu ra, và là tích Hadamard (tích theo phần tử).

* Lỗi ở lớp theo lỗi ở lớp :
* Tốc độ thay đổi của chi phí đối với bias:
* Tốc độ thay đổi của chi phí đối với trọng số:
  1. Thuật toán Backpropagation

Thuật toán backpropagation có thể được tóm tắt như sau:

1. Feedforward: Tính toán activations của tất cả các lớp trong mạng cho một đầu vào cụ thể.
2. Tính lỗi ở lớp đầu ra: Sử dụng Phương trình (BP1).
3. Lan truyền ngược lỗi: Tính toán lỗi cho các lớp trước đó bằng cách sử dụng Phương trình (BP2) lặp đi lặp lại từ lớp đầu ra trở về lớp thứ hai.
4. Tính gradient: Tính toán đạo hàm riêng của hàm chi phí đối với trọng số và bias bằng cách sử dụng Phương trình (BP3) và (BP4).
   1. Ảnh hưởng của Activation và Sự Bão hòa

Các phương trình backpropagation cho thấy rằng tốc độ học của một trọng số phụ thuộc vào activation của nơ-ron đầu vào và đạo hàm của hàm sigmoid tại nơ-ron đầu ra. Nếu activation của nơ-ron đầu vào gần 0, hoặc nếu nơ-ron đầu ra bị bão hòa (activation gần 0 hoặc 1), thì trọng số sẽ học chậm.

## 3. Hiện tượng Overfitting và cách khắc phục (L1, L2 Regularization)

### 3.1 Hiện tượng Overfitting

Nguồn tham khảo: Neural Networks and Deep Learning - Michael Nielsen

a) Giới thiệu về Overfitting

Overfitting là hiện tượng khi mô hình học quá chi tiết và cụ thể từ dữ liệu huấn luyện mà không thể tổng quát hóa tốt đối với dữ liệu kiểm tra. Khi xảy ra overfitting, mô hình có thể đạt độ chính xác rất cao trên dữ liệu huấn luyện nhưng lại không hiệu quả khi áp dụng trên dữ liệu mới, chưa thấy.

Trong mạng nơ-ron, overfitting thường xảy ra khi mô hình có quá nhiều tham số và được huấn luyện với một bộ dữ liệu huấn luyện nhỏ. Mô hình có thể học thuộc các đặc điểm ngẫu nhiên trong dữ liệu huấn luyện, từ đó không còn khả năng tổng quát cho các dữ liệu mới.

b) Dấu hiệu của Overfitting

Một dấu hiệu rõ ràng của overfitting là khi độ chính xác trên dữ liệu huấn luyện (training data) tăng rất cao, có thể đạt đến 100%, trong khi độ chính xác trên dữ liệu kiểm tra (validation data) lại không cải thiện hoặc chỉ đạt mức rất thấp.

Nếu độ chính xác trên dữ liệu huấn luyện (training data) tiếp tục tăng mà không có sự cải thiện tương ứng trên dữ liệu kiểm tra, mô hình có thể đang học những chi tiết ngẫu nhiên không có ích.

Ví dụ, khi sử dụng 1.000 hình ảnh huấn luyện, mô hình có thể đạt độ chính xác 100% trên dữ liệu huấn luyện, nhưng chỉ đạt khoảng 82.27% trên dữ liệu kiểm tra, cho thấy mô hình chỉ học thuộc lòng dữ liệu huấn luyện mà không tổng quát hóa được.

c) Phát hiện Overfitting

Một cách để phát hiện overfitting là theo dõi độ chính xác trên dữ liệu kiểm tra (validation data) trong suốt quá trình huấn luyện. Nếu độ chính xác trên dữ liệu kiểm tra không còn cải thiện hoặc thậm chí giảm sau một thời gian, đó là dấu hiệu cho thấy mô hình đang overfit.

d) Giải quyết Overfitting bằng cách tăng kích thước dữ liệu huấn luyện

Một trong những phương pháp hiệu quả để giảm thiểu overfitting là tăng kích thước dữ liệu huấn luyện. Khi mô hình được huấn luyện trên một tập dữ liệu đủ lớn và đa dạng, nó có thể tổng quát tốt hơn và giảm khả năng overfitting.

Mặc dù tăng kích thước dữ liệu huấn luyện là một chiến lược hiệu quả để giảm thiểu overfitting, nhưng trong thực tế, việc thu thập dữ liệu huấn luyện đủ lớn có thể rất tốn kém và khó thực hiện. Do đó, không phải lúc nào cũng khả thi để thu thập một lượng dữ liệu lớn.

e) Các chiến lược khác để giảm thiểu Overfitting

Ngoài việc tăng kích thước dữ liệu huấn luyện, một số kỹ thuật cũng có thể giúp giảm thiểu overfitting và cải thiện khả năng tổng quát của mô hình như:

* Regularization (điều chỉnh trọng số của mạng)
* Early stopping (ngừng huấn luyện khi độ chính xác trên dữ liệu kiểm tra không còn cải thiện)

### 3.2 L2 Regulation

Nguồn tham khảo: Neural Networks and Deep Learning - Michael Nielsen

a) Giới Thiệu về Regularization

Một cách để giảm thiểu overfitting trong mạng nơ-ron là sử dụng các kỹ thuật Regularization. Trong khi việc tăng kích thước dữ liệu huấn luyện là một giải pháp, thì việc giảm kích thước mạng hoặc sử dụng các kỹ thuật regularization có thể giúp giảm overfitting ngay cả khi mạng và dữ liệu huấn luyện đã được cố định.

b) L2 Regularization (Weight Decay)

Một trong những kỹ thuật regularization phổ biến là L2 Regularization, còn được gọi là Weight decay. Ý tưởng của L2 regularization là thêm một đại lượng bổ sung vào hàm chi phí, được gọi là , nhằm ngăn chặn các trọng số trở nên quá lớn. Điều này được thực hiện bằng cách thêm vào tổng bình phương của tất cả các trọng số trong mạng.

Hàm chi phí Cross-entropy có Regularization được tính như sau:

Trong đó:

* + là tham số regularization, điều khiển mức độ ảnh hưởng của Regularization.
  + là kích thước của tập huấn luyện.
  + là tổng các bình phương trọng số trong mạng, giúp hạn chế giá trị của các trọng số.

Regularization có thể được áp dụng cho các hàm chi phí khác ngoài Cross-entropy, như hàm Mean Squared Error. Công thức regularization cho hàm Mean Squared Error được viết như sau:

Trong cả hai trường hợp, ta có thể viết lại hàm chi phí với Regularization là:

Trong đó là hàm chi phí chưa áp dụng Regularization.

c) Lợi ích của Regularization

Việc áp dụng regularization giúp giảm overfitting bằng cách ngăn các trọng số trở nên quá lớn và giúp cho mô hình học được các đặc điểm tổng quát thay vì học quá chi tiết từ dữ liệu huấn luyện.

Regularization giúp giảm bớt số lượng nơ-ron trong các lớp ẩn, hoặc loại bỏ ảnh hưởng của một số nơ-ron, từ đó làm giảm độ phức tạp và tăng tính tuyến tính của mô hình, giúp mô hình tổng quát tốt hơn.

Khi áp dụng L2 regularization, trọng số sẽ được rescale bằng cách nhân với một đại lượng . Điều này có tên gọi là weight decay, giúp làm giảm giá trị của các trọng số theo thời gian trong quá trình huấn luyện.

Ví dụ: Trong ví dụ hồi quy, nếu mô hình quá phức tạp với nhiều tham số, nó sẽ khớp chính xác với từng điểm dữ liệu trong tập huấn luyện. Tuy nhiên, điều này không giúp mô hình tổng quát tốt khi gặp dữ liệu mới. Thay vào đó, một mô hình đơn giản, với ít tham số hơn, sẽ tạo ra một đường cong mượt mà hơn và tổng quát tốt hơn.

d) Áp dụng Regularization trong Gradient Descent

Khi áp dụng L2 regularization trong thuật toán gradient descent, ta sẽ tính toán các đạo hàm từng trọng số và cập nhật chúng theo công thức:

Trong đó, là hàm chi phí không có regularization. Đối với trọng số, ta sẽ cập nhật theo công thức:

Đây chính là công thức cập nhật với weight decay.

Đối với bias, ta sẽ cập nhật theo công thức:

e) Stochastic Gradient Descent với Regularization

Trong quá trình huấn luyện mạng nơ-ron có sử dụng Regularization (L2 Regularization), ta sẽ áp dụng Stochastic Gradient Descent (SGD) để cập nhật trọng số dựa trên chi phí đã được regularized. Công thức cho việc cập nhật trọng số và bias trong trường hợp này có sự thay đổi so với phương pháp không regularization.

Trọng số (w) được cập nhật như sau:

Trong đó:

* là learning rate (tỷ lệ học).
* là tham số regularization.
* là số lượng ví dụ trong tập huấn luyện.
* là kích thước của mini-batch.
* là hàm chi phí chưa được áp dụng Regularization cho mỗi ví dụ huấn luyện trong mini-batch.

Quy tắc cập nhật cho biases không thay đổi khi có regularization, công thức được giữ nguyên:

Tương tự như với trọng số, là hàm chi phí chưa được áp dụng Regularization cho mỗi ví dụ huấn luyện trong mini-batch.

### 3.3 L1 Regulation

Nguồn tham khảo: Neural Networks and Deep Learning - Michael Nielsen

a) Giới thiệu L1 Regularization

L1 Regularization là một kỹ thuật giúp giảm hiện tượng overfitting trong mô hình học máy bằng cách điều chỉnh hàm chi phí (cost function). Kỹ thuật này liên quan đến việc thêm vào một đại lượng mới vào hàm chi phí, đó là tổng của giá trị tuyệt đối của các trọng số (weights) trong mô hình.

Biểu thức hàm chi phí trong trường hợp này là:

Trong đó,

* ​ là hàm chi phí ban đầu,
* là các trọng số của mô hình và
* là tham số regularization.

Để hiểu rõ hơn cách thức hoạt động của L1 regularization, chúng ta xét đạo hàm của hàm chi phí với đối tượng là các trọng số . Sau khi đạo hàm, ta có biểu thức:

Trong đó, là hàm dấu của trọng số , tức là nếu dương và nếu âm.

Nhờ vào biểu thức này, chúng ta có thể điều chỉnh thuật toán gradient descent để áp dụng L1 regularization. Quy tắc cập nhật trọng số trong L1 regularization sẽ là:

Trong đó, là tốc độ học và ​​ có thể được ước tính thông qua trung bình của mini-batch.

Đối với bias, ta sẽ cập nhật theo công thức:

b) So sánh giữa L1 và L2 Regularization

Cả L1 và L2 regularization đều có tác dụng làm giảm các trọng số của mô hình, giúp giảm sự phức tạp của mô hình và từ đó giảm overfitting. Tuy nhiên, cách thức mà các trọng số giảm đi giữa L1 và L2 có sự khác biệt.

* Trong L1 regularization, so với cách cập nhật trọng số trong thuật toán Gradient Decend các trọng số được thay đổi một lượng cố định () theo hướng về 0. Điều này có nghĩa là mỗi trọng số bị thay đổi một giá trị không phụ thuộc vào giá trị ban đầu của nó, giúp đẩy các trọng số nhỏ về 0.
* Trong khi đó, L2 regularization làm giảm các trọng số theo tỷ lệ trực tiếp với giá trị của chúng. Nghĩa là, trọng số càng lớn thì bị giảm mạnh hơn, còn trọng số nhỏ bị giảm ít hơn.

## 4. Các hàm kích hoạt (Activation Functions)

Nguồn tham khảo:

* Introduction to Activation Functions in Neural Networks - Datacamp
* Activation Functions | Fundamentals Of Deep Learning – Analytics Vidhya
* Activation Function in Neural Networks - Analytics Vidhya

### 4.1 Hàm kích hoạt (Activation Functions)

4.1.1 Giới thiệu hàm kích hoạt

Hàm kích hoạt là một thành phần cốt lõi của mạng nơ-ron, giúp chúng học được các mẫu dữ liệu phức tạp. Chúng biến đổi tín hiệu đầu vào của một nút trong mạng nơ-ron thành tín hiệu đầu ra, sau đó được truyền đến lớp tiếp theo.

Nếu không có hàm kích hoạt, mạng nơ-ron sẽ chỉ có thể mô hình hóa các mối quan hệ tuyến tính giữa đầu vào và đầu ra. Hàm kích hoạt đưa vào tính phi tuyến, cho phép mạng nơ-ron học được các ánh xạ phức tạp giữa đầu vào và đầu ra.

Việc lựa chọn hàm kích hoạt phù hợp rất quan trọng để huấn luyện mạng nơ-ron có khả năng tổng quát hóa tốt và đưa ra dự đoán chính xác.

4.1.2 Cách sử dụng hàm kích hoạt

Trong một mạng nơ-ron, mỗi nơ-ron sẽ thực hiện hai phép tính

1. Tổng tuyến tính của các đầu vào: với đầu vào có trọng số và bias . Tổng tuyến tính được tính bằng:
2. Tính toán hàm kích hoạt: Phép tính này quyết định xem một nơ-ron có nên được kích hoạt hay không bằng cách tính tổng trọng số và cộng thêm bias. Mục đích của hàm kích hoạt là đưa tính phi tuyến vào đầu ra của nơ-ron.

Hầu hết các mạng nơ-ron bắt đầu bằng việc tính tổng trọng số của các đầu vào. Mỗi nút trong một lớp có thể có trọng số riêng biệt. Tuy nhiên, hàm kích hoạt lại giống nhau trên tất cả các nút trong lớp đó. Chúng thường có dạng cố định, trong khi các trọng số được coi là tham số học (learning parameters).

### 4.2 Hàm kích hoạt Sigmoid (Sigmoid Activation Function)

4.2.1 Giới thiệu

Hàm sigmoid, thường được ký hiệu là , là một hàm trơn liên tục và khả vi, đóng vai trò quan trọng trong lịch sử phát triển của mạng nơ-ron. Về mặt toán học, hàm sigmoid được định nghĩa như sau:

Đồ thị hàm sigmoid

A graph of a function

AI-generated content may be incorrect.

Nhận xét

* Hàm này nhận đầu vào là một giá trị thực và cho đầu ra trong khoảng từ 0 đến 1.
* Đồ thị của hàm sigmoid có dạng hình chữ "S", tiệm cận về 0 khi đầu vào là số âm rất lớn và tiệm cận về 1 khi đầu vào là số dương rất lớn.
* Do đầu ra nằm trong khoảng xác suất (0,1), hàm sigmoid thường được sử dụng một cách tự nhiên cho các bài toán phân loại nhị phân (binary classification).

Triển khai trong Python

import numpy as np

def sigmoid\_function(x):

return 1 / (1 + np.exp(-x))

print(sigmoid\_function(7), sigmoid\_function(-22))

# Output: (0.9990889488055994,2.7894680920908113e-10)

Đạo hàm của hàm sigmoid

Đồ thị đạo hàm của hàm sigmoid

A graph of a function

AI-generated content may be incorrect.

Nhận xét

* Các giá trị gradient có độ lớn đáng kể trong khoảng từ đến , nhưng ở các vùng khác đồ thị trở nên phẳng hơn.
* Điều này có nghĩa là những giá trị lớn hơn hoặc nhỏ hơn sẽ có gradient rất nhỏ. Khi giá trị gradient tiến gần đến 0, mạng thực sự không học được gì.

4.2.2 Ưu điểm

* Đầu ra nằm trong khoảng (0, 1): Hữu ích cho bài toán phân loại nhị phân
* Liên tục và khả vi: Có thể tính gradient để tối ưu bằng lan truyền ngược (backpropagation).

4.2.3 Nhược điểm

Hàm sigmoid gặp phải vấn đề gọi là vanishing gradient, đặc biệt khi mạng nơ-ron có nhiều lớp (deep neural networks). Khi đầu vào có giá trị quá lớn (dương hoặc âm), hàm sigmoid bão hòa (saturate) tại 0 hoặc 1. Trong những trường hợp này, gradient trở nên cực kỳ nhỏ, dẫn đến:

* Cập nhật trọng số rất chậm trong quá trình lan truyền ngược.
* Các lớp đầu mạng (early layers) hầu như không học được gì, khiến việc huấn luyện trở nên kém hiệu quả hoặc thậm chí dừng lại.

4.2.4 Ứng dụng chính

Ngày nay, hàm sigmoid chủ yếu được sử dụng ở lớp đầu ra của các mô hình phân loại nhị phân, giúp chuyển đổi đầu ra thành xác suất thuộc về một lớp cụ thể.

### 4.3 Hàm kích hoạt Tanh (Hyperbolic Tangent)

4.3.1 Giới thiệu

Hàm tanh (tangent hyperbolic) được định nghĩa như sau:

Đồ thị hàm tanh

A graph of a function

AI-generated content may be incorrect.

Nhận xét: Hàm này cho đầu ra trong khoảng −1 đến +1, giúp xử lý tốt hơn các giá trị âm so với hàm sigmoid (vốn chỉ nằm trong khoảng 0 đến 1).

Triển khai trong Python

import numpy as np

def tanh\_function(x):

z = (2/(1 + np.exp(-2\*x))) -1

return z

print(tanh\_function(0.5), tanh\_function(-1))

# Output: (0.4621171572600098, -0.7615941559557646)

Đạo hàm của hàm tanh

Đồ thị đạo hàm của hàm tanh

A graph of a function

AI-generated content may be incorrect.

Nhận xét: Gradient của hàm dốc hơn so với hàm .

4.3.2 Ưu điểm so với Sigmoid

- Đối xứng qua gốc tọa độ (zero-centered):

* Đầu ra của tanh cân bằng quanh điểm 0. Sự cân bằng này giúp điều chỉnh các cập nhật trọng số hiệu quả hơn trong quá trình tối ưu bằng gradient descent.

- Gradient mạnh hơn:

* Gradient của tanh thường lớn hơn so với sigmoid.
* Điều này giúp mạng nơ-ron học nhanh hơn và giảm thiểu phần nào vấn đề vanishing gradients.

4.3.3 Nhược điểm

Vẫn gặp vấn đề tiêu biến gradient. Dù tốt hơn sigmoid, hàm tanh vẫn có thể bị bão hòa (saturate) khi đầu vào quá lớn hoặc quá nhỏ, dẫn đến gradient gần 0. Đặc biệt trong các mạng sâu (deep networks), điều này khiến:

* Gradient truyền ngược trở nên cực kỳ nhỏ.
* Các lớp đầu mạng khó cập nhật trọng số, làm chậm quá trình huấn luyện hoặc dẫn đến hội tụ kém.

4.3.4 Ứng dụng

* Tanh thường được dùng trong các lớp ẩn (hidden layers) của mạng nơ-ron, đặc biệt khi dữ liệu được chuẩn hóa về trung bình 0 (zero-mean).
* Trong hầu hết trường hợp, tanh được ưu tiên hơn sigmoid do tính chất zero-centered và gradient mạnh hơn.
* Tuy nhiên, lựa chọn cuối cùng phụ thuộc vào bài toán cụ thể và kết quả thử nghiệm ban đầu.

### 4.4 Hàm kích hoạt ReLU (Rectified Linear Unit)

4.4.1 Giới thiệu

Hàm ReLU có dạng đơn giản:

Nó hoạt động bằng cách:

* Trả về 0 nếu đầu vào âm .
* Giữ nguyên giá trị đầu vào nếu dương .

Đồ thị hàm ReLU

#### A graph of a function AI-generated content may be incorrect.

Triển khai Python

def relu\_function(x):

if x<0:

return 0

else:

return x

print(relu\_function(7), relu\_function(-7))

# Output: (7, 0)

Đạo hàm của hàm ReLU

Đồ thị đạo hàm của hàm ReLU

A graphing of a function

AI-generated content may be incorrect.

4.4.2 Ưu điểm

Khắc phục Vanishing Gradient:

* + Với , ReLU có đạo hàm luôn bằng , giúp gradient truyền ngược không bị suy giảm qua các lớp.
  + Điều này giải quyết vấn đề nghiêm trọng của sigmoid/tanh trong mạng sâu.

Tính phi tuyến (Non-linearity):

* + Dù có dạng tuyến tính khi , ReLU vẫn là hàm phi tuyến do điểm gãy tại .
  + Đặc tính này cho phép mạng nơ-ron học các mẫu phức tạp.

Hiệu quả tính toán:

* + Chỉ cần phép so sánh và gán đơn giản → tốc độ tính toán cực nhanh so với sigmoid/tanh (không cần hàm mũ).
  + Tạo sự kích hoạt thưa (sparse activation): Nhiều nơ-ron trả về 0, giảm đáng kể chi phí tính toán.

4.4.3 Nhược điểm

Vấn đề Dying ReLU:

* + Nếu một nơ-ron luôn xuất ra 0 (do đầu vào âm liên tục), nó sẽ ngừng học vì gradient tại đó bằng 0.

4.4.4 Ứng dụng

ReLU là lựa chọn mặc định cho các lớp ẩn (hidden layers) trong hầu hết kiến trúc mạng nơ-ron hiện đại.

### 4.5 Hàm kích hoạt Leaky ReLU

4.5.1 Giới thiệu

Leaky ReLU là một phiên bản cải tiến của ReLU, khắc phục nhược điểm Dying ReLU bằng cách cho phép một gradient nhỏ khi đầu vào âm .

Công thức toán học

Đồ thị hàm Leaky ReLU

A graph of a function

AI-generated content may be incorrect.

Triển khai Python

def leaky\_relu\_function(x, alpha=0.01):

if x < 0:

return alpha \* x

else:

return x

print(leaky\_relu\_function(7), leaky\_relu\_function (-7))

# Output: (7, -0.07)

Đạo hàm của hàm Leaky ReLU

Đồ thị đạo hàm của hàm Leaky ReLU

A graphing of a function

AI-generated content may be incorrect.

4.5.2 Ưu điểm so với ReLU

Khắc phục Dying ReLU:

* Khi đầu vào , , vì thế trọng số vẫn được cập nhật

Giữ được tính kích hoạt thưa (sparse activation):

* Nhiều nơ-ron trả về 0, giảm đáng kể chi phí tính toán.

4.5.3 Ứng dụng

* Mạng rất sâu (ví dụ: Transformer) nơi Dead ReLU dễ xảy ra.
* Dữ liệu có nhiều giá trị âm

### 4.6 Hàm kích hoạt Parametric ReLU (PReLU)

4.6.1 Giới thiệu

Parametric ReLU (PReLU) là một phiên bản cải tiến của Leaky ReLU, trong đó hệ số cho phần âm không cố định mà được học tự động trong quá trình huấn luyện. Điều này giúp tối ưu hóa hiệu suất của mạng cho từng bài toán cụ thể.

Công thức toán học

Đồ thị hàm Parametric ReLU

A graph of a function

AI-generated content may be incorrect.

Đạo hàm của hàm Parametric ReLU

4.6.2 Ưu điểm

Tối ưu hóa tự động: được tự động điều chỉnh phù hợp với dự liệu -> cả thiện độ chính xác

4.6.3 Nhược điểm

Cần lưu trữ và cập nhật thêm tham số  → Tăng dung lượng mô hình.

4.6.4 Ứng dụng

Hàm ReLU tham số hóa (Parametric ReLU - PReLU) được sử dụng khi hàm Leaky ReLU vẫn không khắc phục được hiện tượng Dying ReLU

### 4.7 Hàm kích hoạt ELU (Exponential Linear Unit)

4.7.1 Giới thiệu

ELU (Exponential Linear Unit) là một biến thể ReLU, khắc phục hiện tượng Dying ReLU bằng cách sử dụng hàm mũ cho giá trị âm.

Công thức toán học

* Với

Đồ thị hàm ELU

A graph of a function

AI-generated content may be incorrect.

Triển khai python

import numpy as np

def elu\_function(x, a):

if x<0:

return a\*(np.exp(x)-1)

else:

return x

print(elu\_function(5, 0.1), elu\_function(-5, 0.1))

# Output: (5, -0.09932620530009145)

Đạo hàm của hàm ELU

4.7.2 Ưu điểm

Khắc phục Dead ReLU:

* Giá trị âm được xử lý nhờ hàm mũ → Tránh hiện tượng Dying ReLU.

4.7.3 Nhược điểm

Tính toán phức tạp hơn:

Sử dung hàm -> tốn chi phí hơn so với .

### 4.8 Hàm kích Hoạt Swish

4.8.1 Giới thiệu

Swish là một hàm kích hoạt ít được biết đến hơn, được phát hiện bởi các nhà nghiên cứu tại Google. Swish thể hiện hiệu suất vượt trội hơn ReLU trên các mô hình sâu hơn. Công thức của Swish:

* là hàm Sigmoid
* Miền giá trị

Đồ thị hàm Swish

A graph with a line drawn on it

AI-generated content may be incorrect.

Nhận xét

* Đồ thị của hàm này liên tục và hàm có thể vi phân được tại mọi điểm. Đặc tính này rất hữu ích trong quá trình tối ưu hóa mô hình và được coi là một trong những lý do giúp hàm swish vượt trội so với ReLU.
* Một điểm đặc biệt về hàm này là hàm swish không đơn điệu. Điều này có nghĩa là giá trị của hàm có thể giảm ngay cả khi giá trị đầu vào đang tăng.

Triển khai Python

import numpy as np

def swish\_function(x):

return x/(1-np.exp(-x))

print(swish\_function(-67), swish\_function(4))

# Output: (5.35e-28, 4.0746)

4.8.2 Ưu điểm

Tính chất hàm:

* Swish: hàm liên tục và khả vi tại mọi điểm, không đơn điệu
* ReLU: hàm tuyến tính từng khúc

Xử lý giá trị âm:

* Swish cho phép xử lý giá trị âm
* ReLU ép tất cả giá trị âm về 0

4.8.3 Nhược điểm

Tốn chi phí tính toán hơn ReLU:

* Cần tính hàm

### 4.9 Hàm kích hoạt Softmax

4.9.1 Giới thiệu

Hàm softmax (còn gọi là hàm mũ chuẩn hóa) là một công cụ quan trọng cho các bài toán phân loại đa lớp (multi-class classification). Nó biến đổi một vector đầu vào (thường gọi là logits - điểm số dự đoán từ các lớp trước đó của mạng) thành một phân phối xác suất.

* Công thức toán học

Với vector đầu vào , softmax được định nghĩa:

* : Đầu vào của nơ-ron thứ (logit)
* Mẫu số: Tổng hàm mũ của tất cả logits -> Chuẩn hóa thành phân phối xác suất
* Ví dụ minh họa:

Lớp đầu ra có 3 nơ-ron, mỗi nơ-ron tương ứng một lớp.

Công thức Softmax cho nơ-ron thứ

Đầu vào (logits) là:

Tính hàm mũ (exponential) cho từng logit

Tính tổng hàm mũ

Chuẩn hóa thành xác suất

Kết quả

→ Xác suất lần lượt là 42%, 31%, 27% (tổng = 100%).

* Triển khai Python

import numpy as np

def softmax\_function(x):

z = np.exp(x)

z\_ = z/z.sum()

return z\_

print(softmax\_function([0.8, 1.2, 3.1]))

# Output: [0.0802, 0.1197, 0.8001]

4.9.2 Đặc điểm nội bật

- Đầu ra là xác suất:

* + Mỗi phần tử đầu ra nằm trong khoảng và tổng cả vector bằng → có thể hiểu là xác suất thuộc về từng lớp.
  + Hàm mũ eˣ đảm bảo đầu ra luôn dương, phù hợp với tính chất xác suất.

- Khuếch đại sự khác biệt:

* + Softmax phóng đại sự chênh lệch giữa các giá trị đầu vào (Ngay cả những khác biệt nhỏ trong giá trị đầu vào cũng có thể dẫn đến sự khác biệt đáng kể trong xác suất đầu ra.).
  + Lớp có điểm số cao nhất sẽ chiếm xác suất áp đảo, giúp mô hình đưa ra quyết định rõ ràng.

- Ứng dụng chính:

* + Thường dùng ở lớp đầu ra của mạng nơ-ron cho bài toán phân loại đa lớp (ví dụ: nhận diện chữ số MNIST).
  + Xác suất đầu ra có thể xem như độ tin cậy của mô hình với từng dự đoán.

- Nhạy cảm với giá trị ngoại lai (outliers):

* + Nếu một giá trị đầu vào quá lớn, softmax sẽ "dồn hết xác suất" vào lớp đó, khiến mô hình trở nên quá tự tin (overconfident).

# **IV. Tiền xử lý dữ liệu (Data Preprocessing)**

## 1. Tổng quan về tiền xử lý dữ liệu

Nguồn:

Data Mining: Concepts and Techniques - Jiawei Han, Micheline Kamber, Jian Pei

### 1.1 Giới thiệu

Tiền xử lý dữ liệu là bước quan trọng trong quá trình khai thác dữ liệu (data mining), giúp cải thiện chất lượng dữ liệu và nâng cao hiệu quả phân tích. Trong thực tế, dữ liệu thường chứa nhiều vấn đề như thiếu sót, nhiễu, không nhất quán hoặc dư thừa.

### 1.2 Các yếu tố ảnh hưởng đến chất lượng dữ liệu

a) Độ chính xác (Accuracy)  
Dữ liệu chất lượng phải đảm bảo tính chính xác tuyệt đối, không chứa sai sót từ khâu nhập liệu, truyền tải hay vận hành thiết bị.

b) Độ đầy đủ (Completeness)  
Dữ liệu lý tưởng cần bao gồm đầy đủ tất cả các thuộc tính quan trọng, không để xảy ra tình trạng thiếu giá trị hoặc thông tin chưa hoàn thiện.

c) Tính nhất quán (Consistency)  
Hệ thống dữ liệu chuẩn phải duy trì sự đồng bộ cao trong cách đặt tên, định dạng và phương thức mã hóa xuyên suốt các nguồn dữ liệu khác nhau.

d) Tính kịp thời (Timeliness)  
Dữ liệu chất lượng luôn được cập nhật liên tục và kịp thời, đảm bảo phản ánh chính xác nhất trạng thái hiện tại của hệ thống.

e) Độ tin cậy (Believability)  
Dữ liệu đáng tin cậy phải tạo được niềm tin vững chắc từ người dùng thông qua quá trình kiểm chứng nghiêm ngặt và lịch sử chính xác.

f) Khả năng diễn giải (Interpretability)  
Dữ liệu chuẩn mực cần được trình bày rõ ràng, dễ hiểu với hệ thống mã hóa thân thiện, giúp người dùng dễ dàng nắm bắt thông tin.

### 1.3 Các kỹ thuật tiền xử lý dữ liệu chính

a) Làm sạch dữ liệu (Data Cleaning)

Data Cleaning là một quy trình quan trọng để "làm sạch" dữ liệu bằng cách xử lý các vấn đề như:

* Điền giá trị thiếu: Đảm bảo rằng các giá trị bị thiếu được điền đầy đủ.
* Làm mịn dữ liệu bị nhiễu: Xử lý các dữ liệu không đồng nhất hoặc bị sai lệch để không ảnh hưởng đến quá trình phân tích.
* Loại bỏ giá trị ngoại lệ: Phát hiện và loại bỏ các dữ liệu không hợp lý hoặc ngoài phạm vi thông thường.
* Giải quyết sự không nhất quán: Xử lý các vấn đề về sự không nhất quán trong dữ liệu, như các giá trị có tên gọi khác nhau cho cùng một khái niệm.

Dữ liệu bẩn có thể làm cho quá trình khai thác dữ liệu trở nên khó khăn và không đáng tin cậy. Do đó, việc làm sạch dữ liệu là rất cần thiết trước khi thực hiện các bước khai thác dữ liệu tiếp theo.

b) Tích hợp dữ liệu (Data Integration)

Trong nhiều trường hợp, dữ liệu được thu thập từ nhiều nguồn khác nhau, chẳng hạn như cơ sở dữ liệu, tệp dữ liệu, hoặc các khối dữ liệu. Việc tích hợp dữ liệu từ các nguồn khác nhau có thể gặp phải một số vấn đề như:

* Khác biệt trong tên gọi thuộc tính: Các thuộc tính đại diện cho một khái niệm có thể có tên gọi khác nhau trong các cơ sở dữ liệu khác nhau, gây ra sự không nhất quán và dư thừa.
* Sự không nhất quán trong giá trị thuộc tính: Các giá trị thuộc tính có thể khác nhau mặc dù chúng thực sự đại diện cho cùng một đối tượng. Ví dụ, tên “Bill” có thể xuất hiện trong cơ sở dữ liệu này, nhưng lại là “William” hoặc “B.” trong cơ sở dữ liệu khác.

Việc xử lý sự không nhất quán và dư thừa trong quá trình tích hợp dữ liệu là rất quan trọng. Ngoài việc làm sạch dữ liệu, cần phải tránh sự dư thừa khi tích hợp dữ liệu để không làm chậm quá trình khai thác tri thức.

c) Giảm kích thước dữ liệu (Data Reduction)

Khi xử lý dữ liệu có kích thước quá lớn, việc giảm dung lượng của dữ liệu mà không làm mất đi giá trị phân tích là rất cần thiết. Các chiến lược giảm dữ liệu bao gồm:

* Giảm chiều dữ liệu (Dimensionality Reduction): Áp dụng các phương pháp mã hóa dữ liệu để giảm kích thước dữ liệu mà không làm mất đi thông tin quan trọng. Các ví dụ bao gồm phân tích thành phần chính (PCA) hoặc biến đổi sóng.
* Giảm số lượng dữ liệu (Numerosity Reduction): Sử dụng các mô hình tham số hoặc không tham số để thay thế dữ liệu gốc bằng các đại diện nhỏ hơn, chẳng hạn như thông qua phân cụm hoặc mẫu.

Giảm dữ liệu giúp làm giảm khối lượng công việc tính toán, đồng thời giữ lại các thông tin cần thiết cho quá trình khai thác dữ liệu.

d) Biến đổi dữ liệu (Data Transformation)

Chuyển đổi dữ liệu bao gồm các quy trình để thay đổi hoặc tái cấu trúc dữ liệu sao cho phù hợp với các thuật toán khai thác dữ liệu. Một số kỹ thuật chuyển đổi dữ liệu bao gồm:

* Chuẩn hóa dữ liệu (Normalization): Các thuộc tính có phạm vi giá trị khác nhau, như tuổi và lương hàng năm, có thể gây ra sự không công bằng trong việc tính toán khoảng cách. Chuẩn hóa giúp các giá trị này nằm trong một phạm vi đồng nhất, ví dụ như [0.0, 1.0].
* Phân biệt hóa dữ liệu (Discretization và Concept Hierarchy Generation): Các giá trị thô của thuộc tính có thể được thay thế bằng các phạm vi hoặc cấp độ khái niệm cao hơn, ví dụ như thay thế giá trị tuổi bằng các nhóm như "thanh niên", "người trưởng thành" hoặc "người cao tuổi".
* Tạo cấp độ khái niệm (Concept Hierarchy Generation): Các thuộc tính có thể được nhóm thành các cấp độ khái niệm cao hơn để giúp phân tích dữ liệu ở các mức độ trừu tượng khác nhau.

Các phương pháp này giúp dữ liệu phù hợp hơn với các thuật toán khai thác dữ liệu như mạng nơ-ron, phân loại gần nhất hoặc phân cụm.

A diagram of data processing

AI-generated content may be incorrect.

## 2. Data cleaning

Nguồn tham khảo:

Data Mining: Concepts and Techniques - Jiawei Han, Micheline Kamber, Jian Pei

Tổng quan

Dữ liệu trong thế giới thực thường có xu hướng không đầy đủ, nhiễu và không nhất quán. Các quy trình làm sạch dữ liệu cố gắng điền vào các giá trị bị thiếu, làm mịn nhiễu trong dữ liệu trong khi nhận diện các giá trị ngoại lệ và chỉnh sửa những sự không nhất quán trong dữ liệu.

Các kỹ thuật chính

### 2.1 Xử lý giá trị thiếu (Missing Values)

2.1.1 Bỏ qua bản ghi (tuple) có giá trị bị thiếu

* Phương pháp này thường được áp dụng khi nhãn lớp bị thiếu, đặc biệt nếu nhiệm vụ khai thác dữ liệu liên quan đến phân loại.
* Nhược điểm:
  + Hiệu quả không cao nếu tuple chỉ thiếu một vài thuộc tính còn lại đầy đủ thông tin hữu ích.
  + Phương pháp này đặc biệt kém khi tỉ lệ giá trị thiếu của từng thuộc tính có sự biến đổi lớn.
  + Bỏ qua tuple sẽ bỏ lỡ các giá trị hợp lệ ở các thuộc tính khác, có thể có ích cho nhiệm vụ phân tích.

2.1.2 Điền giá trị bị thiếu thủ công

* Cập nhật giá trị cho thuộc tính bị thiếu theo cách thủ công.
* Nhược điểm:
  + Tốn thời gian.
  + Không khả thi khi tập dữ liệu quá lớn và có nhiều giá trị bị thiếu.

2.1.3 Sử dụng một hằng số toàn cục để điền giá trị

* Thay thế tất cả các giá trị thiếu bằng một hằng số cố định, chẳng hạn như “Unknown” hoặc .
* Nhược điểm:
  + Khi thay thế bằng giá trị “Unknown”, chương trình khai thác dữ liệu có thể hiểu nhầm rằng giá trị này tạo thành một khái niệm quan trọng, do tất cả các giá trị đều giống nhau.
  + Dù đơn giản nhưng phương pháp này không đảm bảo độ chính xác cao.

2.1.4 Sử dụng giá trị trung tâm (trung bình hoặc trung vị) của thuộc tính

* Áp dụng các biện pháp thống kê mô tả để chọn giá trị thay thế.
  + Với phân phối dữ liệu đối xứng, giá trị trung bình **(Mean)** có thể được sử dụng.
  + Với phân phối lệch, giá trị trung vị **(Median)** sẽ là lựa chọn phù hợp hơn.
* Ví dụ: Nếu phân phối thu nhập của khách hàng AllElectronics là đối xứng và thu nhập trung bình là 56,000 USD, giá trị này có thể được sử dụng để thay thế giá trị bị thiếu.

2.1.5 Sử dụng giá trị trung tâm tính theo lớp của tuple

* Tính giá trị trung bình hoặc trung vị của thuộc tính cho tất cả các mẫu cùng thuộc một lớp (ví dụ: cùng nhóm rủi ro tín dụng).
* Ưu điểm:
  + Giúp tăng độ chính xác vì giá trị thay thế được tính theo đặc điểm của nhóm mà tuple thuộc về.
  + Với dữ liệu phân phối lệch, trung vị sẽ mang lại kết quả đáng tin cậy hơn.

2.1.6 Sử dụng giá trị có khả năng xảy ra nhất để điền giá trị thiếu

* Xác định giá trị dự đoán cho thuộc tính bị thiếu dựa trên hồi quy (Regression), các công cụ dựa trên suy luận sử dụng hình thức Bayesian (Bayesian formalism) hoặc quy nạp cây quyết định (Decision tree).
* Ưu điểm:
  + Phương pháp này được ưa chuộng do tận dụng được hầu hết thông tin từ các thuộc tính còn lại trong dữ liệu.
  + Giúp bảo toàn mối quan hệ giữa thu nhập và các thuộc tính khác trong quá trình dự đoán.

2.1.7 Đánh giá chung các phương pháp

* Các phương pháp từ 3 đến 6 đều có thể tạo ra sự thiên lệch cho dữ liệu vì giá trị đã điền có thể không chính xác hoàn toàn.
* Tuy nhiên, phương pháp 6 (sử dụng giá trị có khả năng xảy ra nhất) được đánh giá cao nhờ việc sử dụng nhiều thông tin có sẵn trong dữ liệu để dự đoán giá trị bị thiếu, từ đó bảo toàn mối quan hệ giữa các thuộc tính.

2.1.8 Lưu ý về giá trị bị thiếu

* Không phải tất cả giá trị bị thiếu đều là lỗi trong dữ liệu.

Ví dụ: Trong đơn xin thẻ tín dụng, người nộp đơn có thể không cung cấp số giấy phép lái xe nếu họ không có, và trường đó có thể hợp lý để để trống hoặc ghi “không áp dụng”.

* Các biểu mẫu nên cho phép người dùng chỉ định các giá trị như “không áp dụng”, “không biết”, hoặc để trống nếu sẽ được cung cấp ở bước sau.
* Thiết kế cơ sở dữ liệu và quy trình nhập liệu hợp lý có thể giúp giảm số lượng giá trị thiếu hoặc lỗi ngay từ đầu.

### 2.2. Xử lý dữ liệu nhiễu (Noisy Data)

2.2.1 Khái niệm về nhiễu (Noise)

Nhiễu là:

* Là sai số ngẫu nhiên hoặc biến động không mong muốn trong dữ liệu
* Thường biểu hiện qua các giá trị ngoại lệ (outliers)

2.2.2 Các kỹ thuật làm mịn dữ liệu nhằm Loại Bỏ Nhiễu

2.2.2.1 Phân nhóm (Binning):

a) Giới thiệu

Phân nhóm (Binning) là một kỹ thuật quan trọng trong tiền xử lý dữ liệu, được sử dụng để làm mịn dữ liệu số bằng cách giảm thiểu nhiễu và biến động ngẫu nhiên.

b) Các phương pháp phân nhóm chính

Phân nhóm tần số bằng nhau (Equal-frequency binning)

* Mỗi nhóm chứa cùng số lượng giá trị
* Ví dụ: Với kích thước nhóm = 3, dữ liệu giá cả [4, 8, 15, 20, 21, 24, 25, 28, 29] được chia thành:
  + Nhóm 1: [4, 8, 15]
  + Nhóm 2: [20, 21, 24]
  + Nhóm 3: [25, 28, 29]

Làm mịn bằng giá trị trung bình nhóm (smoothing by bin means)

* Thay thế mọi giá trị trong nhóm bằng trung bình của nhóm
* Ví dụ:
  + Nhóm 1: [4, 8, 15]
  + Trung bình: (4+8+15)/3 = 9
  + Kết quả: [9, 9, 9, ...]

Làm mịn bằng giá trị trung vị nhóm (smoothing by bin medians)

* Thay thế bằng median của nhóm
* Ví dụ:
  + Nhóm 1: [1, 2, 3, 4, 100]
  + Trung vị: 3
  + Kết quả: [3, 3, 3, ...]

Làm mịn bằng biên nhóm Smoothing by bin boundaries

* Xác định min/max của nhóm làm biên
* Thay thế mỗi giá trị bằng biên gần nhất
* Ví dụ:
  + Nhóm 1: [4, 8, 15]
  + Nhóm 1: Biên [4, 15]
  + Giá trị 8 → 4 (vì gần 4 hơn 15)
  + Kết quả: [4, 4, 15, ...]

2.2.2.2 Regression (Hồi Quy)

Hồi quy (Regression) là phương pháp làm mịn dữ liệu bằng cách điều chỉnh các giá trị dữ liệu theo một hàm số

Có hai dạng chính:

* Hồi quy tuyến tính (Linear regression):
  + Tìm đường thẳng "tốt nhất" để biểu diễn mối quan hệ giữa hai thuộc tính
  + Cho phép dự đoán một thuộc tính từ thuộc tính kia
* Hồi quy tuyến tính đa biến (Multiple linear regression):
  + Mở rộng của hồi quy tuyến tính
  + Xử lý nhiều hơn hai thuộc tính
  + Biểu diễn dữ liệu trên một bề mặt đa chiều

2.2.2.3 Phân tích ngoại lệ (Outlier Analysis):

Việc phát hiện giá trị ngoại lai có thể được thực hiện thông qua các phương pháp phân cụm, nơi các giá trị tương đồng được tổ chức thành các nhóm (clusters).

Phát hiện ngoại lệ bằng phân cụm:

* Các giá trị tương tự được tổ chức thành nhóm (cluster)
* Giá trị nằm ngoài các cụm được xem là ngoại lệ

A black and white drawing of three dots

AI-generated content may be incorrect.

Nhiều phương pháp làm mịn dữ liệu đồng thời cũng là phương pháp rời rạc hóa dữ liệu (một dạng biến đổi dữ liệu) (data discretization) và giảm dữ liệu(data reduction). Ví dụ:

* Các kỹ thuật phân nhóm (binning) đã mô tả ở trên làm giảm số lượng giá trị riêng biệt cho mỗi thuộc tính
* Điều này hoạt động như một dạng giảm dữ liệu cho các phương pháp khai thác dữ liệu dựa trên logic, như quy nạp cây quyết định - vốn thường xuyên so sánh các giá trị trên dữ liệu đã sắp xếp

Hệ thống phân cấp khái niệm (Concept hierarchies) là một dạng rời rạc hóa dữ liệu cũng có thể được sử dụng để làm mịn dữ liệu. Ví dụ:

* Một hệ thống phân cấp khái niệm cho thuộc tính *giá cả* có thể ánh xạ các giá trị *giá cả* thực tế thành các mức: *rẻ*, *trung bình* và *đắt*
* Qua đó giảm số lượng giá trị dữ liệu cần xử lý trong quá trình khai thác

### 2.3 Làm sạch dữ liệu như một quy trình trong khai thác dữ liệu (Data Cleaning as a Process)

2.3.1 Tổng quan

Làm sạch dữ liệu là một phần thiết yếu trong xử lý dữ liệu nhằm đảm bảo tính chính xác và nhất quán. Các giá trị bị thiếu, nhiễu, và không nhất quán có thể làm sai lệch kết quả phân tích. Trước đó, chúng ta đã tìm hiểu về cách xử lý dữ liệu bị thiếu và kỹ thuật làm mịn dữ liệu. Tuy nhiên, việc làm sạch dữ liệu không chỉ đơn thuần là sử dụng các kỹ thuật riêng lẻ mà còn là một quy trình toàn diện.

2.3.2 Phát hiện sai lệch (Discrepancy Detection)

a) Bước đầu tiên trong quá trình làm sạch dữ liệu là phát hiện sai lệch. Sai lệch có thể phát sinh từ nhiều nguyên nhân, bao gồm:

* Biểu mẫu nhập liệu thiết kế kém với nhiều trường tùy chọn.
* Lỗi do con người khi nhập dữ liệu.
* Lỗi cố ý (ví dụ người cung cấp không muốn tiết lộ thông tin cá nhân).
* Dữ liệu lỗi thời (ví dụ địa chỉ không còn chính xác).
* Sự không nhất quán trong cách biểu diễn dữ liệu và mã hóa.
* Lỗi từ thiết bị đo lường hoặc hệ thống ghi nhận dữ liệu.
* Sử dụng dữ liệu vào mục đích khác với mục đích ban đầu.
* Tích hợp dữ liệu từ nhiều nguồn với cách đặt tên thuộc tính khác nhau.

Để phát hiện sai lệch, người phân tích dữ liệu cần sử dụng metadata (dữ liệu về dữ liệu), bao gồm:

* Kiểu dữ liệu và miền giá trị của từng thuộc tính.
* Các giá trị hợp lệ.
* Mô tả thống kê cơ bản: trung bình, trung vị, mode, độ lệch chuẩn, phạm vi giá trị.
* Mối phụ thuộc giữa các thuộc tính.
* Các giá trị cách quá 2 lần độ lệch chuẩn (standard deviation) so với trung bình có thể được xem là ngoại lệ. Trong giai đoạn này, người dùng có thể viết script hoặc sử dụng các công cụ chuyên dụng.

Một số sai lệch điển hình có thể kể đến:

* Sử dụng không nhất quán mã hóa hoặc định dạng dữ liệu (ví dụ “2010/12/25” so với “25/12/2010”).
* Field overloading – khi nhà phát triển tận dụng phần chưa dùng của trường dữ liệu để lưu thông tin mới, gây lỗi.
* Không tuân thủ các quy tắc
* Quy tắc duy nhất (Unique rule): mỗi giá trị trong thuộc tính phải khác nhau.
* Quy tắc liên tục (Consecutive rule): giá trị không được thiếu trong khoảng nhất định.
* Quy tắc giá trị null (Null rule): cách xử lý giá trị trống, dấu hỏi, hoặc ký tự đặc biệt.

b) Xử lý giá trị null

Các lý do phổ biến gây ra giá trị null bao gồm:

* Người cung cấp từ chối trả lời hoặc thấy thông tin không phù hợp (ví dụ người không lái xe bỏ trống trường “số bằng lái”).
* Người nhập không biết giá trị đúng.
* Giá trị sẽ được bổ sung trong bước xử lý sau.

Quy tắc null cần xác định cách biểu diễn giá trị null (ví dụ: 0 với số, dấu cách với chuỗi, “?” hoặc “không biết”) và cách xử lý tương ứng.

c) Công cụ hỗ trợ phát hiện và xử lý sai lệch

* Data scrubbing tools: Sử dụng kiến thức miền như địa chỉ, kiểm tra chính tả để phát hiện và sửa lỗi.
* Data auditing tools: Phân tích dữ liệu để phát hiện quy luật và mối quan hệ, từ đó xác định các giá trị vi phạm. Có thể sử dụng phân tích thống kê hoặc phân cụm để phát hiện ngoại lệ.

2.3.3 Biến đổi dữ liệu (Data Transformation)

Sau khi phát hiện sai lệch, cần thực hiện biến đổi dữ liệu để sửa lỗi. Việc biến đổi có thể bao gồm:

* Sửa lỗi nhập liệu thủ công bằng cách đối chiếu giấy tờ.
* Dùng công cụ di chuyển dữ liệu (data migration tools) để thực hiện thay thế đơn giản (ví dụ đổi “gender” thành “sex”).
* Sử dụng ETL tools (Extraction/Transformation/Loading) với giao diện đồ họa để định nghĩa và thực hiện biến đổi. Tuy nhiên, chúng chỉ hỗ trợ tập hợp biến đổi giới hạn nên đôi khi cần viết script riêng.

2.3.4 Vấn đề và hướng tiếp cận mới

Quá trình phát hiện và biến đổi sai lệch là lặp đi lặp lại, dễ sai sót và tốn thời gian. Một số vấn đề gồm:

* Biến đổi sai có thể gây ra lỗi mới.
* Một số sai lệch lồng nhau chỉ lộ ra sau khi các lỗi khác được xử lý.
* Người dùng không nhận được phản hồi cho đến khi toàn bộ quy trình hoàn tất.
* Các dòng dữ liệu lỗi không được xử lý thường bị ghi vào file mà không giải thích lý do.

Để giải quyết vấn đề này, các công cụ mới tập trung vào tăng tính tương tác:

* Potter’s Wheel là một công cụ làm sạch dữ liệu công khai, tích hợp phát hiện sai lệch và biến đổi. Người dùng có thể thực hiện từng bước biến đổi trên giao diện dạng bảng tính, dễ dàng hoàn tác khi gặp lỗi. Công cụ này phát hiện sai lệch tự động dựa trên phiên bản dữ liệu mới nhất sau mỗi biến đổi.
* Một hướng khác là phát triển ngôn ngữ khai báo cho phép người dùng định nghĩa các toán tử biến đổi một cách hiệu quả (ví dụ mở rộng SQL với các cú pháp chuyên biệt).

2.3.5 Cập nhật metadata

Trong suốt quá trình làm sạch, việc cập nhật metadata là cực kỳ quan trọng nhằm lưu giữ các phát hiện mới về dữ liệu. Việc này giúp tăng tốc quá trình làm sạch cho các phiên bản dữ liệu sau.

## 3. Data integration

Nguồn tham khảo:

Data Mining: Concepts and Techniques - Jiawei Han, Micheline Kamber, Jian Pei

### 3.1. Tổng quan

Khai phá dữ liệu thường đòi hỏi tích hợp dữ liệu - quá trình hợp nhất thông tin từ nhiều nguồn lưu trữ khác nhau. Việc tích hợp (Integration) cẩn thận giúp giảm thiểu tình trạng dư thừa và mâu thuẫn trong bộ dữ liệu cuối cùng, từ đó nâng cao độ chính xác và tốc độ của quá trình khai phá tiếp theo.

### 3.2. Vấn đề nhận diện thực thể (The Entity Identification Problem)

3.2.1. Giới thiệu  
Trong quá trình phân tích dữ liệu, nhiệm vụ tích hợp dữ liệu là điều không thể tránh khỏi, khi các nguồn dữ liệu khác nhau được kết hợp lại thành một kho dữ liệu thống nhất như trong trường hợp xây dựng dữ liệu cho kho dữ liệu (data warehousing). Các nguồn dữ liệu này có thể bao gồm nhiều cơ sở dữ liệu, các khối dữ liệu (data cubes) hoặc các tệp dữ liệu dạng phẳng.

3.2.2. Vấn đề nhận diện thực thể (Entity Identification Problem)

Một trong những vấn đề quan trọng trong quá trình tích hợp dữ liệu là làm sao để ghép nối các thực thể cùng loại từ các nguồn dữ liệu khác nhau, hay nói cách khác, xác định rằng các thực thể trong thế giới thực được trình bày dưới các tên gọi khác nhau trong các nguồn dữ liệu khác nhau đều đề cập đến cùng một đối tượng.

* Ví dụ:
  + Làm sao để nhà phân tích dữ liệu hoặc hệ thống tự động khẳng định rằng “customer id” trong một cơ sở dữ liệu và “cust number” trong cơ sở dữ liệu khác thực sự chỉ về cùng một thuộc tính.

3.2.3. Vai trò của metadata trong xác định thực thể

Để giải quyết vấn đề nhận diện thực thể, metadata đóng vai trò rất quan trọng. Các thông tin metadata của mỗi thuộc tính bao gồm:

* Tên thuộc tính
* Ý nghĩa của thuộc tính
* Kiểu dữ liệu
* Phạm vi giá trị cho phép
* Quy tắc xử lý giá trị trống, số 0 hoặc giá trị null

Những thông tin này giúp xác định và tránh được các lỗi trong quá trình tích hợp sơ đồ (schema integration). Metadata cũng được sử dụng để hỗ trợ quá trình biến đổi dữ liệu, ví dụ: chuyển đổi các mã cho loại hình thanh toán (trong một cơ sở dữ liệu mã có thể là “H” và “S”, trong khi ở cơ sở dữ liệu khác là 1 và 2).

3.2.4. Chú ý đặc biệt khi ghép nối thuộc tính giữa các cơ sở dữ liệu  
Khi tiến hành ghép nối thuộc tính từ một cơ sở dữ liệu này sang cơ sở dữ liệu khác, cần lưu ý đến cấu trúc của dữ liệu, nhằm đảm bảo:

* Các phụ thuộc hàm của các thuộc tính (functional dependencies)
* Các ràng buộc tham chiếu (referential constraints)

Ví dụ, trong một hệ thống, chiết khấu có thể được áp dụng cho toàn bộ đơn hàng, trong khi ở hệ thống khác, chiết khấu lại được áp dụng cho từng dòng sản phẩm trong đơn hàng. Nếu không phát hiện kịp thời sự khác biệt này trước khi tích hợp, các mục trong hệ thống đích có thể bị áp dụng chiết khấu không chính xác.

### 3.3. Redundancy and Correlation Analysis

3.3.1. Giới thiệu   
Dư thừa là một vấn đề quan trọng trong quá trình tích hợp dữ liệu. Một thuộc tính (chẳng hạn “doanh thu hằng năm”) có thể bị coi là dư thừa nếu nó có thể được suy ra từ một hoặc một tập các thuộc tính khác. Bên cạnh đó, sự không nhất quán trong đặt tên thuộc tính hoặc chiều dữ liệu cũng có thể dẫn đến tình trạng dư thừa trong tập dữ liệu thu được sau quá trình tích hợp.

Một số dư thừa có thể được nhận diện nhờ phân tích tương quan (correlation analysis). Khi xét hai thuộc tính, phương pháp này đo lường mức độ một thuộc tính hàm ý giá trị của thuộc tính còn lại, dựa trên dữ liệu sẵn có.

* Nominal data
* Là dữ liệu định tính dùng để đặt tên hoặc gán nhãn cho các biến mà không cung cấp giá trị số (Ví dụ: Màu sắc: Đỏ, Xanh, Vàng.)
* Có thể áp dụng phép kiểm định chi bình phương để đánh giá.
* Thuộc tính số (Numeric attributes):
* Là dữ liệu định lượng, có thể đo lường bằng số
* Sử dụng hệ số tương quan (correlation coefficient) và hiệp phương sai (covariance) để đánh giá mức độ biến thiên của một thuộc tính so với thuộc tính còn lại.

3.3.2.Kiểm định tương quan cho nominal data ( Correlation Test for Nominal Data)

a. Tổng quan

Đối với nominal data, việc xác định mối tương quan (correlation) giữa hai thuộc tính có thể được thực hiện thông qua kiểm định tương quan .

Ví dụ, ta có hai thuộc tính và . Giả sử thuộc tính có giá trị phân biệt, ký hiệu và thuộc tính có giá trị phân biệt, ký ​.

Khi đó, các bộ dữ liệu (hay các “tuple”) được mô tả bởi hai thuộc tính và có thể trình bày dưới dạng *contingency table*, trong đó:

* Các giá trị của tạo thành cột,
* Các giá trị của tạo thành hàng.

Mỗi hàng ứng với một giá trị của , mỗi cột ứng với một giá trị của . Bên trong bảng, ô thể hiện cặp kết hợp .

b. Công thức tính giá trị

Giá trị (còn gọi là thống kê Pearson ) được xác định bằng công thức sau:

Trong đó:

* ​ là tần suất quan sát được (observed frequency) của cặp
* là tần suất kỳ vọng (expected frequency) của cặp khi giả định hai thuộc tính độc lập, và được tính như sau:

với

* là tổng số bộ dữ liệu.
* là số lượng bộ dữ liệu có giá trị ​ cho thuộc tính .
* là số lượng bộ dữ liệu có giá trị ​ cho thuộc tính .

Thống kê được dùng để kiểm định giả thuyết rằng và là độc lập, tức là không có mối tương quan giữa chúng. Bài kiểm định này dựa trên một mức ý nghĩa xác định, với bậc tự do. Nếu giả thuyết độc lập bị bác bỏ, điều đó cho thấy A và B có **tương quan thống kê**.

Nếu hai thuộc tính độc lập, ​ (tần suất quan sát được) sẽ xấp xỉ giá trị ​. Nhưng nếu quan sát thực tế khác nhiều so với giá trị kỳ vọng, mỗi sự khác biệt đó sẽ đóng góp vào tổng , làm tăng lên.

c. Ví dụ minh họa

**A table with numbers and a few letters

AI-generated content may be incorrect.**

Giả sử một nhóm gồm 1.500 người được khảo sát. Giới tính của từng người được ghi lại. Mỗi người cũng được hỏi về loại sách họ thích đọc: hư cấu (fiction) hay phi hư cấu (non-fiction). Như vậy, chúng ta có hai thuộc tính: giới tính (gender) và loại sách ưa thích (preferred\_reading).

* Giới tính: male, female,
* Thể loại đọc ưa thích: fiction, non-fiction.

Tần suất quan sát (hoặc số lần xuất hiện) của từng cặp kết hợp có thể xảy ra được tổng hợp trong *contingency table* như minh họa.

Trong đó các con số trong ngoặc là giá trị kỳ vọng, các giá trị kỳ vọng này được tính dựa trên phân phối dữ liệu của cả hai thuộc tính, sử dụng phương trình.

- Tính số lượng bộ dữ liệu cho mỗi thuộc tính

* Tổng quan sát

- Tính tần suất kỳ vọng

Để tiến hành kiểm định , trước hết ta tính tần suất kỳ vọng cho từng ô trong bảng, giả sử hai thuộc tính là độc lập. Công thức như sau:

Cụ thể

* Ô (male, fiction)
* Ô (male, non-fiction)
* Ô (female, fiction)
* Ô (female, non-fiction)

- Tính giá trị

Giá trị (còn gọi là thống kê Pearson ) được tính bằng

Với là tần suất quan sát được

* Ô (male, fiction)
* Ô (male, fiction)
* Ô (male, fiction)
* Ô (male, fiction)
* Tính tổng tất cả các ô

- Xác định bậc tự do và so sánh với giá trị tới hạn

* Bậc tự do (Degree of Freedom): Ở đây và , do đó bậc tự do .
* So sánh với giá trị tới hạn:
* Ở mức ý nghĩa , tra Distribution Table, giá trị tới hạn xấp xỉ.
* Nếu tính được lớn hơn , ta bác bỏ giả thuyết độc lập; ngược lại, nếu nhỏ hơn , ta chưa đủ bằng chứng để bác bỏ giả thuyết

Trong ví dụ này (theo số liệu gốc của sách), giá trị tính được lớn hơn , vì vậy ta có thể kết luận rằng gender và preferred\_reading là **không độc lập** và do đó **tương quan** với nhau ở mức ý nghĩa 0.001.

3.3.3. Hệ số tương quan cho dữ liệu số (Correlation Coefficient for Numeric Data)

a. Tổng quan

Khi phân tích dữ liệu số (numeric data), một trong những mục tiêu quan trọng là xác định xem hai thuộc tính có xu hướng biến đổi cùng nhau hay không. Để đo lường mức độ liên hệ đó, chúng ta có thể sử dụng **hệ số tương quan** (Correlation Coefficient), còn được gọi là **hệ số tương quan tích moment****của Pearson** (Pearson’s product moment coefficient), do Karl Pearson phát triển.

b. Công thức tính hệ số tương quan

Giả sử chúng ta có hai thuộc tính số và . Xem xét một tập dữ liệu gồm bộ, với là giá trị của và trong bộ dữ liệu thứ . Khi đó, **hệ số tương quan Pearson** giữa và được xác định bởi công thức:

Trong đó:

* và lần lượt là giá trị trung bình (mean) của và .
* ​ và ​ là độ lệch chuẩn (standard deviation) của và .
* đại diện cho tổng tất cả các tích trên các bộ dữ liệu.
* là tổng số bộ dữ liệu (tuples).

c. Ý nghĩa của hệ số tương quan

* Miền giá trị: Hệ số tương quan ​ nằm trong khoảng .
* Tương quan dương: Nếu , giá trị của tăng sẽ đi kèm với việc cũng tăng; ​ càng gần , mối liên hệ tuyến tính càng mạnh.
* Tương quan âm: Nếu , tăng thì giảm; ​ càng gần , mối liên hệ tuyến tính ngược chiều càng mạnh.
* Gần 0: Nếu , ta có thể kết luận rằng và độc lập và không có mối tương quan giữa chúng.

d. Chú ý

Lưu ý rằng tương quan không đồng nghĩa với quan hệ nhân quả. Nghĩa là, nếu A và B có tương quan, điều đó không có nghĩa là A gây ra B hoặc B gây ra A.

Ví dụ, khi phân tích một cơ sở dữ liệu dân số, ta có thể thấy rằng số lượng bệnh viện và số vụ trộm ô tô trong một khu vực có tương quan với nhau. Tuy nhiên, điều này không có nghĩa là một yếu tố gây ra yếu tố kia. Thực chất, cả hai đều liên quan nhân quả đến một thuộc tính thứ ba, đó là dân số.

### 3.4. Hiệp phương sai (Covariance of Numeric Data)

3.4.1. Giới thiệu

Trong lĩnh vực xác suất và thống kê, **hệ số tương quan (correlation)** và **hiệp phương sai (covariance)** là hai thước đo quan trọng được sử dụng để đánh giá xem hai thuộc tính số thay đổi cùng nhau nhiều hay ít.

Để làm rõ ý tưởng này, chúng ta sẽ xem xét hai thuộc tính số và trên một tập quan sát có bộ dữ liệu . Giá trị trung bình của và được ký hiệu lần lượt là và , hay còn gọi là kỳ vọng của và .

3.4.2.Định nghĩa hiệp phương sai (Covariance)

Hiệp phương sai giữa và được định nghĩa như sau:

Trong đó:

* và là giá trị trung bình (kỳ vọng) của và .

Hiệp phương sai giữa và cũng có thể được tính bằng công thức

Phương trình này sẽ làm đơn giản việc tính toán

Ý nghĩa của hiệp phương sai:

* Nếu , khi tăng thì thường có xu hướng tăng.
* Nếu , khi tăng thì thường có xu hướng giảm (ngược chiều).
* Nếu và là độc lập, tức là chúng không có tương quan với nhau, thì:

Khi đó, hiệp phương sai:

Tuy nhiên, **điều ngược lại không đúng**. Một số cặp biến ngẫu nhiên có thể có hiệp phương sai bằng 0 nhưng **vẫn không độc lập**. Chỉ khi có thêm giả định nhất định (ví dụ: dữ liệu tuân theo phân phối chuẩn đa biến), thì hiệp phương sai bằng 0 mới đồng nghĩa với tính độc lập.

3.4.3. Ví dụ minh họa

Bảng dưới đây minh họa một ví dụ thực tế về giá cổ phiếu của hai công ty AllElectronics và HighTech tại 5 thời điểm khác nhau.

Ta đặt câu hỏi: “Liệu giá cổ phiếu của hai công ty này có bị ảnh hưởng bởi cùng một xu hướng ngành (nghĩa là tăng giảm theo cùng chiều) hay không?”

A table with numbers and text

AI-generated content may be incorrect.

- Tính giá trị trung bình (kỳ vọng)

* Giả sử giá cổ phiếu (theo thời điểm) của AllElectronics là 6, 5, 4, 3, 2, thì:
* Tương tự, giả sử giá cổ phiếu HighTech là 20, 10, 14, 5, 5, thì:

- Tính hiệp phương sai

* Dùng Phương trình ở trên:
* Khi đó

Vì vậy, dựa vào hiệp phương sai dương, ta có thể kết luận rằng giá cổ phiếu của cả hai công ty có xu hướng tăng **cùng nhau**.

3.4.4. Mối liên hệ giữa hiệp phương sai và hệ số tương quan

Hệ số tương quan (correlation coefficient) giữa và có thể được tính từ hiệp phương sai, bằng công thức:

Trong đó ​ và ​ lần lượt là độ lệch chuẩn của và .

### 3.5. Trùng lặp dữ liệu (Tuple Duplication)

3.5.1. Giới thiệu

Trong quá trình phát hiện các dạng dư thừa dữ liệu, bên cạnh việc xem xét trùng lặp ở mức độ thuộc tính, ta cũng cần chú ý đến hiện tượng trùng lặp dữ liệu (tuple duplication). Hiện tượng này xảy ra khi có từ hai bộ dữ liệu trở lên giống hệt nhau (hay gần như giống hệt nhau) cho cùng một trường hợp nhập liệu duy nhất.

Việc sử dụng bảng phi chuẩn hoá (denormalized tables)—thường nhằm tối ưu hóa hiệu năng bằng cách tránh sử dụng các phép nối (joins)—là một trong những nguyên nhân gây nên dư thừa dữ liệu.

Các điểm không nhất quán (inconsistencies) thường nảy sinh do:

* Nhập dữ liệu không chính xác hoặc sai sót trong quá trình thu thập thông tin.
* Cập nhật dữ liệu không đồng bộ, chỉ cập nhật một vài vị trí mà bỏ qua những vị trí khác có cùng bản sao dữ liệu.

3.5.2. Ví dụ

Giả sử một cơ sở dữ liệu đơn đặt hàng (purchase order) chứa các thuộc tính về tên và địa chỉ của người mua. Thay vì chỉ lưu một khoá (key) trỏ đến thông tin này trong một cơ sở dữ liệu khách hàng riêng, người thiết kế lại đưa trực tiếp tên và địa chỉ của người mua vào nhiều bản ghi đơn đặt hàng khác nhau.

Hậu quả:

* Nếu cùng một khách hàng thay đổi địa chỉ, và việc cập nhật chỉ được thực hiện ở một số đơn đặt hàng (nhưng không phải tất cả), sự không nhất quán xuất hiện.
* Một khách hàng có thể “xuất hiện” với nhiều địa chỉ khác nhau trong cùng một cơ sở dữ liệu đơn đặt hàng, dù thực tế chỉ nên có một địa chỉ hiện tại.

### 3.6. Phát hiện và giải quyết xung đột giá trị dữ liệu (Detection and Resolution of Data Value Conflicts)

3.6.1. Giới thiệu

Khi tiến hành tích hợp dữ liệu từ nhiều nguồn, ta không chỉ đối mặt với vấn đề nhận diện thực thể hay dư thừa dữ liệu, mà còn phải xử lý xung đột giá trị dữ liệu (data value conflicts). Đây là những trường hợp mà, đối với cùng một thực thể trong thế giới thực, các giá trị của thuộc tính từ những nguồn khác nhau có thể không thống nhất.

3.6.2. Xung đột giá trị dữ liệu

Trong thực tế, cùng một thực thể trong thế giới thực có thể có các giá trị thuộc tính khác nhau khi đến từ những nguồn dữ liệu khác nhau. Các xung đột này có thể xuất phát từ khác biệt trong cách biểu diễn, thang đo hoặc mã hóa.

Ví dụ:

* Thuộc tính cân nặng (weight) có thể được lưu dưới đơn vị hệ mét trong một hệ thống, nhưng lại được lưu dưới đơn vị hệ Anh trong một hệ thống khác.
* Giá phòng khách sạn trong cùng một chuỗi khách sạn nhưng ở các thành phố khác nhau có thể không chỉ khác nhau về đơn vị tiền tệ, mà còn về dịch vụ kèm theo (ví dụ bữa sáng miễn phí) và thuế suất áp dụng.

3.6.3. Khác biệt giữa các hệ thống tổ chức

Khi trao đổi thông tin giữa các trường học, mỗi trường có thể có chương trình đào tạo và hệ thống đánh giá riêng biệt:

* Một trường đại học có thể sử dụng hệ thống theo quý, giảng dạy ba môn học về cơ sở dữ liệu và cho điểm theo thang từ A+ đến F.
* Một trường khác có thể dùng hệ thống học kỳ, chỉ có hai môn học về cơ sở dữ liệu và cho điểm theo thang điểm từ 1 đến 10.

Việc xây dựng quy tắc chuyển đổi giữa các khóa học và hệ thống điểm số giữa hai trường là rất khó, dẫn đến khó khăn trong việc trao đổi thông tin.

3.6.4. Khác biệt về mức độ trừu tượng

Các thuộc tính cũng có thể khác nhau về mức độ trừu tượng. Ví dụ:

* Trong một cơ sở dữ liệu, doanh số bán hàng tổng có thể ám chỉ một chi nhánh cụ thể của chuỗi All Electronics.
* Trong khi đó, cùng tên thuộc tính trong một cơ sở dữ liệu khác có thể đại diện cho tổng doanh số của tất cả cửa hàng trong khu vực.

Những khác biệt này tạo ra sự không thống nhất trong biểu diễn dữ liệu, làm phức tạp thêm quá trình tích hợp.

# **V. Kiến trúc Transformer và Cơ chế Attention**

Nguồn tham khảo: Transformers for Machine Learning: A Deep Dive - Wael Emara, Kenneth Graham, Uday Kamath

## 1. Cơ chế Attention

### 1.1 Background

1.1.1 Khái quát về Attention Mechanism

Cơ chế chú ý (attention mechanism) là một phương pháp chọn lọc có chủ đích nhằm tập trung vào những yếu tố quan trọng và bỏ qua những yếu tố ít liên quan hơn. Dây thần kinh thị giác con người có khả năng nhận tín hiệu lên tới hàng tỷ bit mỗi giây, tuy nhiên não bộ chỉ có thể xử lý một phần nhỏ trong đó. Do đó, cần có một hệ thống giúp não chọn lọc thông tin để xử lý hiệu quả hơn.

Chú ý thị giác – một dạng đặc biệt của attention – giúp định hướng và duy trì sự tập trung vào một tác nhân cụ thể (có thể là người, vật vô tri hay nhiệm vụ), từ đó nâng cao hiệu quả xử lý thông tin. Điều này cho phép não bộ chỉ tập trung vào một phần nhỏ của dữ liệu, góp phần vào việc sử dụng tài nguyên tối ưu hơn và thúc đẩy sự phát triển cá nhân.

1.1.2 Khung lý thuyết của William James về Attention

William James – người được mệnh danh là “cha đẻ của tâm lý học Mỹ” – đã đề xuất một khung lý thuyết gồm hai thành phần cho cơ chế chú ý:

* Chú ý không chủ đích (Nonvolitional): Là loại chú ý xảy ra một cách tự động, không cần nỗ lực của cá nhân. Nó được kích hoạt bởi độ nổi bật của các tác nhân trong môi trường. Ví dụ: sự chú ý đến một em bé đang khóc hoặc một vật thể có màu sắc khác biệt.
* Chú ý có chủ đích (Volitional): Là sự chú ý dựa trên ý chí và sự lựa chọn của cá nhân. Ví dụ: trả lời một câu hỏi cụ thể hoặc tập trung vào một chi tiết nhất định trong một nhiệm vụ phức tạp.

Trong học máy, khái niệm này được ánh xạ như sau:

* Query ~ chú ý có chủ đích
* Key ~ chú ý không chủ đích
* Value ~ đầu vào cảm giác (sensory input)

Cơ chế attention trong deep learning chính là quá trình lựa chọn thông tin phù hợp (values) dựa trên mức độ tương đồng giữa queries và keys.

1.1.3 Mô hình hóa Attention bằng toán học

A diagram of a memory

AI-generated content may be incorrect.

Cơ chế attention có thể được coi như một bộ nhớ chứa các cặp khóa – giá trị (key-value pairs) và một lớp xử lý đầu ra. Khi truy vấn một key nào đó, cơ chế attention sẽ sinh ra kết quả dựa trên giá trị (value) tương ứng.

Giả sử ta có một bộ nhớ gồm n cặp key-value:

Trong đó:

* là key
* là value
* là query
* là output

Lớp attention sẽ tính độ tương đồng giữa query và mỗi key thông qua một hàm điểm số :

Sau đó, các điểm số này được chuẩn hóa bằng hàm softmax để thành trọng số attention:

Cuối cùng, output là tổng có trọng số của các giá trị :

## 1.2 Các Loại Attention Dựa Trên Hàm Điểm Số (Score-Based Attention)

1.2.1 Dot Product

Hàm điểm số dựa trên tích vô hướng là dạng đơn giản nhất, không yêu cầu tham số nào cần học.  
Cách tính:

Đây là cơ chế nhanh và dễ thực hiện trong thực tế.

1.2.2 Scaled Dot Product

Biến thể này được đề xuất trong mô hình Transformer của Vaswani et al. Hàm điểm số được điều chỉnh bằng cách chia kết quả của tích vô hướng cho căn bậc hai của kích thước vector key ​.

Dot product có xu hướng tăng cường độ lớn, có xu hướng làm cho hàm softmax trả về những giá trị rất nhỏ (Hàm softmax có xu hướng tăng cường giá trị lớn, giảm nhẹ giá trị nhỏ)

Việc chia này nhằm giảm ảnh hưởng tiêu cực của việc dot product tăng quá lớn khi kích thước vector tăng.

1.2.3 Linear, MLP, or Additive

Luồng chú ý dạng additive được đề xuất bởi Luong và cộng sự, trong đó truy vấn (query) và khóa (key) được chiếu qua một tầng ẩn có kích thước , sử dụng các ma trận trọng số học được , và sau đó áp dụng hàm tanh và nhân với vector . Công thức:

Mặc dù dạng attention này có khả năng biểu đạt phức tạp hơn, nhưng nó lại tiêu tốn nhiều tài nguyên bộ nhớ và thời gian tính toán hơn so với các phương pháp dot product.

### 1.3. Mô Hình Attention Trong Sequence-to-Sequence

Trong các mô hình sequence-to-sequence truyền thống như Encoder-Decoder, việc bổ sung lớp attention đã mang đến một bước cải tiến lớn. Cụ thể, khi tích hợp attention, quá trình ánh xạ giữa query, key và value diễn ra như sau:

1. Key và Value: Đầu ra từ các trạng thái của encoder được dùng làm keys (k) và values (v).
2. Query: Đầu ra của decoder tại thời điểm được sử dụng làm query (q).
3. Context vector: Kết quả đầu ra từ lớp attention (gọi là , hay biến ngữ cảnh) được đưa vào để tạo ra trạng thái tiếp theo của decoder tại thời điểm .

Ví dụ cụ thể với dịch máy

**A diagram of a algorithm

AI-generated content may be incorrect.**

Xét một ví dụ dịch từ tiếng Pháp sang tiếng Anh.

* **Encoder** nhận đầu vào là chuỗi tiếng Pháp: *{J'aime, le, thé, .}*. Các trạng thái encoder tương ứng sẽ đóng vai trò là **keys** và **values**.
* **Decoder** đã sinh ra chuỗi tiếng Anh: *{I, love}* tại thời điểm

Tại thời điểm hiện tại , từ **love** sẽ được đưa vào lớp attention làm **query**. Query này sẽ kết hợp với các keys để tạo ra điểm số chưa chuẩn hóa theo công thức: .

Các điểm số này sau đó được chuẩn hóa bằng hàm softmax để tạo thành trọng số attention.

Tiếp theo, các trọng số này kết hợp với các **values** (đầu ra encoder) để tạo ra **vector ngữ cảnh** .

Cuối cùng, vector được kết hợp với trạng thái trước đó của decoder để tạo ra từ tiếp theo — trong ví dụ này là từ *tea* tại thời điểm .

## 2. Transformer

Transformer được thiết kế dựa trên kiến trúc encoder – decoder, kết hợp những điểm mạnh của hai dòng mạng nơ-ron:

* Mạng nơ-ron tích chập (CNN): giúp thực hiện tính toán song song, tăng tốc độ huấn luyện mô hình.
* Mạng nơ-ron hồi tiếp (RNN): giúp nắm bắt thông tin tuần tự dài hạn, có độ dài biến đổi.

Kết hợp hai đặc điểm này, Transformer trở thành một công cụ mạnh mẽ trong việc xử lý thông tin chuỗi, đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

**A diagram of a software architecture

AI-generated content may be incorrect.**

### 2.1. Biểu diễn nguồn và đích (Source and Target Representation)

Trong mô hình xử lý ngôn ngữ tự nhiên (NLP), cả câu đầu vào (source) và đầu ra (target) đều được mã hóa bằng cách:

1. Tách từ thành các token.
2. Chuyển mỗi token qua word embedding.
3. Kết hợp thêm thông tin vị trí bằng mã hóa vị trí (positional encoding).

Như vậy, mỗi câu được biểu diễn bằng một chuỗi vector có thông tin ngữ nghĩa và vị trí.

2.1.1 Word Embedding

Word embedding sẽ chuyển đổi mỗi từ trong câu thành một vector có kích thước cố định. Nếu câu có độ dài , và chiều không gian embedding là , ta có thể biểu diễn toàn bộ câu bằng một ma trận .

2.1.2 Positional Encoding

Vì mô hình Transformer không có tính tuần tự như mạng RNN, nên cần có một cơ chế để cung cấp thông tin về vị trí của từ trong câu — đó chính là mã hóa vị trí (positional encoding). Mã hóa này giúp mô hình nhận biết được vị trí tương đối và tuyệt đối của từng từ.

**Giải pháp bằng hàm lượng giác**

Một phương pháp đơn giản đáp ứng các yêu cầu trên là sử dụng các hàm và .

Cho độ dài câu là , và số chiều embedding là , ma trận positional encoding sẽ có kích thước .

**Công thức tính như sau:**

Trong đó:

* : vị trí của từ trong câu (từ đến ).
* : chỉ số chiều (từ đến )

Hàm sin áp dụng cho các chiều chẵn, và cos cho chiều lẻ.

2.1.3 Kết hợp Embedding và Positional Encoding

Cuối cùng, hai ma trận word embedding và positional encoding được cộng lại để tạo thành biểu diễn đầu vào cuối cùng:

### 2.2 Lớp Attention (Attention Layers)

Lớp attention là khối xây dựng cơ bản trong mô hình Transformer, xuất hiện ở cả encoder và decoder. Thành phần cốt lõi trong các lớp này là **self-attention**. Tùy theo vị trí sử dụng (ở encoder hay decoder), sẽ có một số khác biệt nhỏ, tuy nhiên nguyên lý chung vẫn giữ nguyên.

2.2.1 Self-Attention

A diagram of a machine

AI-generated content may be incorrect.

**Nguyên lý hoạt động:**

Tại mỗi thời điểm, mỗi vector đầu vào ​ sẽ được chiếu qua ba ma trận trọng số (có thể học được):

* ​ để sinh ra **query vector** .​
* ​ để sinh ra **key vector** .
* ​ để sinh ra **value vector** .

Các vector này sẽ được kết hợp để tính điểm attention giữa các phần tử trong chuỗi đầu vào.

**Vai trò:**

1. **Query vector ​**: được dùng để tính điểm liên kết với tất cả key vectors trong câu:
2. **Key vector ​**: đóng vai trò xác định mức độ liên quan giữa các query và key để tạo điểm attention.
3. **Value vector ​**: mang thông tin thực tế, được kết hợp với trọng số attention để tạo ra output vector .

A diagram of a computer algorithm

AI-generated content may be incorrect.

**Công thức tính attention:**

Với input ma trận , đầu ra được tính bằng:

2.2.2 Multi-Head Attention

Thay vì chỉ có một self-attention, mô hình Transformer sử dụng nhiều self-attention chạy song song (gọi là các “head”).

**Lợi ích của Multi-head Attention:**

* Mỗi head có thể học các biểu diễn khác nhau trong không gian đặc trưng
* Nắm bắt các khía cạnh khác nhau của thông tin
* Tăng khả năng mô hình hóa các phụ thuộc dài hạn trong chuỗi

**Cách hoạt động:**

Mỗi head sử dụng các ma trận riêng biệt , , sau đó các đầu ra được nối lại và nhân với ma trận ​ để tạo ra output cuối cùng:

A diagram of a diagram

AI-generated content may be incorrect.

2.2.3 Masked Multi-head Attention

Trong decoder, cần đảm bảo rằng mô hình không được “nhìn thấy” các từ tương lai khi dự đoán từ hiện tại. Điều này được xử lý bằng cơ chế **masking**, tạo thành **masked multi-head attention**.

* Áp dụng ma trận mặt nạ , trong đó:
  + Các phần tử tương lai được gán giá trị
  + Các phần tử hiện tại và quá khứ được gán
* Công thức attention sẽ trở thành:

2.2.4. Encoder-Decoder Multi-head Attention

Ở phía decoder, cần học mối quan hệ attention giữa **tập đầu vào (source)** và **tập đầu ra (target)**. Cụ thể:

* Query vector được lấy từ chuỗi target (phía decoder)
* Key và Value được lấy từ chuỗi input (phía encoder)

Lớp attention này giúp decoder tham chiếu trực tiếp đến các thông tin quan trọng từ encoder để dự đoán chính xác hơn tại mỗi bước thời gian.

### 2.3 Residuals và Layer Normalization

Các đầu vào trong Transformer được kết nối tắt (shortcut) đến đầu ra , sau đó hai thành phần này được cộng lại và đưa qua một lớp chuẩn hóa tầng (layer normalization). Toàn bộ quá trình được biểu diễn bằng công thức:

Lớp **layer normalization** giúp đảm bảo rằng mỗi tầng trong mô hình có trung bình bằng 0 và phương sai bằng 1, nhằm ổn định quá trình huấn luyện.

**Công thức tính Layer Normalization**

Với mỗi đơn vị ẩn , đầu ra được chuẩn hóa theo công thức:

Trong đó:

* là gain variable (có thể đặt bằng )
* là giá trị trung bình:
* là độ lệch chuẩn:

**Ý nghĩa**

* Giảm hiện tượng lệch hiệp phương sai (covariance shift):  
  Đây là hiện tượng sự phụ thuộc giữa các tầng thay đổi, khiến việc huấn luyện chậm lại. Layer normalization làm giảm sự thay đổi này và giúp mô hình hội tụ nhanh hơn, giảm số lần lặp cần thiết.
* Khác biệt với Batch Normalization:  
  Batch normalization chuẩn hóa trên toàn bộ một batch, còn layer normalization chuẩn hóa trên từng đơn vị đầu ra của từng mẫu. Điều này khiến layer normalization không phụ thuộc vào kích thước batch, rất hữu ích khi mô hình xử lý một mẫu duy nhất hoặc batch nhỏ.

### 2.4 Positionwise Feed-forward Networks

Sau mỗi lớp attention trong encoder và decoder, Transformer sử dụng một mạng fully connected feed-forward network để xử lý riêng biệt từng vị trí trong chuỗi đầu vào.

Biến đổi này được mô tả bởi công thức:

Trong đó:

* là các ma trận trọng số
* là các vector bias
* là hàm kích hoạt ReLU

Mỗi vị trí (position) trong chuỗi đầu vào sẽ đi qua cùng một mạng feed-forward này, nhưng mỗi vị trí được xử lý độc lập.

### 2.5 Bộ mã hóa (Encoder)

Khối encoder trong Transformer bao gồm lớp lặp lại, mỗi lớp có cấu trúc gồm các thành phần sau:

**Nguyên lý hoạt động:**

* Mỗi lớp multi-head attention trong encoder thực hiện attention nội tại, tức là giữa các phần tử trong chính chuỗi đầu vào (source).
* Điều này giúp mô hình học được mối quan hệ giữa các từ trong câu đầu vào.

**Ứng dụng:**

Theo Tenney và các cộng sự, từng lớp trong encoder có thể tự học để thực hiện các tác vụ NLP truyền thống như:

* Gán nhãn từ loại (Part-of-speech tagging)
* Phân tích cấu trúc câu (Constituents)
* Phụ thuộc cú pháp (Dependencies)
* Nhận diện thực thể (Entity resolution)

### 2.6 Bộ giải mã (Decoder)

Khối decoder cũng bao gồm lớp, nhưng cấu trúc mỗi lớp gồm các thành phần sau:

**Nguyên lý hoạt động:**

* Lớp đầu tiên là **masked multi-head attention**, chỉ cho phép attention đến những token trước đó trong chuỗi đầu ra (target). Điều này đảm bảo mô hình không “nhìn trước” khi dự đoán token tiếp theo.
* Tiếp theo là **encoder-decoder attention**, kết nối thông tin giữa đầu vào và đầu ra. Ở đây, truy vấn (query) đến từ chuỗi đầu ra, trong khi khóa (key) và giá trị (value) đến từ chuỗi đầu vào.
* Sau đó là mạng FFN và các bước chuẩn hóa tương tự như trong encoder.

# VI. Federated Learning

Nguồn tham khảo

* Sahoo J. Federated Learning. Principles, Paradigms, and Applications 2025
* Krishnan S. Handbook on Federated Learning. Advances, App and Opportunities 2024
* Jin Y. Federated Learning. Fundamentals and Advances 2022

## 1. Tổng quan

### 1.1 Giới thiệu

Federated Learning là phương pháp khắc phục những hạn chế của cách học truyền thống. Khác với phương pháp cũ, các máy khách (client) không cần chia sẻ dữ liệu với máy chủ (server). Thay vào đó, họ giữ lại dữ liệu của mình để huấn luyện. Một số tiêu chí để lựa chọn tham gia bao gồm thiết bị thông minh, băng thông, v.v. Thông thường, tất cả máy khách đều có thể tham gia, nhưng cũng có một số ngoại lệ.

Trong kỹ thuật này, các máy khách tham gia vào quá trình học để phát triển mô hình cục bộ (local model), sau đó gửi mô hình này đến máy chủ trung tâm. Máy chủ nhận tất cả các mô hình cục bộ và tổng hợp chúng thành một mô hình chung (global model). Mô hình chung này được gửi lại cho tất cả các máy khách tham gia. Các máy khách cập nhật mô hình của họ dựa trên mô hình chung và gửi lại cho máy chủ để tiếp tục tổng hợp. Quá trình này lặp đi lặp lại cho đến khi thu được một mô hình tốt nhất phù hợp với tất cả dữ liệu.

### 1.2 Ví dụ chi tiết

Gọi là vòng lặp cục bộ của máy khách và là vòng lặp tổng hợp mô hình của máy chủ. là tập hợp các mô hình cục bộ do các máy khách phát triển. Cụ thể:

-

-

-

- ...

-

Giả sử có máy khách tham gia. Sau mỗi vòng lặp , máy chủ tổng hợp mô hình bằng công thức:

Mô hình tổng hợp này được sử dụng cho các vòng lặp tiếp theo. Quá trình này tiếp tục cho đến khi đạt được vòng lặp thứ . Mô hình cuối cùng được dùng để cập nhật các mô hình cục bộ, đảm bảo phù hợp với tất cả dữ liệu.

### 1.3 Ưu điểm của Federated Learning

- Hỗ trợ ra quyết định với khả năng tái sử dụng rộng rãi – Mô hình học được có thể áp dụng cho nhiều tác vụ khác nhau.

- Bảo vệ dữ liệu người dùng – Thay vì chia sẻ toàn bộ dữ liệu, máy khách chỉ gửi mô hình đã huấn luyện đến máy chủ, giảm nguy cơ rò rỉ thông tin.

### 1.4 Nhược điểm của Federated Learning

- Sự khác biệt giữa các thiết bị – Các máy khách có thể có cấu hình phần cứng, hệ điều hành hoặc dữ liệu khác nhau, gây khó khăn trong việc đồng bộ hóa.

- Giới hạn băng thông mạng – Việc truyền tải mô hình giữa máy khách và máy chủ có thể gặp trở ngại nếu đường truyền không ổn định hoặc chậm.

### 1.5 Các bước chính trong Federated Learning

a) Huấn luyện mô hình cục bộ – Mỗi máy khách tự đào tạo mô hình bằng dữ liệu riêng của mình.

b) Mã hóa và gửi mô hình lên máy chủ – Máy khách mã hóa mô hình đã huấn luyện để đảm bảo an toàn trước khi gửi đi.

c) Tổng hợp mô hình toàn cục – Máy chủ nhận các mô hình từ máy khách và kết hợp chúng thành một mô hình chung (global model).

d) Phân phối mô hình toàn cục – Máy chủ gửi mô hình tổng hợp về các máy khách để tiếp tục cải thiện trong các vòng lặp tiếp theo.

## 2. Phân loại

Dựa trên cách lựa chọn mẫu từ tập dữ liệu, Federated Learning được chia thành 3 loại chính:

1. Federated Learning ngang (Horizontal Federated Learning)

2. Federated Learning dọc (Vertical Federated Learning)

3. Federated Learning chuyển giao (Federated Transfer Learning)

Quy ước thuật ngữ

- : Đại diện cho không gian đặc trưng (feature) và nhãn dữ liệu.

- : Không gian nhãn của các máy khách (client).

- : Tập dữ liệu của máy khách.

- : ID không gian mẫu (định danh các mẫu dữ liệu).

- Tập dữ liệu huấn luyện bao gồm:

+) : Các đặc trưng (features).

+) : Nhãn (labels).

+) : Định danh mẫu (sample IDs).

### 2.1 Horizontal Federated Learning

- Khái niệm

Trong Horizontal Federated Learning, các máy khách (client) sử dụng các mẫu dữ liệu khác nhau nhưng có cùng đặc trưng (features) để huấn luyện. Cách tiếp cận này còn được gọi là Federated Learning dựa trên mẫu (sample-based federated learning) hoặc Federated Learning đồng nhất (homogeneous federated learning).

- Công thức biểu diễn

* Các máy khách có cùng không gian đặc trưng (features giống nhau).
* : Nhãn dữ liệu (labels) giống nhau.
* : Các mẫu dữ liệu (samples) khác nhau giữa các máy khách.

- Ví dụ

A table with numbers and text

AI-generated content may be incorrect.

Dòng dữ liệu đại diện cho thông tin cá nhân của người dùng, bao gồm tuổi, giới tính, chiều cao, cân nặng và nhãn tương ứng. Tập dữ liệu cục bộ của máy khách 1 và máy khách 2 có cùng không gian đặc trưng, nhưng nhóm người dùng lại khác nhau.

### 2.2 Vertical Federated Learning

- Khái niệm

Federated Learning Dọc là phương pháp Federated Learning dựa trên đặc trưng (feature-based). Trong phương pháp này:

* Các máy khách (client) sử dụng cùng một tập mẫu dữ liệu (samples) nhưng với các đặc trưng (features) khác nhau.
* Còn được gọi là Federated Learning dị thể (heterogeneous federated learning).

- Công thức biểu diễn

* : Các máy khách có đặc trưng khác nhau.
* : Nhãn dữ liệu (labels) có thể khác nhau.
* : Cùng tập mẫu dữ liệu (ví dụ: cùng người dùng nhưng dữ liệu từ nhiều nguồn).

- Ví dụ: A table with numbers and text

AI-generated content may be incorrect.

Client 1 và Client 2 cùng có dữ liệu về các cá nhân (Person A, B, C,...) nhưng mỗi bên chỉ lưu trữ một phần thông tin khác nhau:

* Client 1: Giữ nhãn (label) và một số đặc trưng (tuổi, giới tính).
* Client 2: Chỉ giữ các đặc trưng khác (chiều cao, cân nặng) không có nhãn.

### 2.3 Federated Transfer Learning

- Khái niệm

Federated Learning Chuyển Giao là phương pháp áp dụng khi:

* Các mẫu dữ liệu có ID khác nhau (khác đối tượng)
* Đặc trưng dữ liệu khác nhau (khác thuộc tính)

Phương pháp này có một số điểm tương đồng với Federated Learning Dọc nhưng khác biệt trong cách tính toán gradient.

- Công thức biểu diễn

* : Các máy khách có đặc trưng hoàn toàn khác nhau.
* : Nhãn dữ liệu khác nhau.
* : Mẫu dữ liệu từ các đối tượng khác nhau.

- Ví dụ minh họa

Một ngân hàng ở Anh và một công ty chứng khoán ở Trung Quốc có dữ liệu khách hàng khác nhau về dịch vụ và đối tượng, nhưng muốn hợp tác xây dựng mô hình AI chung mà không chia sẻ dữ liệu gốc.

## 3. Các thành phần của Federated Learning

Có 5 thành phần chính trong Federated Learning:

* Máy khách (Client)
* Máy chủ trung tâm (Central server)
* Fetching global model
* Local model training
* Model aggregation

### 3.1 Máy khách (Client)

Trong Federated Learning (federated learning), máy khách là một thiết bị chứa tập dữ liệu đã được xử lý. Điểm đặc biệt là các máy khách không cần gửi tập dữ liệu thô lên máy chủ.

Thay vào đó, mỗi máy khách sẽ tự xây dựng một mô hình cục bộ (local model) từ chính tập dữ liệu của mình. Do đó, các mô hình này sẽ khác nhau giữa các máy khách.

### 3.2. Máy chủ trung tâm (Central Server)

Máy chủ trung tâm là một trong những thành phần chính của Federated Learning. Mô hình cục bộ do máy khách phát triển không thể áp dụng chung cho tất cả các tập dữ liệu, do có sự khác biệt về mẫu dữ liệu giữa các máy khách.

Nhiệm vụ của máy chủ trung tâm là tạo ra một mô hình thống nhất có thể áp dụng cho tất cả. Điều này được thực hiện bằng cách tổng hợp tất cả các mô hình cục bộ nhận được từ các máy khách, từ đó xây dựng nên một mô hình toàn cục (global model).

### 3.3 Thu nhận mô hình toàn cục (Fetching Global Model)

Các máy khách tham gia huấn luyện duy trì trạng thái hoạt động và lưu giữ mô hình cục bộ đã được đào tạo. Mô hình toàn cục là kết quả được máy chủ trung tâm tạo ra thông qua việc tổng hợp các mô hình cục bộ  từ các máy khách. Sau đó, mô hình toàn cục này sẽ được gửi ngược lại cho các máy khách để họ cập nhật mô hình của mình.

### 3.4 Huấn luyện mô hình cục bộ (Local Model Training)

Quá trình huấn luyện mô hình cục bộ diễn ra phía máy khách. Mỗi máy khách duy trì tập dữ liệu riêng để huấn luyện. Việc lựa chọn mẫu dữ liệu đã được thảo luận ở phần trước. Do các máy khách sử dụng tập dữ liệu khác nhau nên các mô hình họ phát triển cũng khác biệt. Những mô hình này không thể áp dụng chung cho mọi tập dữ liệu.

Để giải quyết vấn đề này, chúng tôi sử dụng mô hình toàn cục M = M - kết quả tổng hợp từ các mô hình cục bộ. Trong quá trình huấn luyện cục bộ, gradient được tính toán bằng công thức:

Trong đó:

* : số hiệu batch (batch number)
* : định danh máy khách

Sau khi huấn luyện, các mô hình  được gửi đến máy chủ để tổng hợp.

### 3.5 Tổng hợp mô hình (Model Aggregation)

Tổng hợp mô hình là một quy trình quan trọng được thực hiện bởi máy chủ trung tâm. Máy chủ sẽ tạo ra một mô hình tổng hợp (gọi là mô hình toàn cục) từ các mô hình cục bộ  nhận được từ các máy khách. Mô hình toàn cục này đóng vai trò là phiên bản cập nhật dùng chung cho tất cả các máy khách đã tham gia quá trình huấn luyện.

## 4. Kỹ thuật tổng hợp mô hình (Model Aggregation)

Kỹ thuật tổng hợp được thực hiện bằng cách tính trung bình các hệ số (coefficients) của mô hình mạng nơ-ron từ các máy khách. Trong Federated Learning, kỹ thuật này giúp:

* Cập nhật mô hình cục bộ trên từng thiết bị.
* Máy chủ trung tâm đóng vai trò tổng hợp và phân phối lại mô hình toàn cục.

**Thuật toán 1: Huấn luyện Cục bộ trong Federated Learning**

A screenshot of a computer

AI-generated content may be incorrect.

Thuật toán này mô tả quy trình huấn luyện mô hình mạng nơ-ron (Neural Network - NN) ở phía các máy khách (clients) trong một hệ thống Federated Learning. Mục tiêu là để mỗi máy khách tận dụng dữ liệu cục bộ của mình để cập nhật mô hình, sau đó gửi các cập nhật này về máy chủ trung tâm để tổng hợp.

Ký hiệu:

* : Tổng số lượng máy khách tham gia vào quá trình huấn luyện.
* : Chỉ số của một máy khách cụ thể ().
* : Kích thước của mini-batch, là kích thước của các phần nhỏ dữ liệu được sử dụng trong mỗi bước cập nhật gradient.
* : Số lượng local epochs, là số lần mỗi máy khách lặp lại việc huấn luyện trên toàn bộ dữ liệu cục bộ của mình.
* ​: Tập dữ liệu được phân bổ cho máy khách thứ .
* : Mô hình cục bộ của máy khách thứ . Tập hợp tất cả các mô hình cục bộ được ký hiệu là .
* : Tốc độ học (learning rate), một siêu tham số quyết định độ lớn của bước cập nhật trọng số trong quá trình tối ưu hóa.

Đầu vào:

* Mô hình toàn cầu hoặc mô hình cập nhật từ máy chủ trung tâm, ký hiệu là . Đây là mô hình ban đầu hoặc mô hình đã được tổng hợp từ các vòng huấn luyện trước đó, được phân phối cho các máy khách.

Đầu ra:

* Mô hình đã được huấn luyện cục bộ của máy khách thứ sau local epochs, ký hiệu là .

Các bước thực hiện:

1. Phân phối Mô hình (Bước 1-3):
   * Máy chủ trung tâm gửi mô hình hiện tại () đến máy khách thứ .
   * Máy khách thứ có thể thực hiện một bước cập nhật cục bộ ban đầu dựa trên mô hình nhận được ().
2. Chuẩn bị Dữ liệu (Bước 4):
   * Máy khách thứ chia tập dữ liệu cục bộ của mình (​) thành các mini-batch có kích thước . Ký hiệu đại diện cho một mini-batch.
3. Huấn luyện Cục bộ (Bước 5):
   * Máy khách thứ thực hiện một vòng lặp qua local epochs.
   * Trong mỗi local epoch (từ đến ):
     + Máy khách lặp qua tất cả các mini-batch trong ​.
     + Đối với mỗi mini-batch , máy khách cập nhật mô hình cục bộ của mình ( hoặc các trọng số ). Quá trình cập nhật thường sử dụng thuật toán gradient descent. Các công thức và biểu thị việc cập nhật mô hình dựa trên gradient của hàm mất mát tính trên mini-batch , nhân với tốc độ học . và đại diện cho gradient (đạo hàm) của hàm mất mát theo các tham số của mô hình.
4. Kết thúc Huấn luyện Cục bộ (Bước 6): Sau khi hoàn thành local epochs, quá trình huấn luyện cục bộ trên máy khách kết thúc.
5. Gửi Mô hình Đã Huấn luyện (Bước 7): Máy khách thứ gửi mô hình đã được huấn luyện cục bộ () trở lại máy chủ trung tâm.

### 4.1 All Model Averaging (AMA) - Trung bình cộng tất cả mô hình

Kỹ thuật AMA là một phương pháp cơ bản và phổ biến trong Federated Learning để tổng hợp các mô hình được huấn luyện cục bộ từ nhiều máy khách. Ý tưởng cốt lõi của AMA là lấy trung bình các hệ số (trọng số) tương ứng của tất cả các mô hình cục bộ để tạo ra một mô hình toàn cầu duy nhất.

Quy trình thực hiện:

1. Huấn luyện Cục bộ: Trong giai đoạn huấn luyện, mỗi máy khách sử dụng Thuật toán 1 (đã được mô tả trước đó) để phát triển mô hình cục bộ của riêng mình dựa trên dữ liệu cục bộ của họ. Sau khi hoàn thành quá trình huấn luyện cục bộ, mỗi máy khách sẽ gửi mô hình đã được huấn luyện này đến máy chủ trung tâm.
2. Tổng hợp tại Máy chủ Trung tâm: Khi máy chủ trung tâm nhận được các mô hình cục bộ từ tất cả các máy khách tham gia, nó sẽ tiến hành quá trình tổng hợp bằng cách sử dụng Thuật toán 2 (cũng đã được mô tả). Thuật toán này thực hiện việc tính tổng và sau đó lấy trung bình các trọng số tương ứng của các nơ-ron trong các mô hình cục bộ.
3. Tạo Mô hình Tổng hợp: Kết quả của quá trình tổng hợp là một mô hình toàn cầu được gọi là mô hình tổng hợp (aggregate model). Mô hình này chứa các trọng số là giá trị trung bình của các trọng số tương ứng từ tất cả các mô hình cục bộ.
4. Phân phối Mô hình Tổng hợp: Mô hình tổng hợp này sau đó được gửi trả lại cho các máy khách đã tham gia vào quá trình huấn luyện.
5. Cập nhật Mô hình Cục bộ: Các máy khách sẽ sử dụng mô hình tổng hợp nhận được để cập nhật mô hình cục bộ của họ. Quá trình này thường là bước khởi đầu cho vòng huấn luyện liên kết tiếp theo, giúp các mô hình cục bộ tiếp tục học hỏi dựa trên kiến thức tổng hợp từ toàn bộ hệ thống.

Công thức toán học:

Giải thích công thức:

* : Đại diện cho mô hình tổng hợp cuối cùng được tạo ra.
* : Ký hiệu cho thao tác tổng hợp bằng phương pháp trung bình cộng tất cả mô hình.
* : Tổng số lượng mô hình cục bộ (từ n máy khách).
* : Số lượng nơ-ron trong mỗi lớp của mô hình.
* : Chỉ số của lớp trong mô hình.
* : Chỉ số của nơ-ron trong lớp i.
* ​: Trọng số của nơ-ron thứ ở lớp thứ trong mô hình cục bộ của máy khách thứ .
* Công thức này thực hiện việc tính tổng tất cả các trọng số tương ứng (cùng vị trí nơ-ron và lớp) từ tất cả mô hình cục bộ, và sau đó chia cho tổng số mô hình () để lấy giá trị trung bình. Giá trị trung bình này sẽ trở thành trọng số tương ứng trong mô hình tổng hợp .

**Thuật toán 2: Tổng hợp Mô hình (Average of Model Aggregation - AMA)**

A screenshot of a white sheet of paper

AI-generated content may be incorrect.

Thuật toán này mô tả quy trình tổng hợp các mô hình đã được huấn luyện cục bộ từ các máy khách để tạo ra một mô hình toàn cầu được cập nhật. Trong nghiên cứu này, máy chủ trung tâm nhận được mô hình đã huấn luyện từ bốn máy khách, ký hiệu là , trong đó ​ là mô hình đã được huấn luyện của máy khách thứ .

Ký hiệu:

* : Cấu trúc của mô hình toàn cầu, với đại diện cho các trọng số của mô hình.
* : Tập hợp các mô hình đã được huấn luyện từ máy khách.
* : Số lượng mô hình cục bộ được sử dụng để tổng hợp.
* ​: Trọng số thứ ở lớp thứ của mô hình ​ (mô hình thứ trong danh sách các mô hình cục bộ).
* : Số lượng lớp trong mô hình mạng nơ-ron.
* : Số lượng nơ-ron (hoặc tham số) trong mỗi lớp (chúng ta giả định rằng tất cả các mô hình cục bộ đều có cấu trúc giống nhau).
* : Giá trị trung bình của các trọng số tương ứng từ tất cả các mô hình cục bộ.
* : Mô hình toàn cầu đã được tổng hợp hoặc cập nhật.

Đầu vào:

* Các mô hình đã được huấn luyện từ các máy khách: .

Đầu ra:

* Mô hình toàn cầu đã được tổng hợp hoặc cập nhật: .

Các bước thực hiện:

1. Nhận Mô hình Cục bộ (Bước 1): Máy chủ trung tâm nhận được các mô hình đã được huấn luyện từ tất cả các máy khách tham gia.
2. Tạo Danh sách Mô hình (Bước 2): Máy chủ tạo một danh sách chứa tất cả các mô hình cục bộ đã nhận được: .
3. Xác định Số Lượng Mô hình (Bước 3): Máy chủ xác định tổng số mô hình cục bộ sẽ được sử dụng để tổng hợp.
4. Tiến hành Tổng hợp (Bước 4):
   * Máy chủ lặp qua tất cả các lớp của mô hình (với chỉ số từ đến ).
   * Đối với mỗi lớp, máy chủ lặp qua tất cả các nơ-ron (hoặc tham số) trong lớp đó (với chỉ số từ đến ).
   * Đối với mỗi vị trí trọng số (), máy chủ tính giá trị trung bình của các trọng số tương ứng từ tất cả các mô hình cục bộ:
   * Sau đó, máy chủ tính giá trị trung bình của các trọng số này bằng cách chia tổng cho số lượng mô hình ():

Giá trị trung bình này sẽ trở thành trọng số mới cho vị trí () trong mô hình toàn cầu được cập nhật.

1. Kết thúc Tổng hợp (Bước 5): Sau khi lặp qua tất cả các lớp và tất cả các trọng số trong mỗi lớp, mô hình toàn cầu đã được tổng hợp hoàn chỉnh.
2. Gửi Mô hình Toàn cầu (Bước 6): Máy chủ trung tâm gửi mô hình toàn cầu đã được tổng hợp trở lại cho các máy khách để họ sử dụng trong vòng huấn luyện tiếp theo.

### 4.2 One Model Selection (OMS) - Lựa chọn một mô hình

Kỹ thuật OMS là một phương pháp tổng hợp mô hình trong Federated Learning dựa trên hiệu suất của các mô hình cục bộ. Thay vì kết hợp thông tin từ tất cả các mô hình, OMS chọn ra duy nhất một mô hình có hiệu suất tốt nhất để làm mô hình toàn cầu hoặc để phân phối cho vòng huấn luyện tiếp theo.

Quy trình thực hiện:

1. Huấn luyện Cục bộ: Tương tự như AMA, trong giai đoạn huấn luyện, mỗi máy khách phát triển mô hình cục bộ của riêng mình.
2. Gửi Mô hình và Hiệu suất: Điểm khác biệt quan trọng của OMS là khi các máy khách gửi mô hình đã được huấn luyện đến máy chủ trung tâm, họ cũng gửi kèm theo thông tin về hiệu suất của mô hình đó. Hiệu suất này thường được đo bằng độ chính xác (accuracy) trên một tập dữ liệu cục bộ hoặc một tập dữ liệu kiểm tra chung nhỏ mà máy khách có thể truy cập.
3. Lựa chọn Mô hình Tốt nhất: Máy chủ trung tâm sẽ so sánh hiệu suất của tất cả các mô hình cục bộ nhận được. Dựa trên tiêu chí hiệu suất (thường là độ chính xác cao nhất), máy chủ sẽ chọn ra mô hình có hiệu suất tốt nhất trong số đó.
4. Sử dụng Mô hình Được Chọn: Mô hình được chọn sẽ được coi là mô hình toàn cầu (global model) cho vòng liên kết hiện tại.
5. Phân phối Mô hình Toàn cầu: Mô hình toàn cầu được chọn sẽ được gửi trở lại cho các máy khách để họ sử dụng làm điểm khởi đầu để huấn luyện mô hình cục bộ của mình trong vòng tiếp theo.

**Thuật toán 3: Tổng hợp Mô hình (Model Aggregation - OMS - One Model Selection)**

A screenshot of a computer

AI-generated content may be incorrect.

Thuật toán này mô tả quy trình tổng hợp các mô hình đã được huấn luyện cục bộ từ các máy khách bằng cách chọn ra một mô hình duy nhất được coi là tốt nhất dựa trên hiệu suất của nó. Trong nghiên cứu này, máy chủ trung tâm nhận được mô hình đã huấn luyện từ bốn máy khách, ký hiệu là .

Ký hiệu:

* : Cấu trúc của mô hình toàn cầu, với đại diện cho các trọng số của mô hình.
* : Tập hợp các mô hình đã được huấn luyện từ n máy khách.
* : Độ chính xác (accuracy) hoặc một số chỉ số đánh giá hiệu suất khác của mô hình ​ (mô hình của máy khách thứ ) trên một tập dữ liệu nào đó (có thể là tập kiểm tra cục bộ hoặc một tập kiểm tra chung nhỏ mà máy chủ có).
* : Danh sách chứa các giá trị hiệu suất của tất cả các mô hình cục bộ.
* : Hàm trả về giá trị lớn nhất trong danh sách , tức là độ chính xác cao nhất trong số các mô hình cục bộ.
* : Mô hình toàn cầu được chọn.

Đầu vào:

* Các mô hình đã được huấn luyện từ các máy khách: .

Đầu ra:

* Mô hình toàn cầu được chọn: , đây chính là mô hình cục bộ có hiệu suất tốt nhất.

Các bước thực hiện:

1. Nhận Mô hình Cục bộ (Implicit): Máy chủ trung tâm nhận được các mô hình đã được huấn luyện từ tất cả các máy khách tham gia.
2. Thu thập Hiệu suất Mô hình (Bước 2): Máy chủ thu thập hoặc tính toán hiệu suất của từng mô hình cục bộ. Các giá trị hiệu suất này được lưu trữ trong danh sách .
3. Lựa chọn Mô hình Tốt nhất (Bước 3): Máy chủ tìm ra mô hình có hiệu suất cao nhất bằng cách xác định giá trị lớn nhất trong danh sách . Giả sử ​ là giá trị lớn nhất trong , thì mô hình ​ sẽ được chọn làm mô hình toàn cầu.
4. Gửi Mô hình Toàn cầu (Bước 4): Máy chủ trung tâm gửi mô hình toàn cầu đã được chọn (chính là mô hình của một trong các máy khách) trở lại cho các máy khách để họ sử dụng trong vòng huấn luyện tiếp theo.

### 4.3 Best Model Average (BMA) - Trung bình cộng các mô hình tốt nhất

Kỹ thuật BMA tính toán trung bình của các mô hình được coi là "tốt nhất" trong số các mô hình cục bộ được huấn luyện bởi các máy khách.

Quy trình thực hiện:

1. Huấn luyện Cục bộ: Tương tự như AMA và OMS, mỗi máy khách huấn luyện mô hình cục bộ của riêng mình.
2. Gửi Mô hình và Hiệu suất: Các máy khách không chỉ gửi mô hình đã huấn luyện mà còn gửi kèm theo thông tin về hiệu suất mô hình (thường là độ chính xác) cho máy chủ trung tâm.
3. Lựa chọn Các Mô hình Tốt nhất: Máy chủ trung tâm sẽ sử dụng thông tin về hiệu suất để sắp xếp các mô hình cục bộ theo thứ tự hiệu suất (ví dụ: từ cao đến thấp). Sau đó, máy chủ sẽ chọn ra một số lượng nhất định các mô hình có hiệu suất cao nhất để tiến hành tổng hợp.
4. Tính Trung bình: Quá trình tính trung bình các trọng số của các mô hình tốt nhất được chọn tương tự như kỹ thuật AMA. Máy chủ sẽ lấy trung bình các trọng số tương ứng của các nơ-ron trong các mô hình tốt nhất này để tạo ra mô hình toàn cầu.
5. Phân phối Mô hình Toàn cầu: Mô hình toàn cầu được tạo ra bằng cách lấy trung bình các mô hình tốt nhất sẽ được gửi trở lại cho các máy khách để cập nhật mô hình cục bộ của họ.

Công thức toán học:

Giải thích công thức :

* : Mô hình tổng hợp được tạo ra bằng kỹ thuật BMA.
* : Ký hiệu cho thao tác tổng hợp bằng phương pháp trung bình cộng các mô hình tốt nhất.
* : Tổng số lượng mô hình cục bộ.
* : Số lượng nơ-ron trong mỗi lớp.
* : Chỉ số của lớp.
* : Chỉ số của nơ-ron trong lớp .
* ​: Trọng số của nơ-ron thứ ở lớp thứ trong một trong các mô hình được chọn là "tốt nhất".
* : Số lượng mô hình tốt nhất được chọn để tính trung bình.

**Thuật toán 4: Tổng hợp Mô hình BMA**

A white paper with black text

AI-generated content may be incorrect.

Thuật toán này mô tả một phương pháp tổng hợp các mô hình đã được huấn luyện cục bộ từ các máy khách bằng cách lấy trung bình của một phần các mô hình "tốt nhất" sau khi đã sắp xếp chúng. Trong nghiên cứu này, máy chủ trung tâm nhận được mô hình đã huấn luyện từ bốn máy khách, ký hiệu là .

Ký hiệu:

* : Cấu trúc của mô hình toàn cầu, với đại diện cho các trọng số của mô hình.
* : Tập hợp các mô hình đã được huấn luyện từ n máy khách.
* : Một danh sách chứa các mô hình cục bộ, được sắp xếp dựa trên hiệu suất của chúng (ví dụ: độ chính xác).
* : Số lượng mô hình cục bộ được sử dụng để tổng hợp.
* : Số lượng mô hình tốt nhất được chọn để tính trung bình.
* ​: Trọng số thứ ở lớp thứ của mô hình (Mô hình thứ trong danh sách đã sắp xếp ).
* : Số lượng lớp trong mô hình mạng nơ-ron.
* : Số lượng nơ-ron trong mỗi lớp (giả định rằng tất cả các mô hình cục bộ đều có cấu trúc giống nhau).
* : Trọng số thứ ở lớp thứcủa mô hình toàn cầu được cập nhật.
* : Mô hình toàn cầu đã được tổng hợp hoặc cập nhật.

Đầu vào:

* Các mô hình đã được huấn luyện từ các máy khách: .

Đầu ra:

* Mô hình toàn cầu đã được tổng hợp hoặc cập nhật: .

Các bước thực hiện:

1. Nhận Mô hình Cục bộ (Implicit): Máy chủ trung tâm nhận được các mô hình đã được huấn luyện từ tất cả các máy khách tham gia.
2. Sắp xếp Mô hình (Bước 2): Máy chủ sắp xếp các mô hình cục bộ trong danh sách A dựa trên hiệu suất của chúng (ví dụ: độ chính xác giảm dần).
3. Xác định Số Lượng Mô hình Tốt Nhất (Bước 3 & 4): Máy chủ xác định số lượng mô hình "tốt nhất" để lấy trung bình, . Điều này có nghĩa là nếu có n mô hình, thuật toán sẽ chọn lấy trung bình của mô hình có hiệu suất cao nhất.
4. Tiến hành Tổng hợp (Bước 5):
   * Máy chủ lặp qua tất cả các lớp của mô hình (với chỉ số từ đến ).
   * Đối với mỗi lớp, máy chủ lặp qua tất cả các nơ-ron trong lớp đó (với chỉ số từ đến ).
   * Đối với mỗi vị trí trọng số (i,j), máy chủ tính giá trị trung bình của các trọng số tương ứng từ k mô hình "tốt nhất" trong danh sách đã sắp xếp A:

Trong đó đến là mô hình đầu tiên (tốt nhất) trong danh sách đã sắp xếp .

1. Kết thúc Tổng hợp (Bước 6): Sau khi lặp qua tất cả các lớp và tất cả các trọng số trong mỗi lớp, mô hình toàn cầu đã được tổng hợp hoàn chỉnh.
2. Gửi Mô hình Toàn cầu (Bước 7): Máy chủ trung tâm gửi mô hình toàn cầu đã được tổng hợp trở lại cho các máy khách để họ sử dụng trong vòng huấn luyện tiếp theo.