BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

---------- ☙📖❧ ----------

BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC

NGHIÊN CỨU MỘT SỐ THUẬT TOÁN ADABOOST VÀ ỨNG DỤNG CHO BÀI TOÁN PHÂN LỚP

Sinh viên thực hiện: Nguyễn Trung Đức
Lớp, khoa: Khoa học máy tính 02 - Khoa CNTT
Sinh viên thực hiện: Đinh Minh Đại
Lớp, khoa: Khoa học máy tính 02 - Khoa CNTT

Người hướng dẫn: TS. Trần Hùng Cường

Hà Nội – 05/2024

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

---------- ☙📖❧ ----------

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC

NGHIÊN CỨU MỘT SỐ THUẬT TOÁN ADABOOST VÀ ỨNG DỤNG CHO BÀI TOÁN PHÂN LỚP

Sinh viên thực hiện: Nguyễn Trung Đức **Nam/Nữ: Nam**

Dân tộc: Kinh

Lớp, khoa: Khoa học máy tính 02 - Khoa CNTT **Năm thứ: 4/4**

Ngành học: Khoa học máy tính

Sinh viên thực hiện: Đinh Minh Đại Nam/Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: Khoa học máy tính 02 - Khoa CNTT Năm thứ: 4/4

Ngành học: Khoa học máy tính

Người hướng dẫn: TS. Trần Hùng Cường

Hà Nội – 05/2024

MỤC LỤC

LỜI CẢM ƠN

Trước khi đến với nội dung chính của bản báo cáo này, nhóm em xin được gửi lời cảm ơn đến thầy Trần Hùng Cường khoa công nghệ thông tin vì đã hướng dẫn tận tình cả nhóm trong quá trình hoàn thành bản báo cáo. Những đóng góp, ý kiến của thầy chỉ ra được những điểm hạn chế của báo cáo và từ đó nhóm thay đổi để bản báo cáo này càng được hoàn thiện hơn.

Bên cạnh đó, nhóm cũng gửi lời cảm ơn đến những người đã giúp nhóm trong quá trình tìm kiếm thông tin liên quan đến đề tài.

Mặc dù là những sinh viên năm 4 nhưng do thời gian dành cho việc thực hiện nghiên cứu khoa học chưa được nhiều nên trong quá trình làm bài chắc chắn khó tránh khỏi những thiếu sót. Do đó, nhóm kính mong nhận được những lời góp ý của thầy/cô để bản báo cáo này ngày càng hoàn thiện hơn.

Xin trân trọng cảm ơn!

LỜI NÓI ĐẦU

Học máy (Machine Learning) là một công nghệ đang phát triển cho phép máy tính tự học hỏi từ dữ liệu lịch sử. Học máy sử dụng nhiều kỹ thuật khác nhau để tạo ra các mô hình toán học và đưa ra dự đoán dựa trên thông tin hoặc dữ liệu trước đó. Hiện nay, nó được sử dụng cho nhiều thứ khác nhau, bao gồm hệ thống đề xuất (Recommender systems), lọc email (email filtering), tự động gắn thẻ trên Facebook (Facebook auto-tagging), nhận dạng hình ảnh (image identification) và nhận dạng giọng nói (speech recognition).

Học kết hợp (Ensemble Learning) là một kỹ thuật được sử dụng để cải thiện hiệu suất của các mô hình học máy. Kỹ thuật này dựa trên cơ sở là kết hợp nhiều mô hình học yếu để tạo ra một mô hình mạnh mẽ hơn.

Boosting là một trong những kỹ thuật học kết hợp nổi tiếng nhất. AdaBoost (Adaptive Boosting) là một thuật toán Boosting được sử dụng rộng rãi để cải thiện hiệu suất của các mô hình học yếu. Thuật toán này hoạt động bằng cách lặp đi lặp lại việc đào tạo một bộ học yếu trên các tập dữ liệu được chọn cẩn thận và kết hợp các dự đoán của chúng theo một cách có trọng số.

Với mong muốn khám phá tiềm năng của AdaBoost trong bài toán phân lớp, nhóm quyết định đã lựa chọn đề tài "**Nghiên cứu một số thuật toán AdaBoost và ứng dụng cho bài toán phân lớp**". Báo cáo này tập trung nghiên cứu và ứng dụng các thuật toán AdaBoost cho các bài toán phân lớp thực tế.

Báo cáo được chia thành 3 chương chính:

- **Chương 1: Giới thiệu chung về học máy và học kết hợp**

Chương này cung cấp những kiến thức nền tảng về học máy, bài toán phân lớp, học kết hợp, và giới thiệu về AdaBoost.

- **Chương 2: Thuật toán AdaBoost**

Chương này trình bày chi tiết về thuật toán AdaBoost và các biến thể của nó như AdaBoost.M1, AdaBoost.MH và AdaBoost.MR.

- **Chương 3: Ứng dụng cho bài toán phân lớp**

Chương này trình bày kết quả thực nghiệm của việc ứng dụng các thuật toán AdaBoost vào các bài toán phân lớp thực tế. Các chỉ số đánh giá hiệu suất của mô hình sẽ được phân tích và so sánh với các phương pháp khác để khẳng định tính hiệu quả của mô hình.

Hy vọng rằng qua ba chương của báo cáo này, độc giả sẽ có cái nhìn tổng quan về bài toán phân lớp, cũng như tiềm năng của AdaBoost trong việc giải quyết bài toán này. Báo cáo cũng kỳ vọng đóng góp vào việc phát triển những phương pháp phân lớp tự động hiệu quả hơn trong tương lai.

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ NGỮ VIẾT TẮT

SVM	Support Vector Machine
KNN	K-Nearest Neighbors
AdaBoost	Adaptive Boosting

MỞ ĐẦU

Lý do chọn đề tài

Học máy (Machine Learning) là một lĩnh vực nghiên cứu sôi động và có nhiều ứng dụng trong thực tế. Trong đó, bài toán phân lớp là một bài toán cơ bản và có nhiều ứng dụng trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, phân tích dữ liệu, v.v. Học kết hợp (Ensemble Learning) là một kỹ thuật được sử dụng để cải thiện hiệu suất của các mô hình học máy. Kỹ thuật này dựa trên cơ sở là kết hợp nhiều mô hình học yếu để tạo ra một mô hình mạnh mẽ hơn. AdaBoost (Adaptive Boosting) là một thuật toán Boosting được sử dụng rộng rãi để cải thiện hiệu suất của các mô hình học yếu.

Với mong muốn khám phá tiềm năng của AdaBoost trong bài toán phân lớp, nhóm quyết định lựa chọn đề tài "Nghiên cứu một số thuật toán AdaBoost và ứng dụng cho bài toán phân lớp".

Mục tiêu của đề tài

Đề tài này có những mục tiêu chính sau:

- Nghiên cứu và trình bày chi tiết về thuật toán AdaBoost và các biến thể như AdaBoost.M1, AdaBoost.MH và AdaBoost.MR.
- Ứng dụng các thuật toán AdaBoost vào các bài toán phân lớp thực tế, cụ thể là phân loại dữ liệu Marketing and Sales, phân loại đậu khô, phân loại đánh giá Yelp và dự đoán vị trí protein trong tế bào Yeast.
- Phân tích và so sánh hiệu suất của mô hình AdaBoost với các phương pháp phân lớp khác, từ đó khẳng định tính hiệu quả của mô hình.

Phương pháp nghiên cứu

Để đạt được mục tiêu đề ra, nhóm sử dụng các phương pháp nghiên cứu sau:

- *Nghiên cứu lý thuyết*: Tìm hiểu và phân tích các tài liệu khoa học liên quan đến thuật toán AdaBoost và các biến thể của nó.
- *Thực nghiệm*: Xây dựng và đánh giá hiệu suất của mô hình AdaBoost trên các bộ dữ liệu thực tế. Sử dụng các độ đo như Accuracy và F1-score để đánh giá hiệu suất của mô hình.

- *So sánh*: So sánh hiệu suất của mô hình AdaBoost với các phương pháp phân lớp khác như SVM, Logistic Regression, Random Forest, v.v.

Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Thuật toán AdaBoost và các biến thể.

Phạm vi nghiên cứu:

- Dữ liệu: Bộ dữ liệu Marketing and Sales, phân loại đậu khô, Yelp dataset, Yeast.
- Bài toán: Bài toán phân lớp nhị phân, đa lớp và đa nhãn.

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ HỌC MÁY VÀ HỌC KẾT HỢP

Chương này trình bày những kiến thức nền tảng về học máy, bài toán phân lớp, học kết hợp, và giới thiệu về thuật toán AdaBoost.

Phần đầu tiên giới thiệu về khái niệm học máy, các phương pháp học máy phổ biến, và đi sâu vào bài toán phân lớp. Phần này cũng trình bày các phương pháp phân lớp phổ biến, cũng như những vấn đề thường gặp trong phân lớp.

Phần tiếp theo trình bày về học kết hợp, một kỹ thuật học máy mạnh mẽ dựa trên việc kết hợp nhiều mô hình học yếu để tạo ra một mô hình mạnh hơn. Phần này giới thiệu tổng quan về học kết hợp, phân loại các phương pháp học kết hợp và trình bày các bước để xây dựng một hệ thống học kết hợp.

Cuối cùng, chương này giới thiệu về thuật toán AdaBoost, một thuật toán boosting được sử dụng rộng rãi để cải thiện hiệu suất của các mô hình học yếu.

1.1. Học máy và bài toán phân lớp

1.1.1. Học máy

Học máy (Machine Learning) là một công nghệ đang phát triển cho phép máy tính tự học hỏi từ dữ liệu lịch sử. Học máy sử dụng nhiều kỹ thuật khác nhau để tạo ra các mô hình toán học và đưa ra dự đoán dựa trên thông tin hoặc dữ liệu trước đó. Hiện nay, nó được sử dụng cho nhiều thứ khác nhau, bao gồm hệ thống đề xuất (Recommender systems), lọc email (email filtering), tự động gắn thẻ trên Facebook (Facebook auto-tagging), nhận dạng hình ảnh (image identification) và nhận dạng giọng nói (speech recognition) [1].

Học máy hiện đại khác biệt với học máy trong quá khứ nhờ vào các công nghệ tính toán mới. Các nhà nghiên cứu quan tâm đến trí tuệ nhân tạo muốn khám phá xem máy tính có thể học hỏi từ dữ liệu hay không; Ý tưởng rằng chúng có thể học hỏi mà không cần được lập trình để thực hiện các nhiệm vụ nhất định đã làm nảy sinh lĩnh vực này. Thành phần lặp lại của học máy là rất quan trọng bởi vì các mô hình có thể tự điều chỉnh độc lập khi tiếp xúc với dữ liệu mới. Để cung cấp các quyết định và kết quả đáng tin cậy, có thể lặp lại được, chúng học hỏi từ các tính toán trước đó. Mặc dù là

một ngành khoa học cũ, nhưng gần đây nó đã đạt được những tiến bộ mới [2]. Học máy được định nghĩa là một phân ngành của trí tuệ nhân tạo, tập trung vào việc tạo ra các thuật toán cho phép máy tính tự học hỏi từ dữ liệu và kinh nghiệm quá khứ. Thuật ngữ "học máy" được Arthur Samuel đặt ra vào năm 1959.

Các thuật toán học máy tạo ra một mô hình toán học với sự trợ giúp của dữ liệu mẫu đã được xử lý sẵn, hay còn gọi là "dữ liệu huấn luyện", hỗ trợ đưa ra dự đoán hoặc phán đoán mà không cần được lập trình rõ ràng. Học máy kết hợp với khoa học máy tính và thống kê để tạo ra các mô hình dự đoán [3]. Các mô hình này sẽ đạt hiệu quả càng cao khi chúng ta cung cấp càng nhiều thông tin. Bất cứ khi nào nhận được dữ liệu mới, hệ thống học máy sẽ dự đoán kết quả cho dữ liệu đó bằng cách học từ dữ liệu huấn luyện. Lượng dữ liệu được sử dụng để phát triển mô hình ảnh hưởng đến mức độ dự đoán đầu ra chính xác, vì lượng dữ liệu lớn hơn cho phép xây dựng mô hình chính xác hơn.

Thay vì viết mã code cụ thể cho một tình huống khó khăn cần đưa ra dự đoán, chúng ta chỉ cần cung cấp dữ liệu cho các thuật toán tổng hợp, sau đó chúng sẽ sử dụng dữ liệu để xây dựng logic và dự đoán kết quả. Học máy đã thay đổi cách chúng ta tiếp cận vấn đề ban đầu [4].

Theo thời gian, học máy ngày càng trở nên cần thiết. Bởi vì học máy có thể thực hiện các hoạt động quá phức tạp đối với con người để thực hiện trực tiếp. Vì chúng ta không thể truy cập vào lượng dữ liệu khổng lồ bằng tay một cách truyền thống, chúng ta cần các hệ thống máy tính và đây là nơi học máy phát huy tác dụng để đơn giản hóa cuộc sống của chúng ta. Bằng cách cung cấp cho các thuật toán học máy một lượng lớn dữ liệu, chúng ta có thể huấn luyện chúng để kiểm tra dữ liệu, xây dựng mô hình và tự động dự đoán đầu ra mong muốn. Hàm chi phí (cost function) có thể được sử dụng để ước tính hiệu suất của thuật toán học máy dựa trên lượng dữ liệu. Chúng ta có thể tiết kiệm thời gian và tiền bạc bằng cách sử dụng học máy. Các trường hợp sử dụng của nó giúp chúng ta dễ dàng hiểu được tầm quan trọng của học máy. Học máy hiện đang được sử dụng trong xe tự lái, phát hiện gian lận trên mạng, nhận dạng khuôn mặt và đề xuất bạn bè trên Facebook, cùng nhiều ứng dụng khác. Một số công ty hàng đầu, bao gồm Netflix và Amazon, đã phát triển các thuật toán học máy sử dụng một lượng lớn dữ liệu để phân tích sở thích của khách hàng và đưa ra đề xuất

sản phẩm. Học máy đã được chứng minh là hữu ích vì nó có thể giải quyết các vấn đề ở tốc độ và quy mô mà trí tuệ của con người không thể sánh được.

Học máy có thể được phân thành 3 phương pháp chính:

- Học có giám sát (Supervised learning)
- Học không giám sát (Unsupervised learning)
- Học tăng cường (Reinforcement in learning)

1.1.1.1 Học có giám sát

Học có giám sát (Supervised Learning) được định nghĩa là việc sử dụng các bộ dữ liệu huấn luyện đã được gán nhãn để thực hiện huấn luyện các mô hình phân lớp hay dự báo một cách chính xác. Khi dữ liệu đầu vào được đưa vào mô hình, mô hình sẽ điều chỉnh các trọng số của nó cho đến khi mô hình đạt được hiệu quả phù hợp. Một số phương pháp được sử dụng phổ biến trong học có giám sát bao gồm: mạng nơ-ron (neural networks), Naïve Bayes, hồi quy tuyến tính (linear regression), hồi quy logistic (logistic regression), random forest, support vector machine (SVM), …

1.1.1.2 Học không giám sát

Học không giám sát (Unsupervised Learning) là việc sử dụng các thuật toán học máy để phân tích và nhóm cụm các bộ dữ liệu không được gán nhãn (các cụm là các tập hợp con). Các thuật toán này khám phá dữ liệu và nhóm chúng lại mà không cần sự can thiệp của con người. Điều này có thể thực hiện bằng cách huấn luyện cho mô hình tìm ra những điểm giống và khác nhau của dữ liệu, qua đó tìm được mối liên hệ giữa chúng. Bởi vậy, các thuật toán học không giám sát thường được sử dụng cho các bài toán phân tích dữ liệu, phân khúc khách hàng, nhận dạng hình ảnh, … với các thuật toán như mạng nơ-ron, phân cụm K-means, …Ngoài ra, chúng cũng thường được sử dụng trong các bài toán "học biểu diễn" để làm giảm số lượng đặc trưng trong một mô hình thông qua quá trình giảm chiều (dimensionality reduction). Với tiêu biểu là các thuật toán như phân tích thành phần chính (principal components analysis), phân tích thành phần ngẫu nhiên (singular value decomposition).

Ngoài ra, còn có một số thuật toán liên quán đến học bán giám sát (Semi-supervised learning) đã được phát triển. Những thuật toán này được coi là trung gian giữa học có giám sát và học không giám sát. Trong quá trình huấn luyện, nó sử dụng

một bộ dữ liệu được gán nhãn nhỏ hơn để phân loại và trích xuất đặc trưng trên một bộ dữ liệu không được gán nhãn lớn hơn. Học bán giám sát có thể được sử dụng để giải quyết một số bài toán khi không đủ bộ dữ liệu được gán nhãn cho một thuật toán học có giám sát.

1.1.1.3 Học tăng cường

Học tăng cường (Reinforcement Learning) là một nhánh của học máy tập trung vào việc huấn luyện các tác nhân (agents) để đưa ra các quyết định tối ưu trong một môi trường nhất định. Thay vì được cung cấp dữ liệu được gán nhãn như trong học có giám sát, các tác nhân học tăng cường tương tác với môi trường, nhận phản hồi về các hành động của chúng thông qua các phần thưởng hoặc hình phạt. Dựa vào các phản hồi này, tác nhân điều chỉnh chiến lược của mình để tối đa hóa phần thưởng trong tương lai.

Học tăng cường thường được sử dụng trong các trường hợp mà dữ liệu được gán nhãn không khả dụng hoặc quá tốn chi phí để thu thập. Quá trình học tăng cường bao gồm 4 thành phần chính:

- Tác nhân (Agent): Đây là thực thể thực hiện các hành động trong môi trường.
- Môi trường (Environment): Là không gian mà tác nhân hoạt động và phản hồi lại các hành động của tác nhân.
- Hành động (Action): Là những hành động mà tác nhân có thể thực hiện trong môi trường.
- Phần thưởng (Reward): Là phản hồi từ môi trường đánh giá sự hiệu quả của các hành động.

Các ứng dụng phổ biến của học tăng cường bao gồm:

- Trò chơi: Các thuật toán học tăng cường đã đạt được thành công đáng kể trong việc tạo ra các tác nhân có khả năng chơi các trò chơi phức tạp như cờ vua, cờ vây và game điện tử.
- Điều khiển robot: Học tăng cường được sử dụng để huấn luyện robot thực hiện các nhiệm vụ phức tạp như đi bộ, điều hướng và thao tác vật thể.

- Tối ưu hóa: Học tăng cường có thể được sử dụng để tối ưu hóa các quy trình phức tạp như quản lý kho hàng, điều khiển mạng lưới giao thông và thiết kế hệ thống quảng cáo.

Một số thuật toán học tăng cường phổ biến bao gồm:

- Q-learning
- SARSA
- Deep Q-learning
- Policy Gradients

1.1.2. Bài toán phân lớp

1.1.2.1 Giới thiệu

Phân lớp, một kỹ thuật trọng tâm trong lĩnh vực học máy và khai thác dữ liệu, là quá trình phân loại dữ liệu vào các nhóm (loại, lớp) đã xác định trước. Nói cách khác, phân lớp là việc dự đoán nhãn lớp cho một trường hợp dữ liệu chưa biết dựa trên các trường hợp dữ liệu đã được phân loại trước đó. Phân lớp là một bài toán học có giám sát (supervised learning), nghĩa là chúng ta cần cung cấp cho thuật toán phân lớp một tập dữ liệu huấn luyện bao gồm các trường hợp dữ liệu đã được phân loại chính xác. Từ tập dữ liệu huấn luyện, thuật toán sẽ học các quy luật, mô hình để phân loại các trường hợp dữ liệu chưa biết.

Phân lớp đóng vai trò quan trọng trong nhiều ứng dụng thực tế, như:

- **Y tế:** Chẩn đoán bệnh, dự đoán kết quả điều trị, phân tích bệnh nhân, phát hiện bệnh sớm.
- **Tài chính:** Phân loại tín dụng, dự đoán khả năng vỡ nợ, phát hiện gian lận, phân tích thị trường chứng khoán.
- **Khoa học máy tính:** Phân loại hình ảnh, nhận dạng giọng nói, phát hiện spam, phân loại văn bản, xử lý ngôn ngữ tự nhiên.
- **Marketing:** Phân nhóm khách hàng, dự đoán hành vi mua hàng, phân tích thị trường, tối ưu hóa chiến lược quảng cáo.
- **An ninh:** Phát hiện tấn công mạng, phân tích tội phạm, giám sát an ninh.

1.1.2.2 Các phương pháp phân lớp phổ biến

Có rất nhiều phương pháp phân lớp được phát triển và ứng dụng trong thực tế. Dưới đây là một số phương pháp phổ biến:

- Mạng lưới Bayes (Bayesian Networks): Là một mô hình đồ thị xác suất biểu diễn mối quan hệ phụ thuộc giữa các biến. Mỗi nút trong mạng lưới đại diện cho một biến và các cạnh kết nối các nút biểu thị mối quan hệ phụ thuộc giữa các biến đó. Mạng lưới Bayes sử dụng kiến thức xác suất có điều kiện để dự đoán xác suất của một biến dựa trên giá trị của các biến khác. Ví dụ, khi dự đoán khả năng mắc bệnh ung thư phổi, mạng lưới Bayes sẽ xem xét các yếu tố như hút thuốc, tiền sử gia đình, tuổi tác,... và tính toán xác suất mắc bệnh dựa trên các yếu tố này.
    - Ưu điểm:
        - Khả năng xử lý dữ liệu thiếu: Mạng lưới Bayes có thể xử lý dữ liệu thiếu bằng cách tính toán xác suất dựa trên các biến có giá trị.
        - Linh hoạt trong việc áp dụng: Mạng lưới Bayes có thể được áp dụng cho cả các vấn đề hồi quy và phân lớp.
        - Dễ hiểu: Cấu trúc đồ thị của mạng lưới Bayes giúp chúng ta dễ dàng hiểu mối quan hệ giữa các biến.
    - Nhược điểm:
        - Khó khăn trong việc ước tính mật độ có điều kiện của các thuộc tính, đặc biệt là khi xử lý các thuộc tính liên tục.
        - Cần một lượng dữ liệu huấn luyện lớn để xây dựng một mạng lưới Bayes chính xác.
- Cây quyết định (Decision Trees): Là một kỹ thuật phân lớp dựa trên việc xây dựng một cấu trúc cây nhị phân. Mỗi nút trong cây đại diện cho một thuộc tính và mỗi cạnh đại diện cho một giá trị có thể có của thuộc tính đó. Lá cây đại diện cho lớp dự đoán. Cây quyết định đưa ra các quyết định dựa trên việc phân chia dữ liệu theo các thuộc tính một cách tuần tự. Ví dụ, khi phân loại khách hàng dựa trên khả năng mua hàng, cây quyết định có thể phân chia dữ liệu theo thu

nhập, độ tuổi, nghề nghiệp, v.v. và đưa ra dự đoán dựa trên kết quả phân chia này.

- Ưu điểm:
  - Dễ hiểu: Cấu trúc cây giúp chúng ta dễ dàng hiểu cách thức đưa ra quyết định.
  - Hiệu quả: Cây quyết định thường hoạt động hiệu quả với bộ dữ liệu lớn.
  - Khả năng xử lý dữ liệu hỗn hợp: Cây quyết định có thể xử lý cả dữ liệu rời rạc và liên tục.
- Nhược điểm:
  - Dễ bị ảnh hưởng bởi dữ liệu nhiễu: Cây quyết định có thể bị ảnh hưởng bởi các trường hợp dữ liệu ngoại lai.
  - Khó khăn trong việc xử lý dữ liệu có nhiều thuộc tính: Cây quyết định có thể trở nên phức tạp và khó hiểu khi số lượng thuộc tính tăng lên.

- K láng riềng gần nhất (K-Nearest Neighbor – KNN): Là một kỹ thuật phân lớp dựa trên khoảng cách giữa các điểm dữ liệu. Nó tìm k điểm gần nhất với điểm dữ liệu cần phân loại và dự đoán nhóm thành viên của điểm dữ liệu đó dựa trên nhóm thành viên của k điểm gần nhất. KNN dựa trên giả thiết rằng các điểm dữ liệu gần nhau thường thuộc cùng một nhóm. Ví dụ, khi phân loại hình ảnh, KNN sẽ so sánh điểm dữ liệu cần phân loại với các hình ảnh đã được phân loại trước đó, tìm k hình ảnh gần nhất và dự đoán lớp của hình ảnh đó dựa trên lớp của k hình ảnh gần nhất.
  - Ưu điểm:
    - Đơn giản: KNN là một kỹ thuật dễ hiểu và dễ triển khai.
    - Hiệu quả với bộ dữ liệu lớn: KNN hoạt động hiệu quả với bộ dữ liệu lớn.
    - Mạnh mẽ với dữ liệu nhiễu: KNN tương đối ít bị ảnh hưởng bởi dữ liệu nhiễu.

- Nhược điểm:
  - Yêu cầu bộ nhớ lớn: KNN cần lưu trữ toàn bộ tập dữ liệu huấn luyện.
  - Thời gian tính toán chậm: KNN có thể tốn thời gian tính toán khi bộ dữ liệu lớn.
  - Bị ảnh hưởng bởi thuộc tính không phù hợp: KNN có thể bị ảnh hưởng bởi các thuộc tính không phù hợp.

- Máy vector hỗ trợ (Support Vector Machines – SVMs): Là một phương pháp học máy dựa trên lý thuyết học thống kê được đề xuất bởi Vapnik. Nó là một trong những kỹ thuật phổ biến nhất trong việc giải quyết các vấn đề liên quan đến phân lớp dữ liệu. SVM sử dụng các vectơ hỗ trợ để tìm siêu phẳng có khoảng cách tối ưu giữa các điểm dữ liệu thuộc các lớp khác nhau. SVM tìm kiếm một siêu phẳng tối ưu để tách biệt các lớp dữ liệu với khoảng cách lớn nhất có thể. Ví dụ, khi phân loại email thành spam và không spam, SVM sẽ tìm kiếm một đường phân cách tối ưu giữa các email thuộc hai lớp này.
  - Ưu điểm:
    - Khả năng xử lý các vấn đề phân lớp đa chiều và không tuyến tính: SVM có thể xử lý các vấn đề phân lớp phức tạp.
    - Độ chính xác cao: SVM thường đạt được độ chính xác cao trong phân lớp.
  - Nhược điểm:
    - Cần lựa chọn chính xác các thông số: Việc lựa chọn các thông số chính xác cho SVM là rất quan trọng để đạt được kết quả tối ưu.
    - Tốn thời gian huấn luyện: SVM có thể tốn nhiều thời gian để huấn luyện, đặc biệt là với bộ dữ liệu lớn.
    - Ứng dụng:
    - Nhận dạng khuôn mặt
    - Phân loại văn bản

- Phân tích hình ảnh y tế

1.1.2.3 Các vấn đề trong phân lớp

Phân lớp là một bài toán phức tạp và có nhiều vấn đề cần giải quyết. Dưới đây là một số vấn đề thường gặp trong phân lớp:

- Xử lý dữ liệu thiếu: Dữ liệu thiếu là một vấn đề thường gặp trong các tập dữ liệu thực tế. Dữ liệu thiếu có thể gây ra vấn đề trong cả giai đoạn huấn luyện và phân lớp.
    - Các phương pháp xử lý dữ liệu thiếu:
        - Loại bỏ các trường hợp dữ liệu thiếu: Phương pháp đơn giản nhất là loại bỏ các trường hợp dữ liệu thiếu. Tuy nhiên, phương pháp này có thể dẫn đến việc mất dữ liệu.
        - Thay thế dữ liệu thiếu bằng giá trị trung bình, trung vị, hoặc giá trị phổ biến nhất: Phương pháp này đơn giản nhưng có thể ảnh hưởng đến độ chính xác của phân lớp.
        - Sử dụng các thuật toán xử lý dữ liệu thiếu: Các thuật toán xử lý dữ liệu thiếu có thể dự đoán các giá trị thiếu dựa trên các trường hợp dữ liệu khác.
- Dữ liệu nhiễu: Dữ liệu nhiễu là các giá trị sai lệch, không chính xác trong tập dữ liệu. Dữ liệu nhiễu có thể ảnh hưởng đến độ chính xác của phân lớp.
    - Các phương pháp xử lý dữ liệu nhiễu:
        - Loại bỏ dữ liệu nhiễu: Phương pháp này có thể giúp loại bỏ các giá trị nhiễu rõ ràng.
        - Sử dụng các thuật toán xử lý dữ liệu nhiễu: Các thuật toán xử lý dữ liệu nhiễu có thể phát hiện và loại bỏ các giá trị nhiễu.
- Thuộc tính không phù hợp: Là các thuộc tính không liên quan đến lớp dự đoán. Các thuộc tính này có thể làm giảm hiệu suất của phân lớp.
    - Các phương pháp xử lý thuộc tính không phù hợp:

        - Lựa chọn thuộc tính: Chọn các thuộc tính liên quan đến lớp dự đoán.
        - Khử chiều: Giảm số lượng thuộc tính bằng cách kết hợp các thuộc tính liên quan.

- Bất cân bằng lớp: Là hiện tượng một số lớp có nhiều trường hợp dữ liệu hơn các lớp khác. Bất cân bằng lớp có thể làm cho thuật toán phân lớp thiên vị về các lớp có nhiều trường hợp dữ liệu hơn.
    - Các phương pháp xử lý bất cân bằng lớp:
        - Oversampling: Tăng số lượng trường hợp dữ liệu trong các lớp có ít trường hợp dữ liệu hơn.
        - Undersampling: Giảm số lượng trường hợp dữ liệu trong các lớp có nhiều trường hợp dữ liệu hơn.
        - Sử dụng các thuật toán xử lý bất cân bằng lớp: Các thuật toán xử lý bất cân bằng lớp có thể cân bằng trọng số cho các lớp để cải thiện độ chính xác của phân lớp.

Phân lớp là một bài toán quan trọng trong học máy với nhiều ứng dụng thực tế. Tuy nhiên, phân lớp cũng là một bài toán phức tạp và có nhiều vấn đề cần giải quyết. Việc lựa chọn kỹ thuật phân lớp phù hợp và xử lý các vấn đề liên quan đến dữ liệu là rất quan trọng để đạt được hiệu suất phân lớp tối ưu.

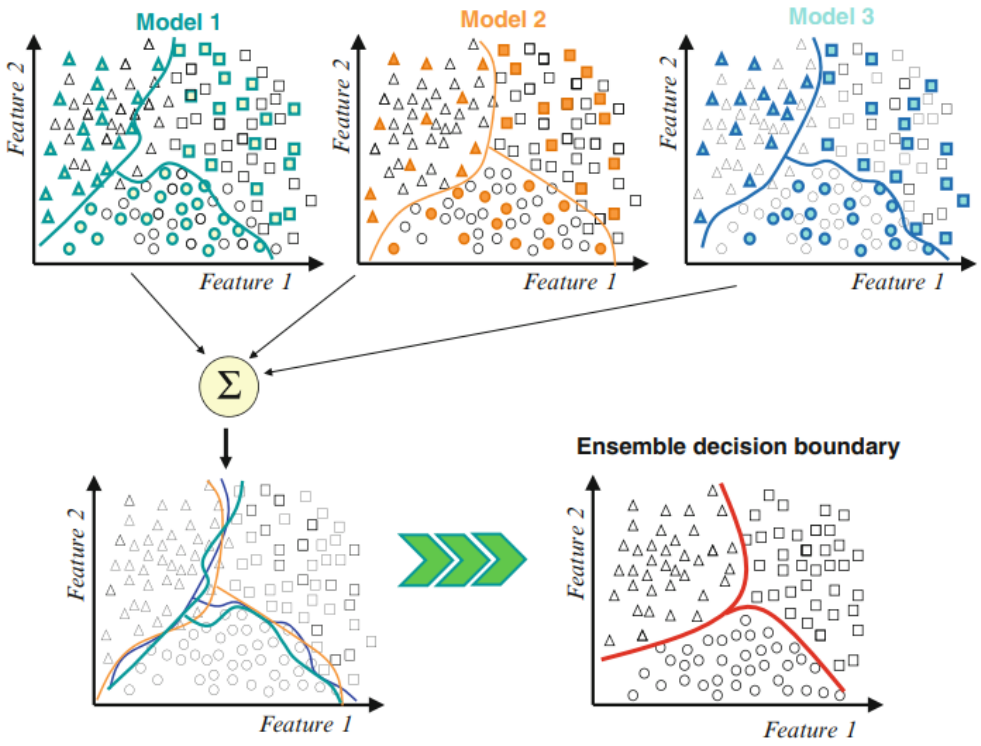
1.2. Học kết hợp

1.2.1. Tổng quan

Học kết hợp - Ensemble Learning đề cập đến các quy trình được sử dụng để huấn luyện nhiều mô hình học máy và kết hợp đầu ra của các mô hình đó, coi những mô hình thuộc một nhóm gồm những mô hình đưa ra quyết định. Với nguyên tắc là quyết định của nhóm phải có trung bình độ chính xác cao hơn so với các mô hình riêng lẻ nếu được kết hợp theo một cách thích hợp. Ý tưởng về việc kết hợp, thu thập những ý kiến, quyết định cho một vấn đề xuất hiện rất nhiều ở cuộc sống hằng ngày, ví dụ như: Bản chất của nền dân chủ nơi một nhóm người bỏ phiếu để đưa ra quyết định, chọn một quan chức được bầu hay quyết định một luật mới, trên thực tế là dựa trên

việc ra quyết định dựa trên tập thể, v.v. Hơn thế, nhiều nghiên cứu thực nghiệm và lý thuyết đã chứng minh rằng các mô hình tập hợp thường đạt được độ chính xác cao hơn các mô hình đơn lẻ. [5]

Lý do cho khiến cho ensemble learning có mang tính khả thi nằm ở hai khái niệm quen thuộc trong học máy. Cụ thể hơn, có thể khẳng định rằng bất kỳ dự đoán của mô hình nào cũng bao gồm hai thành phần là: Sai lệch (bias), độ chuẩn xác của bộ phân lớp (classifier); và phương sai (variance), độ chính xác của bộ phân lớp khi được huấn luyện trên các tập huấn luyện khác nhau. Thông thường, hai thành phần này có mối quan hệ đánh đổi: Các bộ phân lớp có độ lệch thấp có xu hướng có phương sai cao và ngược lại. Mặt khác, phép tính trung bình có tác dụng làm mịn (giảm phương sai). Do đó, mục tiêu của các hệ thống ensemble là tạo ra một số bộ phân lớp có độ lệch tương đối cố định (hoặc tương tự) và sau đó kết hợp các đầu ra của chúng, chẳng hạn như bằng cách lấy trung bình, để giảm phương sai. [6]



Hình 1.1 Ví dụ minh họa cho thuật toán Ensemble

Như vậy, mục tiêu của ensemble learning là tăng độ chính xác bằng cách huấn luyện, kết hợp nhiều mô hình.

1.2.2. Phân loại

Từ những nghiên cứu thành công trước đó bắt đầu từ năm 1979, những nghiên cứu về thuật toán ensemble đã nở rộ với nhiều cái tên như: Bagging, Random Forest, Mixture of Experts, v.v. Tất cả những thuật toán trên và những hệ thống ensemble nói chung đều chỉ khác nhau dựa trên việc lựa chọn dữ liệu huấn luyện cho các bộ phân lớp riêng lẻ, quy trình cụ thể được sử dụng để tạo các bộ phân lớp con của hệ thống ensemble và/hoặc quy tắc kết hợp để có được dự đoán cuối cùng. Hơn thế nữa, đây là ba khía cạnh chính của bất kỳ hệ thống ensemble nào. [5]

Trong hầu hết các trường hợp, các thuật toán thuộc ensemble có thể được phân vào hai loại: *Lựa chọn bộ phân lớp* (Classifier selection) và *hợp nhất bộ phân lớp* (Classifier fusion). Trong đó, việc lựa chọn bộ phân lớp nghĩa là mỗi bộ phân lớp được huấn luyện để có kế quả tốt nhất với một phần của toàn bộ không gian đặc trưng (local expert). Khi có một mẫu dữ liệu mới, bộ phân lớp mà có dữ liệu huấn luyện có sự tương đồng lớn nhất với mẫu mới theo một thang đo cụ thể sẽ được lựa chọn để đưa dự đoán, hoặc sẽ được đánh trọng số lớn nhất trong quá trình dự đoán. Đối với hợp nhất bộ phân lớp, tất cả bộ phân lớp sẽ được huấn luyện trên toàn bộ không gian đặc trưng, sau đó được kết hợp thành một bộ phân lớp có phương sai thấp hơn (như vậy lỗi sẽ thấp hơn). Bagging, random forets, arc-x4 và boosting/Adaboost là những thuật toán phổ biến đại diện cho loại hợp nhất bộ phân lớp. [5]

1.2.3. Xây dựng một hệ thống học kết hợp

Như đã trình bày ở trên, ba khía cạnh chính trong một hệ thống ensemble là: (1) *lựa chọn dữ liệu*; (2) *huấn luyện bộ phân lớp con*; và (3) *kết hợp các bộ phân lớp*.

Chọn dữ liệu

Việc các bộ phân lớp con cho ra các kết quả lỗi khác nhau trên bất kỳ mẫu nào là điều hết sức quan trọng trong các hệ thống ensemble. Xét cho cùng, nếu tất cả các bộ phân lớp con đều cho ra một kết quả thì khi kết hợp lại sẽ không mang lại lợi ích gì. Vì vậy, sự đa dạng trong kết quả đầu ra của các bộ phân lớp con là một điều quan trọng, đặc biệt là khi các bộ phân lớp con cho ra kết quả lỗi – khi đó phương sai của kết quả sau cùng sẽ giảm như đã trình bày ở trên. Lý tưởng nhất là đầu ra của các bộ phân lớp con phải độc lập hoặc tốt nhất là có mối tương quan nghịch. [5]

Có một vài chiến lược để đạt được sự đa dạng trong kết quả đầu ra như đã trình bày ở trên, mặc dù vậy thì sử dụng những bộ dữ liệu con trong bộ dữ liệu huấn luyện ban đầu là cách tiếp cận phổ biến nhất, được minh họa ở Hình 1.1 trong khi đó lấy dữ liệu từ một phân phối mà ưu tiên các mẫu bị phân lớp sai là cốt lõi của thuật toán boosting. Bên cạnh đó, còn nhiều những chiến lược khác như sử dụng tập con của các đặc trưng để huấn luyện bộ phân lớp. [5]

Huấn luyện các bộ phân lớp con

Cốt lõi của bất kỳ hệ thống ensemble là chiến lược được sử dụng để tạo ra từng các bộ phân lớp con. Nhiều thuật toán cạnh tranh đã được phát triển để huấn luyện các bộ phân lớp con; tuy nhiên, bagging, arc-x4, boosting (và các biến thể), v.v. [5]

Kết hợp các bộ phân lớp con

Bước cuối cùng trong bất kỳ hệ thống ensemble nào là cơ chế được sử dụng để kết hợp các bộ phân lớp riêng lẻ. Chiến lược được sử dụng trong bước này phụ thuộc một phần vào loại bộ phân lớp được sử dụng. Ví dụ như Support Vector Machine chỉ có đầu ra nhãn có giá trị rời rạc. Các quy tắc kết hợp được sử dụng phổ biến nhất cho các phân loại như vậy là bỏ phiếu đa số (đơn giản hoặc có trọng số), theo sau là số đếm Borda. Các bộ phân lớp khác, chẳng hạn như Multilayer Perceptron hoặc Naïve Bayes, có đầu ra là giá trị liên tục riêng cho từng lớp, được hiểu là độ hỗ trợ mà bộ phân lớp cho ra ứng với mỗi lớp. Ứng với mỗi kiểu dữ liệu đầu ra được sử dụng, có những lựa chọn phù để kết hợp đầu ra của mô hình. [5]

- Ứng với đầu ra là nhãn là các lớp có các cách kết hợp: Bỏ phiếu đa số, bỏ phiếu đa số có trọng số, đếm Borda.
- Ứng với đầu ra là giá trị liên tục có các cách kết hợp: Ma trận quyết định Kuncheva, quy tắc trung bình, trung bình có trọng số, v.v.

CHƯƠNG 2. CÁC THUẬT TOÁN ADABOOST

2.1. Thuật toán Boosting

Boosting là một kỹ thuật học máy mạnh để cải thiện hiệu suất của những mô hình học yếu (weak learner). Kỹ thuật này dựa trên cơ sở là "**giả định học yếu**" - weak learning assumption. [6]

Giả định học yếu là một thành phần quan trọng trong thuật toán boosting. Giả định này đưa ra những yêu cầu tối thiểu về khả năng của thuật toán học cơ sở (base learner) - ở đây là những thuật toán học yếu, để boosting có thể hoạt động hiệu quả. Giả định này cho rằng thuật toán học cơ sở phải "tốt hơn một chút" so với việc đoán ngẫu nhiên. Nói cách khác, tỷ lệ lỗi của mô hình học yếu phải thấp hơn 50%. [6]

Lý do là vì boosting hoạt động bằng cách liên tục tạo ra các bộ phân loại mới dựa trên các bộ phân loại trước đó. Vì vậy, nếu bộ phân loại cơ sở quá yếu (tỷ lệ lỗi gần 50%), việc tạo ra các bộ phân loại mới sẽ không mang lại nhiều cải thiện.Do đó, để boosting hoạt động hiệu quả, cần có một thuật toán học cơ sở có khả năng học tốt hơn so với việc đoán ngẫu nhiên. Một thuật toán học cơ sở càng chính xác thì boosting càng hiệu quả. Ngoài ra, giả định học yếu chỉ là điều kiện cần chứ không phải là điều kiện đủ để boosting hoạt động hiệu quả. Hiệu suất của boosting còn phụ thuộc vào nhiều yếu tố khác, chẳng hạn như thuật toán boosting được sử dụng và tập dữ liệu được sử dụng để đào tạo. [6]

Giống như hầu hết các thuật toán học có giám sát khác, boosting sử dụng đầu vào là một tập dữ liệu huấn luyện có dạng $(x_1,\ y_1), \dots, (x_m,\ y_m)$. Trong đó mỗi x_i là một thể hiện của $\mathcal{X}$ và y_i là nhãn của lớp. Phương tiện duy nhất của boosting để học từ dữ liệu là việc thông qua các lần gọi tới thuật toán học cơ sở (có thể coi như các "black box"). Vì vậy, nếu các thuật toán học cơ sở chỉ luôn được gọi lặp lại với cùng một bộ dữ liệu đào tạo thì không thể mang lại hiệu quả tốt cho thuật toán boosting; thay vào đó, chúng ta phải **thao tác với dữ liệu** cung cấp cho thuật toán học cơ sở. [6]

Thật vậy, ý tưởng then chốt đằng sau boosting là việc chọn các tập dữ liệu cho các bộ học cơ sở sao cho mô hình phải học được một cái gì đó "mới" sau mỗi lần học, buộc các bộ học cơ sở phải tạo ra các phân loại khác biệt so với những lần gọi trước đó.

2.2. Adaptive Boosting

Adaptive Boosting – AdaBoost: Là một thuật toán boosting được sử dụng rộng rãi để cải thiện hiệu suất của các bộ học yếu. Thuật toán này hoạt động bằng cách lặp đi lặp lại việc đào tạo một bộ học yếu trên các tập dữ liệu được chọn cẩn thận và kết hợp các dự đoán của chúng theo một cách có trọng số. [6]

Để chọn các tập dữ đào tạo trong mỗi lần gọi lại, AdaBoost duy trì một phân phối trên các dữ liệu đào tạo. Phân phối được sử dụng trên lần lặp thứ t được ký hiệu là D_t và trọng số được gán cho mẫu dữ liệu huấn luyện thứ i được ký hiệu là $D_t(i)$. Trọng số này là thước đo tầm quan trọng của việc phân loại chính xác mẫu i trên lần lặp hiện tại. Ban đầu, trọng số của tất cả các mẫu huấn luyện sẽ được đặt bằng nhau, nhưng trên mỗi lần lặp huấn luyện, trọng số của các mẫu được phân loại sai sẽ được tăng lên đáng kể, để làm cho các bộ học cơ sở "chú ý" nhiều hơn đến những mẫu sai này. [6]

2.2.1. Kiểm nghiệm các bộ phân loại (Scouting)

Quá trình tìm kiếm được thực hiện bằng cách thử nghiệm (test) từng bộ phân loại trên tập dữ liệu huấn luyện $\mathcal{X}$. Đối với mỗi bộ phân loại, chúng ta sẽ thử nghiệm và đánh giá từng bộ phân loại bằng cách xử dụng một hàm mất mát, ở đây là hàm mất mát mũ (exponential loss). [6]

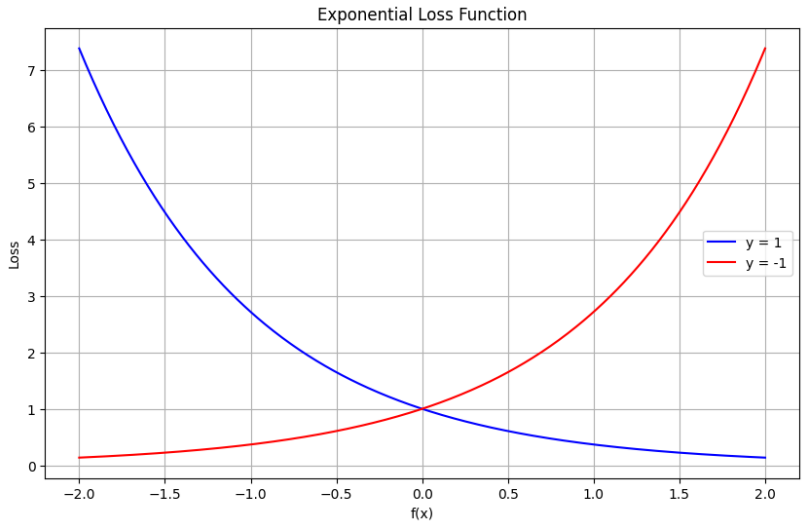
$$L\big(y, f(x)\big) = e^{-y\, f(x)}$$

Trong đó:

- y là nhãn thực tế của điểm dữ liệu x.
- $f(x)$ là giá trị dự báo của bộ phân loại cho điểm dữ liệu x.
- $L\big(y, f(x)\big)$ là giá trị của hàm mất mát mũ tương ứng với cặp dữ liệu (x, y).

Theo đó, mỗi khi mô hình phân loại thất bại đối với mẫu $x_i \in \mathcal{X}$, chi phí cho mẫu hiện tại sẽ được tính bằng e^{β}. Ngược lại, khi mô hình phân loại đúng, chi phí cho mẫu này sẽ được tính bằng $e^{-\beta}$. Bằng cách chọn một hệ số $\beta > 0$, chúng ta đảm bảo rằng mức độ phạt cho các mẫu được phân lớp sai sẽ lớn hơn mức độ phạt cho các mẫu được phân lớp đúng ($e^{\beta} > e^{-\beta}$). Lưu ý là các mẫu được phân loại đúng vẫn bị phạt

nhẹ, mặc dù mức phạt này nhỏ hơn nhiều so với những mẫu phân loại sai, lý do là để tránh hiện tượng overfitting, giữ cho mô hình nhạy cảm với dữ liệu mới.



Hình 2.1 Đồ thị hàm mất mát mũ

Dựa vào đồ thị của hàm mất mát mũ như trên, ta có thể thấy: Khi $y\,f(x) < 0$, giá trị của hàm mất mát mũ sẽ càng lớn khi $y\,f(x)$ càng nhỏ, tức là mẫu càng bị dự báo sai nhiều thì sẽ càng bị phạt nặng. Ngược lại, khi $y\,f(x) > 0$, giá trị của hàm mất mát mũ sẽ càng nhỏ khi $y\,f(x)$ càng lớn, mẫu được phân lớp đúng thì sẽ bị phạt nhẹ hơn.

Ý tưởng chính của AdaBoost là sẽ chọn ra một bộ phân loại chứa nhiều thông tin nhất (lỗi ít nhất) có thể trong mỗi lần lặp thứ t của T lần lặp. Từ đó gán trọng số cho các mẫu trong tập huấn luyện. Ban đầu, tất cả các mẫu được gán cùng một trọng số (thường là 1 hoặc 1/N). Sau đó, trong mỗi lần lặp, các mẫu bị phân loại sai sẽ có trọng số lớn hơn, điều này làm các bộ phân loại sau đó sẽ "tập trung" hơn vào những mẫu này. Điều này không có nghĩa là chỉ những bộ phân loại sau đó mới có giá trị, tất cả những bộ phân loại đều đem lại những thông tin khác nhau (dù ít hay nhiều) và những bộ phân loại ở lần lặp sau chỉ cung cấp một cái nhìn mới cho mô hình. Vì vậy, tất cả các bộ phân loại được kết hợp với nhau một cách tối ưu. [6]

2.2.2. Lựa chọn bộ phân loại thích hợp (Drafting)

Tại mỗi lần lặp thứ t, ta cần lựa chọn ra một mô hình tốt nhất dựa trên kết quả đánh giá từ bước trước.

Trước hết, ta đã có t-1 mô hình được lựa chọn từ các lần lặp trước đó, mô hình phân loại kết hợp bây giờ sẽ có dạng:

$$H_{(t-1)}(x_i) = \alpha_1 h_1(x_i) + \alpha_2 h_2(x_i) + \cdots + \alpha_{t-1} h_{t-1}(x_i)$$

Trong đó:

- α_j là trọng số của bộ phân loại thứ j.
- $h_j(x_i)$ là kết quả phân lớp của bộ phân lớp thứ j cho mẫu thứ i.
- $H_{(t-1)}(x_i)$ là kết quả phân lớp khi kết hợp $t-1$ bộ phân loại trước đó với nhau.

Lưu ý là tại lần lặp đầu tiên ($t = 1$), $H_{(t-1)}$ có giá trị bằng không. Việc cần làm của chúng ta bây giờ là mở rộng bộ phân loại trước đó thành:

$$H_{(t)}(x_i) = H_{(t-1)}(x_i) + \alpha_t h_t(x_i)$$

Hàm mất mát cho mô hình bên trên theo hàm mất mát mũ đã được mô tả ở phần trước sẽ có dạng:

$$E = \sum_{i=1}^{N} e^{-y_i\left(H_{(t-1)}(x_i) + \alpha_t h_t(x_i)\right)}$$

Vì $H_{(t-1)}$ đã được chọn là mô hình tối ưu trong các lần lặp trước đó, chúng ta chỉ cần tìm α_t và h_t sao cho mô hình hiện tại tối ưu (có lỗi nhỏ nhất có thể), nên công thức lỗi bên trên có thể được viết lại dưới dạng:

$$E = \sum_{i=1}^{N} D_i(t) e^{-y_i \alpha_t h_t(x_i)} \quad (1)$$

Trong đó:

$$D_i(t) = e^{-y_i H_{(t-1)}(x_i)}$$

Với $i = 1, \ldots, m$. Trong lần lặp đầu $D_i(1) = \frac{1}{m}$ với mọi $i = 1, \ldots, m$. Trong các lần lặp tiếp theo, giá trị của $D_i(t)$ sẽ là trọng số cho mỗi điểm dữ liệu trong tập huấn luyện tại lần lặp thứ t. Ta có thể viết lại công thức (1) thành hai tổng như sau:

$$E = \sum_{i:h_t(x_i)=y_i} D_i(t) e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_i(t) e^{\alpha_t} \quad (2)$$

Ta có thể thấy, chi phí cho mô hình kết hợp thứ t bằng tổng trọng số của những điểm dữ liệu dự đoán đúng cộng với tổng trọng số của những điểm dự đoán sai.

Vì $\alpha_t > 0$, việc cực tiểu hóa E cũng sẽ tương đương với cực tiểu hóa $e^{\alpha_t}E$. Vì vậy, ta có:

$$e^{\alpha_t}E = e^{\alpha_t}\sum_{i:h_t(x_i)=y_i} D_i(t)e^{-\alpha_t} + e^{\alpha_t}\sum_{i:h_t(x_i)\neq y_i} D_i(t)e^{\alpha_t}$$

$$= \sum_{i:h_t(x_i)=y_i} D_i(t) + \sum_{i:h_t(x_i)\neq y_i} D_i(t)e^{2\alpha_t}$$

Với $e^{2\alpha_t} > 1$, công thức trên có thể được viết lại bằng:

$$e^{\alpha_t}E = \sum D_i(t) + \sum_{i:h_t(x_i)\neq y_i} D_i(t)(e^{2\alpha_t} - 1)$$

Trong đó $\sum D_i(t)$ là tổng trọng số của các điểm dữ liệu trong tập huấn luyện, và là một hằng số tại lần lặp hiện tại. Vì vậy, để cực tiểu hóa vế phải, ta phải cực tiểu hóa $\sum_{i:h_t(x_i)\neq y_i} D_i(t)$. Điều này đồng nghĩa với việc mô hình được chọn phải có tổng trọng số cho các mẫu dự đoán sai ít nhất. Nói cách khác, mô hình được chọn cho lần lặp hiện tại phải là mô hình bị phạt ít nhất dựa trên trọng số của tập huấn luyện hiện tại.

2.2.3. Tính toán trọng số (Weighting)

Sau khi đã chọn được mô hình tốt nhất trong lần lặp t hiện tại, ta cần tính toán trọng số α_t cho từng bộ phân loại. Ta có thể tìm α_t bằng cách tìm cực trị (cực tiểu) của công thức (2), ta có đạo hàm của E là:

$$\frac{dE}{d\alpha_t} = -\sum_{i:h_t(x_i)=y_i} D_i(t)e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_i(t)e^{\alpha_t}$$

Giải phương trình đạo hàm bằng không và nhân cả hai vế với e^{α_t}, ta được:

$$-\sum_{i:h_t(x_i)=y_i} D_i(t) + \sum_{i:h_t(x_i)\neq y_i} D_i(t)e^{2\alpha_t} = 0$$

Hay:

$$\alpha_t = \frac{1}{2}log\left(\frac{\sum_{i:h_t(x_i)=y_i} D_i(t)}{\sum_{i:h_t(x_i)\neq y_i} D_i(t)}\right)$$

Đặt $\epsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i] = \frac{\sum_{i:h_t(x_i)\neq y_i} D_i(t)}{\sum D_i(t)}$ là tỷ lệ lỗi dựa trên trọng số của các điểm dữ liệu, ta có:

$$\alpha_t = \frac{1}{2} log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

2.2.4. Thuật toán AdaBoost

Thuật toán AdaBoost sẽ được trình bày như sau [6]:

- Cho tập dữ liệu huấn luyện $(x_1, y_1), \ldots, (x_m, y_m)$. Trong đó $x_i \in X$ và y_i là nhãn của lớp.
- Khởi tạo trọng số cho tất cả các mẫu huấn luyện với giá trị bằng nhau (Freund, Schapire, 1997):

$$D_1(i) = \frac{1}{m} \qquad \text{với i = 1,…, m.}$$

- Lặp lại quá trình sau T lần, với t = 1, …, T:
  - (1) Huấn luyện mô hình yếu với phân phối D_t
  - (2) Trích xuất nhãn dự đoán h_t
  - (3) Chọn mô hình yếu h_t có tỷ lệ lỗi nhỏ nhất, với:

$$\epsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$$

  - (4) Tính toán trọng số cho mô hình yếu thứ t:

$$\alpha_t = \frac{1}{2}\log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

  - (5) Cập nhật trọng số cho tất cả mẫu dữ liệu huấn luyện theo mô hình yếu được chọn:

$$D_{t+1}(i) = \frac{D_t(i)}{E_t} \times \begin{cases} e^{-\alpha_t} & \text{nếu } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{nếu } h_t(x_i) \neq y_i \end{cases}$$

Trong đó, việc chia cho E_t có tác dụng để đảm bảo rằng phân phối D_{t+1} là một phân phối hợp lệ (có tổng bằng 1).

$$E_t = \sum_{i=1}^{N} D_t(i) e^{-y_i h_t(i) \alpha_t}$$

- Sau khi đã có được T mô hình yếu, đầu ra sẽ được tính bằng cách:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

AdaBoost có thể được sử dụng để giải quyết các bài toán phân loại đa lớp. Với điều kiện là các mô hình học yếu có độ chính xác không dưới 50%, nếu không mô hình có thể không hoạt động hiệu quả. Vì vậy, những bài toán phân loại đa lớp thường được đơn giản hóa bằng việc giải quyết nhiều bài toán phân loại hai lớp.

2.3. AdaBoost.M1

2.3.1. Giới thiệu

Freund và Schapire đã đề xuất thuật toán AdaBoost.M1, đây là một dạng tổng quát hóa đơn giản của AdaBoost cho 2 nhóm cho bài toán đa lớp bằng cách sử dụng các bộ phân lớp đa lớp. Một trong những ý tưởng chính của thuật toán là duy trì phân phối D của các trọng số trên tập huấn luyện L = $\{(x_1, y_1),...,(x_m, y_m); x_i \in X, y_i \in Y\}$. Trọng số của phân phối này trên mẫu thứ i trong vòng lặp t được kí hiệu là $D_t(i)$. Ở mỗi vòng, trọng số của các mẫu được phân loại không chính xác sẽ tăng lên khiến cho mô hình học yếu buộc phải tập trung vào các mẫu "khó" trong quá trình huấn luyện. Mục tiêu của mô hình học yếu là tìm ra giả thuyết h_t phù hợp cho phân bố D_t. [6]

2.3.2. Thuật toán AdaBoost.M1

Thuật toán AdaBoost.M1 sẽ được thực hiện gần tương tự như AdaBoost đã trình bày ở phần trước [6]:

- Đầu vào: Tập huấn luyện L = $\{(x_1, y_1),...,(x_m, y_m); x_i \in X, y_i \in Y\}$, $Y = \{1,...,|Y|\}$, bộ phân lớp $h : X \rightarrow Y$.

  T: Số lần lặp của thuật toán

- Khởi tạo: $D_1(i) = \frac{1}{m}$
- Lặp $t = 1,\ldots,T$:

  (1) Huấn luyện bộ phân lớp yếu h_t với phân phối D_t mà h_t có thể tối thiểu lỗi:

  $$\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq y_i)$$

  (2) Nếu $\epsilon_t \geq 1/2$: thoát khỏi vòng lặp với giá trị $T := t - 1$.

(3) Đặt $\alpha_t = \ln(\frac{1-\epsilon_t}{\epsilon})$

(4) Cập nhật D:

$$D_{t+1}(i) = D_t(i)e^{-\alpha_t I(h_t(x_i)=y_i)}/Z_t$$

Với Z_t là hệ số chuẩn hóa (để có D_{t+1} là một phân phối)

- Kết quả đầu ra: Bộ phân lớp $H(x)$ có dạng:

$$H(x) = \arg\max_{y \in Y} f(x,y) = \arg\max_{y \in Y} \left(\sum_{t=1}^{T} \alpha_t I(h_t(x) = y) \right)$$

2.4. AdaBoost.MH

2.4.1. Bài toán phân loại đa lớp đa nhãn

Với Y là một tập hợp các nhãn (hữu hạn) gồm K nhãn. Với những bài toán phân loại đơn lớp, mỗi mẫu $x \in X$ được gán một nhãn duy nhất là $y \in Y$ để tạo thành một cặp dữ liệu nhãn (x, y), mục tiêu thường là tìm giả thuyết $H: X \to Y$ sao cho cực tiểu hóa xác suất $y \neq H(x)$ trên một mẫu mới (x, y). Trong khi đó, với những bài toán phân loại đa lớp (multi-label), mỗi mẫu $x \in X$ có thể có nhiều nhãn thuộc Y. Vì vậy, nhãn của dữ liệu bây giờ sẽ là (x, g) với $g \subseteq Y$ là một tập hợp các nhãn được gán cho x (phân loại đơn lớp chỉ là trường hợp đặc biệt của phân loại đa lớp với $|g| = 1$ cho tất cả các mẫu) [6]

2.4.2. Hàm mất mát Hamming

Hàm mất mát Hamming hay Hamming loss là một phép đo được sử dụng trong các bài toán phân loại đa nhãn (multi-label classification), nơi mỗi mẫu dữ liệu có thể thuộc về nhiều lớp (nhãn) khác nhau cùng một lúc.

Hamming loss tính toán tỷ lệ các nhãn được dự đoán sai so với tổng số lượng nhãn trên tất cả các mẫu dữ liệu. Cụ thể, để tính Hamming loss, ta thực hiện các bước sau: (i) So sánh các nhãn dự đoán với các nhãn thực tế trên từng mẫu dữ liệu. (ii) Đếm số lượng nhãn được dự đoán sai. (iii) Chia số lượng nhãn dự đoán sai cho tổng số lượng nhãn trên tất cả các mẫu dữ liệu. Kết quả cuối cùng là tỷ lệ trung bình của các nhãn dự đoán sai trên tất cả các mẫu dữ liệu.

Công thức tính Hamming Loss của H sẽ có dạng:

$$hloss_D(H) = \frac{1}{K} E_{(x,g)\sim D}[|H(x) \triangle Y|]$$

Trong đó, $H(x) \triangle Y$ là ký hiệu của phép toán "khác biệt đối xứng" (symmetric diffrence) giữa hai tập hợp $H(x)$ và Y, lấy ra những phần tử chỉ thuộc 1 trong hai tập hợp trên. Phép chia cho K để đảm bảo giá trị nằm trong khoảng [0, 1].

Để giảm thiểu Hamming loss, ta có thể chia nhiệm vụ phân lớp thành K bài toán phân lớp nhị phân theo cách tiếp cận "một so với tất cả" (one-against-all). Nghĩa là, ta có thể xem g như là K nhãn nhị phân, mỗi nhãn thể hiện rằng nhãn y có hay không có trong g. Tương tự, $H(x)$ có thể được xem như K dự đoán nhị phân. Hamming loss sau đó có thể được tính bằng trung bình tỷ lệ lỗi của H trên K bài toán nhị phân này.

Với $g \subseteq Y$, chúng ta định nghĩa $g[l]$ với $l \in Y$:

$$g[l] = \begin{cases} +1 \text{ nếu } l \in g \\ -1 \text{ nếu } l \notin g \end{cases}$$

Do đó, chúng ta có thể xác định tập hợp g sẽ có dạng một vector $\{-1, +1\}^K$, và để đơn giản hóa ký hiệu, ta có thể định nghĩa hàm kết quả đầu ra của mô hình $H: X \times Y \rightarrow \{-1, +1\}$ dưới dạng:

$$H(x, l) = H(x)[l] = \begin{cases} +1 \text{ nếu } l \in H(x) \\ -1 \text{ nếu } l \notin H(x) \end{cases}$$

Công thức tính Hamming loss có thể được viết lại như sau:

$$hloss_D(H) = \frac{1}{K} \sum_{l \in Y} Pr_{(x,g)\sim D}\left[H(x, l) \neq g[l]\right]$$

Chúng ta vẫn sử dụng ý tưởng phạt các dự đoán sai bằng hàm mất mát mũ như trước đó. Vì vậy, công thức tính tổng trọng số dựa trên phân phối hiện tại của tập huấn luyện cho lần lặp thứ t có thể được viết như sau:

$$Z_t = \sum_{i=1}^{m} \sum_{l \in Y} D_t(i, l) e^{-\alpha_t g_i[l] h_t(x_i, l)}$$

Và công thức tính hamming loss cho lần lặp thứ t sẽ là

$$hloss_{D_t(h_t)} = \frac{\sum_{i:g_i[l] \neq h_t(x_i,l)} \sum_{l \in Y} D_t(i, l) e^{-\alpha_t}}{Z_t}$$

2.4.3. Thuật toán AdaBoost.MH

AdaBoost.MH – M có nghĩa là multi-class và H có nghĩa là Hamming (Hamming loss) là một biến thể của AdaBoost sử dụng hàm mất mát Hamming để giải quyết bài toán phân loại đa lớp.

Dựa trên việc quy giản bài toán về phân loại nhị phân như đã trình bày ở trên, việc sử dụng boosting để giảm thiểu hamming loss trở nên khá đơn giản. Ý tưởng chính ở đây là thay thế mỗi mẫu huấn luyện (x_i, g_i) thành K mẫu $\big((x_i, l), g_i[l]\big)$ với $l \in Y$. Nói cách khác, mỗi mẫu thực chất là một cặp dữ liệu-nhãn có dạng (x_i, l) với nhãn nhị phân là +1 nếu $l \in g_i$ và nhãn là -1 trong trường hợp còn lại.

Mã giả của thuật toán AdaBoost.MH sẽ có dạng như sau [6]:

- Cho tập dữ liệu huấn luyện: $(x_1, g_1), \dots, (x_m, g_m)$ với $x_i \in X, g_i \subseteq Y$.
- Khởi tạo trọng số cho từng mẫu dữ liệu với từng nhãn:

$$D_1(i, l) = \frac{1}{mK} \qquad \text{với } i = 1, \dots, m;\ l \in Y; K = |Y|$$

- Lặp lại quá trình sau T lần, với t = 1, …, T:
  - (1) Huấn luyện mô hình yếu với phân phối D_t.
  - (2) Trích xuất nhãn dự đoán $h_t: X \times Y \rightarrow \mathbb{R}$.
  - (3) Chọn $\alpha_t \in \mathbb{R}$.
  - (4) Chọn h_t và α_t sao cho cực tiểu hóa vector chuẩn hóa

$$Z_t = \sum_{i=1}^{m} \sum_{l \in Y} D_t(i, l) e^{-\alpha_t g_i[l] h_t(x_i, l)}$$

  - (5) Cập nhật trong số cho mỗi nhãn $l \in Y$ và cho từng mẫu dữ liệu $i = 1, \dots, m$:

$$D_{t+1}(i, l) = \frac{D_t(i, l) e^{-\alpha_t g_i[l] h_t(x_i, l)}}{Z_t}$$

- Sau khi đã có được T mô hình yếu, đầu ra của mô hình dự đoán sẽ được tính bằng cách:

$$H(x, l) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x, l)\right)$$

Như được thể hiện trong mã giả bên trên, AdaBoost.MH duy trì một phân phối D_t trên các mẫu i và nhãn l. Ở vòng lặp t, thuật toán học yếu nhận phân phối D_t và tạo ra một bộ phân loại yếu $h_t: X \times Y \to \mathbb{R}$. Chúng ta có thể hiểu $h_t(x, l)$ là một dự đoán xem có nên gán nhãn l cho x hay không, được thể hiện bằng dấu của của dự đoán (với độ lớn thể hiện độ tin cậy).

2.5. AdaBoost.MR

2.5.1. Ranking Loss

Một cách tiếp cận khác cho nhiệm vụ phân loại đa lớp – đa nhãn (Multi-Class Multi-Label Classification) đó là dựa trên việc xếp hạng (Ranking). Ý tưởng là coi bất kỳ nhiệm vụ phân lớp nào thành nhiệm vụ xếp hạng, trong đó mục tiêu là xếp hạng nhãn có thể được gán cho một bản ghi x từ "có khả năng cao nhất" đến "ít khả năng nhất". [6]

Giống như đã trình bày trong phần trước AdaBoost.MH. Mỗi mẫu huấn luyện là một tập hợp (x_i, g_i), trong đó $x_i \in X$ và $g_i \subseteq Y$. Giả định học yếu sẽ có dạng $f: X \times Y \to \mathbb{R}$. Giá trị của hàm $f(x, l)$ sẽ thể hiện thứ hạng của nhãn l với bản ghi x. Có nghĩa là nhãn l_0 sẽ được xếp hạng cao hơn nhãn l_1 đối với mẫu x nếu $f(x, l_0) > f(x, l_1)$. Đối với mỗi mẫu dữ liệu (x, g), chỉ cần quan tâm đến thứ hạng của các cặp nhãn quan trọng l_0, l_1 mà $l_1 \in g$ và $l_0 \notin g$ .Giả định sẽ dự đoán sai nếu $f(x, l_0) \geq f(x, l_1)$, có nghĩa là f dự đoán sai thứ hạng của l_0 và l_1 (thứ hạng của l_1 phải cao hơn của l_0 mới đúng). Mỗi mẫu huấn luyện này được xem như một Quasi-bipartite layered feedback, thể hiện rằng mỗi nhãn trong g_i sẽ được xếp hạng cao hơn tất cả các nhãn còn lại trong $Y - g_i$. [6]

Mục tiêu ở đây là giảm thiểu tỷ lệ trung bình các cặp quan trọng bị xếp hạng sai. Giá trị này được gọi là lỗi xếp hạng (ranking loss), được biểu riễn như sau:

$$rloss_D f = E_{(x,g)\sim D}\left[\frac{|\{(l_0, l_1) \in (Y - g) \times g : f(x, l_0) \geq f(x, l_1)\}|}{|g||Y - g|}\right]$$

2.5.2. Thuật toán AdaBoost.MR

Từ đây, ta có thuật toán AdaBoost.MR, trong đó M thể hiện cho Multi-Class, R thể hiện cho Ranking Loss như sau [6]:

- Cho tập dữ liệu huấn luyện: $(x_1, g_1), \dots, (x_m, g_m)$ với $x_i \in X, g_i \subseteq Y$.
- Khởi tạo trọng số cho từng mẫu dữ liệu với từng nhãn:

$$D_1(i, l_0, l_1) = \begin{cases} \dfrac{1}{m|g_i||Y - g_i|} & \text{Nếu } l_0 \notin g_i \text{ và } l_1 \in g_i \\ 0 & \text{Còn lại} \end{cases}$$

- Lặp lại quá trình sau T lần, với t = 1, …, T:

(1) Huấn luyện mô hình yếu với phân phối D_t.

(2) Trích xuất nhãn dự đoán $h_t: X \times Y \rightarrow \mathbb{R}$.

(3) Chọn $\alpha_t \in \mathbb{R}$.

(4) Cập nhật trọng số cho mỗi nhãn cặp (l_0, l_1) và cho từng mẫu dữ liệu $i = 1, \dots, m$:

$$D_{t+1}(i, l_0, l_1) = \frac{D_t(i, l_0, l_1) e^{\frac{1}{2}\alpha_t (h_t(x_i, l_0) - h_t(x_i, l_1))}}{Z_t}$$

Với Z_t là vector chuẩn hóa.

- Sau khi đã có được T mô hình yếu, đầu ra của mô hình dự đoán sẽ được tính bằng cách:

$$f(x, l) = \sum_{t=1}^{T} \alpha_t h_t(x, l)$$

Tương tự như thuật toán AdaBoost nguyên bản, để phạt lỗi theo phân phối D, ta sử dụng hàm mất mát mũ (hay còn gọi trong trường hợp này là Ranking Exponent Loss):

$$Z = \sum_{i, l_0, l_1} D(i, l_0, l_1) e^{\frac{1}{2}\alpha (h(x_i, l_0) - h(x_i, l_1))}$$

Trọng số của từng weak learner: α_t , có thể được chọn để cực tiểu hóa hàm Ranking Loss:

$$\sum_{i, l_0, l_1} D_1(i, l_0, l_1) [\![f(x, l_0) \geq f(x, l_1)]\!]$$

Với $[\![x]\!] = 1$ nếu x đúng và $[\![x]\!] = 0$ trong trường hợp ngược lại.

Nếu $f(x, l_0) \geq f(x, l_1)$, ta có $e^{\frac{1}{2}(f_t(x_i, l_0) - f_t(x_i, l_1))} \geq 1$, nên:

$$[\![f(x,l_0) \geq f(x,l_1)]\!] \leq e^{\frac{1}{2}(f_t(x_i,l_0)-f_t(x_i,l_1))}$$

Từ đó ta có thể tìm được giới hạn của hàm ranking loss bằng:

$$\sum_{i,l_0,l_1} D_1(i,l_0,l_1)[\![f(x,l_0) \geq f(x,l_1)]\!] \leq \sum_{i,l_0,l_1} D_1(i,l_0,l_1)e^{\frac{1}{2}(f(x_i,l_0)-f(x_i,l_1))}$$

Ta lại có:

$$D_{T+1}(i,l_0,l_1) = \frac{D_1(i,l_0,l_1)e^{\frac{1}{2}\sum \alpha_t(h_t(x_i,l_0)-h_t(x_i,l_1))}}{\prod_{t=1}^{T} Z_t} = \frac{D_1(i,l_0,l_1)e^{\frac{1}{2}(f(x_i,l_0)-f(x_i,l_1))}}{\prod_{t=1}^{T} Z_t}$$

Nên, ta có:

$$\sum_{i,l_0,l_1} D_1(i,l_0,l_1)[\![f(x,l_0) \geq f(x,l_1)]\!] \leq \sum_{i,l_0,l_1} D_{T+1}(i,l_0,l_1) \prod_{t=1}^{T} Z_t$$

$$= \prod_{t=1}^{T} Z_t$$

Vì vậy, cực tiểu hóa hàm ranking loss đồng nghĩa với việc cực tiểu hóa hàm Z_t.

Trong trường hợp h chỉ nhận giá trị trong tập $\{-1,+1\}$, ta có:

$$u = \frac{1}{2}\big(h(x_i,l_0) - h(x_i,l_1)\big) \in \{-1,0,+1\}$$

Thay vào công thức tính Z:

$$Z = \sum_{i,l_0,l_1} D(i,l_0,l_1)e^{\alpha u_i} = \sum_{b\in\{-1,0,+1\}} \sum_{i:u_i=b,l_0,l_1} D(i,l_0,l_1)e^{\alpha b}$$

$$= W_0 + W_- e^{\alpha} + W_+ e^{-\alpha}$$

Với

$$W_b = \sum_{i,l_0,l_1} D(i,l_0,l_1)[\![h(x_i,l_0) - h(x_i,l_1) = 2b]\!]$$

Để tìm cực tiểu của Z, ta tìm cực trị của hàm này (vì Z là một hàm lồi):

$$\frac{dZ}{d\alpha} = W_- e^{\alpha} - W_+ e^{-\alpha}$$

$$\frac{dZ}{d\alpha} = 0 \Leftrightarrow W_- e^{\alpha} = W_+ e^{-\alpha} \Leftrightarrow \alpha = \frac{1}{2}\log\left(\frac{W_+}{W_-}\right)$$

Khi đó, $Z = W_0 + 2\sqrt{W_- W_+}$.

Đây chỉ là trường hợp đơn giản khi mô hình yếu h trả về nhãn cố định thuộc $\{-1, +1\}$. Trong các trường hợp khác phức tạp hơn, khi đầu ra của h là một giá trị thực ($h: Y \to \mathbb{R}$). Bằng cách gộp $\alpha/2$ vào h (coi $\alpha/2$ là một phần của h), ta có thể đơn giản hóa Z mà không làm ảnh hưởng đến kết quả cuối cùng của phương án tối ưu:

$$Z = \sum_{i,l_0,l_1} D_t(i, l_0, l_1) e^{\left(h(x_i,l_0) - h(x_i,l_1)\right)} = \sum_{l_0,l_1} \left[\left(\sum_i D_t(i, l_0, l_1) \right) e^{\left(h(x_i,l_0) - h(x_i,l_1)\right)} \right]$$

Biểu thức trên có thể được viết lại dưới dạng:

$$Z = \sum_{l_0,l_1} \left[w(l_0, l_1) e^{\left(h(x_i,l_0) - h(x_i,l_1)\right)} \right]$$

Với $w(l_0, l_1) = \sum_i D_t(i, l_0, l_1)$, ta có:

$$Z \leq \left(\frac{1-r}{2}\right) e^{\alpha} + \left(\frac{1+r}{2}\right) e^{-\alpha}$$

Với:

$$r = \frac{1}{2} \sum_{i,l_0,l_1} D(i, l_0, l_1) \big(h(x_i, l_1) - h(x_i, l_0) \big)$$

Tương tự như trường hợp đơn giản đã được trình bày bên trên, vế phải của bất đẳng thức sẽ được cực tiểu khi

$$\alpha = \frac{1}{2} \log\left(\frac{1+r}{1-r}\right)$$

Và $Z \leq \sqrt{1 - r^2}$. Vì vậy, mục tiêu của weak learner là cực đại hóa $|r|$, có nghĩa là cố gắng xếp hạng đúng l_0, l_1 ($h(x_i, l_1) > h(x_i, l_0)$) nhiều nhất có thể.

CHƯƠNG 3. ỨNG DỤNG CHO BÀI TOÁN PHÂN LỚP

3.1. Giới thiệu các bộ dữ liệu

Các bộ dữ liệu được thực hiện phân lớp với các thuật toán AdaBoost là các bộ dữ liệu: Marketing and Sales [7], phân loại đậu khô [8], Yelp dataset [9], Yeast [10]. Mô tả khái quát của các bộ dữ liệu được thể hiện ở hình dưới đây:

Bảng 3-1 Mô tả khát quát các bộ dữ liệu được sử dụng

Bộ dữ liệu	Tổng số bản ghi	Số thuộc tính	Kiểu phân lớp	Số lớp
Marketing and Sales	45,211	16	Nhị phân	2
Đậu khô	13,166	16	Đa lớp	7
Yelp	10,806	676	Đa nhãn	5
Yeast	2,417	103	Đa nhãn	14

Bộ dữ liệu Marketing and Sales là về chiến dịch quảng bá của một ngân hàng Bồ Đào Nha. Bộ dữ liệu chứa thông tin cơ bản như tuổi, nghề nghiệp, tình trạng hôn nhân, v.v. của các khách hàng và nhãn có hai giá trị là “no” và “yes” thể hiện khách hàng đăng ký dịch vụ gửi tiền ở ngân hàng.

Bộ dữ liệu đậu khô là về việc phân lớp các hạt đậu trong bộ dữ liệu vào 1 trong 7 nhóm đậu khô. Bộ dữ liệu đã được xử lý sẵn từ dạng hình ảnh sang dạng số lưu trữ các thông tin liên quan đến kích thước của hạt đậu.

Yelp là một bộ dữ liệu liên quan đến đánh giá của khách hàng cho nhiều lĩnh vực khác nhau dưới dạng văn bản. Trong báo cáo này, bộ dữ liệu Yelp liên quan đến việc đánh giá của khách hàng cho nhà hàng với các nhãn liên quan đến chất lượng đồ ăn, dịch vụ, bầu không khí, giao dịch và giá cả. Bộ dữ liệu đã được xử lý sẵn từ dạng văn bản thành dạng token.

Cuối cùng, bộ dữ liệu Yeast liên quan đến việc dự đoán vị trí của protein trong tế bào. Các thuộc tính của bản ghi liên quan đến các tín hiệu độ được từ các phương pháp khác nhau.

3.2. Kết quả phân lớp trên các bộ dữ liệu

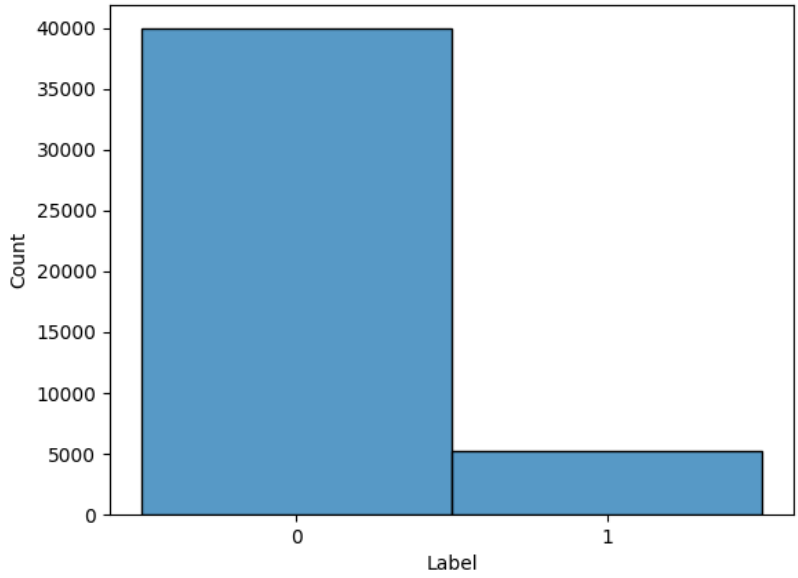
Kết quả phân lớp trên các bộ dữ liệu được thực hiện trên nhiều thuật toán, mô hình khác nhau như SVM, Logistic Regression, Random Forest để so sánh với các thuật toán AdaBoost. Bộ phân lớp cở sở được sử dụng cho thuật toán AdaBoost là Decesion Tree với các tham số phù hợp bằng cách thử lần lượt và số lần lặp tối đa để huấn luyện là 50 lần với toàn bộ thử nghiệm.

Việc thử nghiệm các tham số khác nhau cho thuật toán Decesion Tree sử dụng tham số duy nhất là chiều sâu tối đa của cây (max depth) là biến để thử nghiệm. Khi thu được kết ứng với giá trị max depth, lựa chọn giá trị cho ra kết quả accuracy và f1-score thích hợp nhất để thực hiện với các thuật toán AdaBoost và Random Forest.

Cuối cùng, kết quả thử nghiệm được thực hiện với quy trình K-fold với k=5. Quy trình K-fold là một quy trình đơn giản mà hiệu quả để có thể đánh giá được khả năng khái quát của mô hình. Trong mỗi fold, tính các thang đo accuracy và f1-score trên các tập train, validate và test. Và kết quả cuối cùng sẽ được lấy trung bình từ kết quả của các fold.

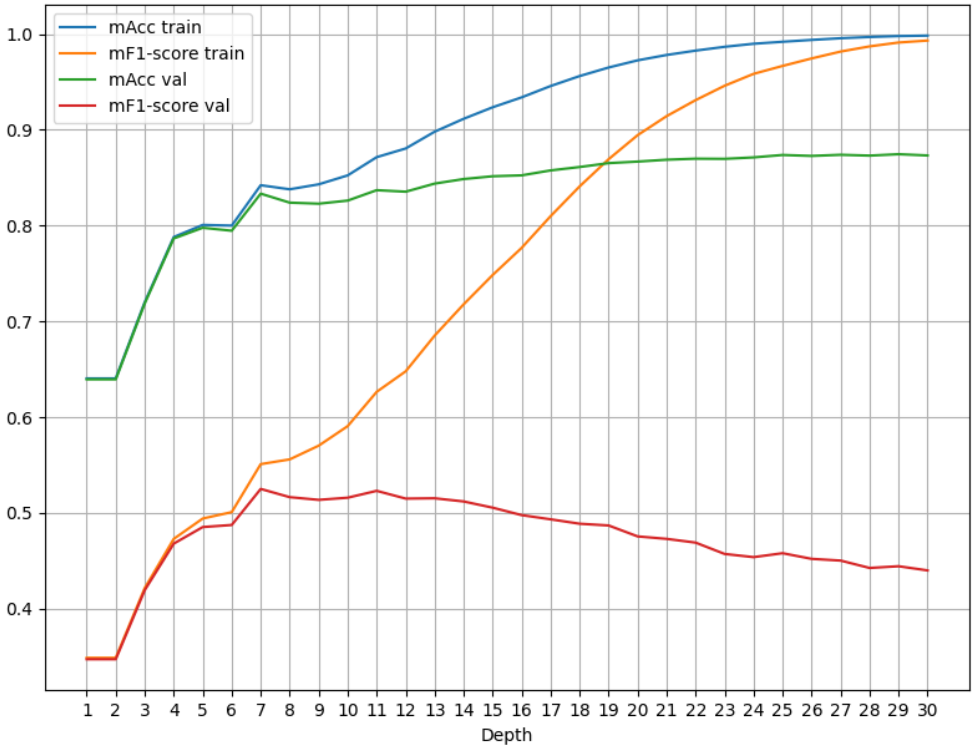
3.2.1. Kết quả thuật toán AdaBoost trên bộ dữ liệu Marketing and Sales

Trong bộ dữ liệu, 45.211 bản ghi, trong đó 39.922 bản ghi có nhãn là “no” và 5.289 bản ghi có nhãn là “yes”. Có thể thấy, bộ dữ liệu bị mất cân bằng nặng với tỉ lệ xấp xỉ 90:10. Một số giải pháp đơn giản là đánh trọng số cho các lớp, các lớp thuộc nhóm đa số sẽ được đánh trọng số thấp, các lớp thuộc nhóm thiểu số sẽ được đánh trọng số cao. Trong thử nghiệm này, trọng số của các lớp (“no”, “yes”) là (0.56, 4.27).



Hình 3.1 Biểu đồ thể hiện phân bố nhãn trong bộ dữ liệu Marketing and Sales

Dưới đây là kết quả thực nghiệm tìm kiếm tham số chiều sâu sao cho thuật toán Decision Tree đạt kết quả thích hợp nhất:



Hình 3.2 Kết quả thuật toán Decesion Tree với các tham số độ sâu tối đa khác nhau trên bộ dữ liệu Marketing and Sales

Dựa vào hình trên, có thể thấy kết quả phù hợp nhất là với tham số chiều sâu nằm trong khoảng [7, 11] vì các giá trị nằm ngoài khoảng này cho ra kết quả F1-score

thấp hơn. Hơn thế, với max depth > 14 thì mô hình có hiện tượng over-fitting khi giá trị F1- score trên tập train tăng nhưng lại giảm ở tập validate.

Cuối cùng, thực hiện phân lớp với bộ dữ liệu Market and Sales sử dụng nhiều mô hình khác nhau. Trong đó, bộ phân lớp cơ sở cho thuật toán AdaBoost là Decesion Tree với tham số max_depth=7. Dưới đây là bảng kết quả khi thực hiện đánh giá bộ dữ liệu với quy trình K-fold với k=5 trên độ đo là accuracy và f1-score:

Bảng 3-2 Bảng kết quả trên bộ dữ liệu Market and Sales với nhiều mô hình khác nhau

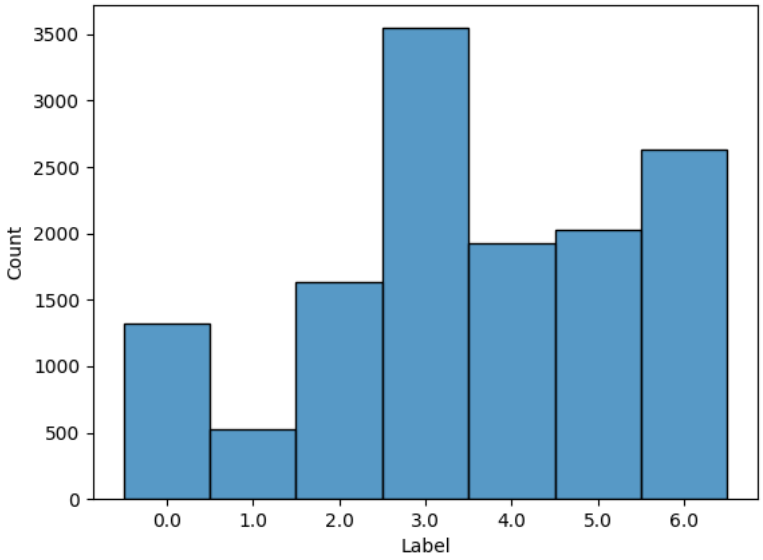
Dữ liệu / **Mô hình**	Tập train		Tập validate		Tập test	
	mAcc	mF1-score	mAcc	mF1-score	mAcc	mF1-score
Logistic Regression	0.793	0.474	0.792	0.472	0.790	0.472
SVM	0.778	0.421	0.777	0.417	0.781	0.431
Decision Tree	0.842	0.555	0.833	0.524	0.835	0.535
Random Forest	0.827	0.544	0.821	0.528	0.824	0.539
AdaBoost	**0.925**	**0.753**	**0.873**	**0.586**	**0.877**	**0.588**

Dựa vào Bảng 3-1, có thể nhận thấy kết quả từ các phương pháp học kết hợp mang lại kết quả tốt hơn so với kết quả các các phương pháp một mô hình. Kết quả của phương pháp AdaBoost mang lại kết quả cao nhất cho bộ dữ liệu Marketing and Sales.

3.2.2. Kết quả thuật toán AdaBoost.M1 trên bộ dữ liệu phân loại đậu khô

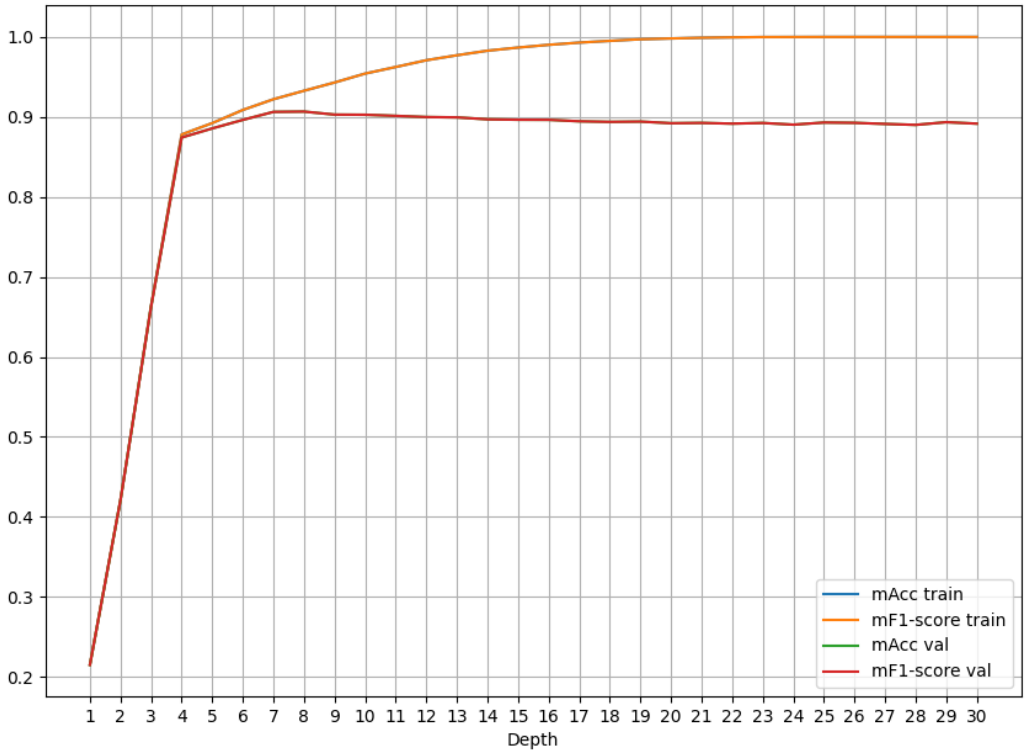
Bộ dữ liệu phân loại đậu khô đã được xử lý thành dạng số, có các trường thuộc tính như hình dạng, kích thước, v.v. Sự phân bố nhãn trong bộ dữ liệu được thể hiện ở Hình 3.3, có thể thấy bộ dữ liệu bị mất cân bằng ở một số nhãn. Giải pháp cũng thực hiện giống với khi thực hiện hiện phân lớp cho bộ dữ liệu Market and Sales, đó là đánh trọng số cho các lớp. Trọng số của các lớp (SEKER, BARBUNYA, BOMBAY, CALI,

HOROZ, SIRA, DERMASON) là (0.95, 1.47, 3.72, 1.19, 1.00, 0.73, 0.54). Ngoài ra, trước khi được sử dụng để phân lớp, bộ dữ liệu được chuẩn hóa theo Z-score.



Hình 3.3 Biểu đồ thể hiện phân bố nhãn trong bộ dữ liệu phân loại đậu khô

Dưới đây là kết quả thực nghiệm tìm kiếm tham số chiều sâu sao cho thuật toán Decision Tree đạt kết quả thích hợp nhất:



Hình 3.4 Kết quả thuật toán Decesion Tree với các tham số độ sâu tối đa khác nhau trên bộ dữ liệu phân loại đậu khô

Dựa vào hình trên, có thể thấy kết quả phù hợp nhất là với tham số chiều sâu khó xác định hơn. Các giá trị depth > 6 mang lại kết quả tương đối như nhau ở tập validate, chỉ khác ở tập train. Dù vậy, nếu depth càng cao thì hiện tượng over-fitting càng thể hiện rõ hơn.

Cũng tương tự như khi thực hiện với thuật toán AdaBoost cho bộ dữ liệu Marketing and Sales với max_depth=7. Dưới đây là kết quả phân lớp:

Bảng 3-3 Bảng kết quả trên bộ dữ liệu phân loại đậu khô với nhiều mô hình khác nhau

Dữ liệu / **Mô hình**	Tập train		Tập validate		Tập test	
	mAcc	mF1-score	mAcc	mF1-score	mAcc	mF1-score
SVM	0.930	0.930	**0.926**	**0.926**	**0.931**	**0.931**
Decision Tree	0.918	0.918	0.906	0.906	0.913	0.913
Random Forest	0.922	0.922	0.908	0.908	0.913	0.913
AdaBoost.M1	**0.998**	**0.998**	0.919	0.919	0.919	0.919

Kết quả các phương pháp trên bộ dữ liệu phân loại đậu khô không có quá nhiều sự khác biệt so với các phương pháp khác. Phương pháp AdaBoost chỉ mang lại kết quả cao nhất ở bộ dữ liệu huấn luyện.

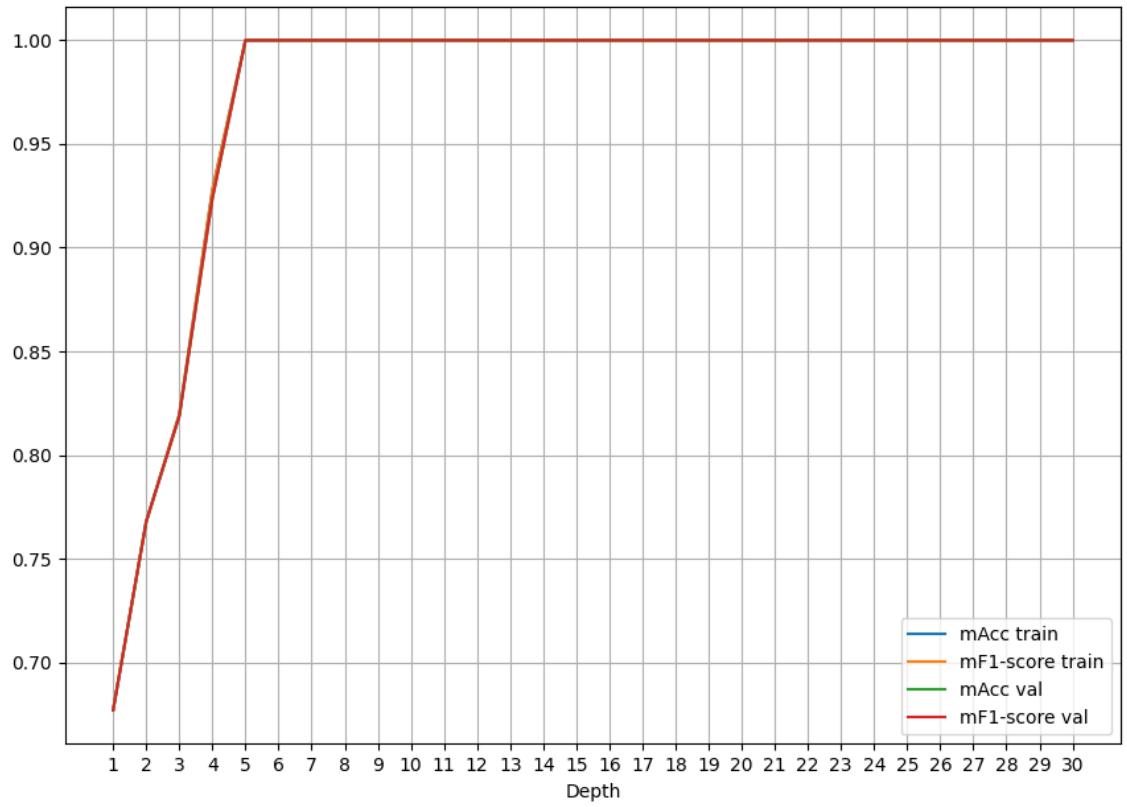
3.2.3. Kết quả thuật toán AdaBoost.MH trên bộ dữ liệu Yelp

Bộ dữ liệu được sử dụng để thực hiện thực ngiệm là bộ dữ liệu Yelp, một bộ dữ liệu liên quan đến đánh giá của khách hàng cho nhiều lĩnh vực khác nhau dưới dạng văn bản. Trong báo cáo này, bộ dữ liệu Yelp liên quan đến việc đánh giá của khách hàng cho nhà hàng với các nhãn liên quan đến chất lượng đồ ăn, dịch vụ, bầu không khí, giao dịch và giá cả. Bộ dữ liệu đã được xử lý sẵn từ dạng văn bản thành dạng token với hơn 10,000 bản ghi.

Việc phân lớp đa nhãn có trong giới hạn báo cáo này được thực hiện theo phương pháp biến đổi bài toán (Problem transformation methods). Cụ thể hơn, phương pháp được sử dụng là phương pháp PT5 [7] biến đổi bài toán thành dạng bài toán nhị phân

để thực hiện với thuật toán AdaBoost.MH. Nhãn ban đầu sẽ sở thành thuộc tính mới trong bản ghi, nhãn mới là nhãn có giá trị nhị phân.

Dưới đây là kết quả thực nghiệm tìm kiếm tham số chiều sâu sao cho thuật toán Decision Tree đạt kết quả thích hợp nhất:



Hình 3.5 Kết quả thuật toán Decesion Tree với các tham số độ sâu tối đa khác nhau trên bộ dữ Yelp

Có thể thấy, thuật toán Decision Tree đã fit hoàn toàn (perfect fit) bộ dữ liệu Yelp với max depth >= 5. Tuy vậy. việc lựa chọn max depth cao cũng không mang lại lợi ích gì nhiều và thậm chí còn có thể gây ra hiện tượng over-fitting.

Thực hiện đánh giá phân lớp bộ dữ liệu Yelp trên nhiều mô hình khác nhau, bộ phân lớp cơ sở cho Random Forest, AdaBoost.MH là Decesion Tree (max depth=5). Dưới đây là bảng kết quả khi thực hiện với nhiều mô hình khác nhau:

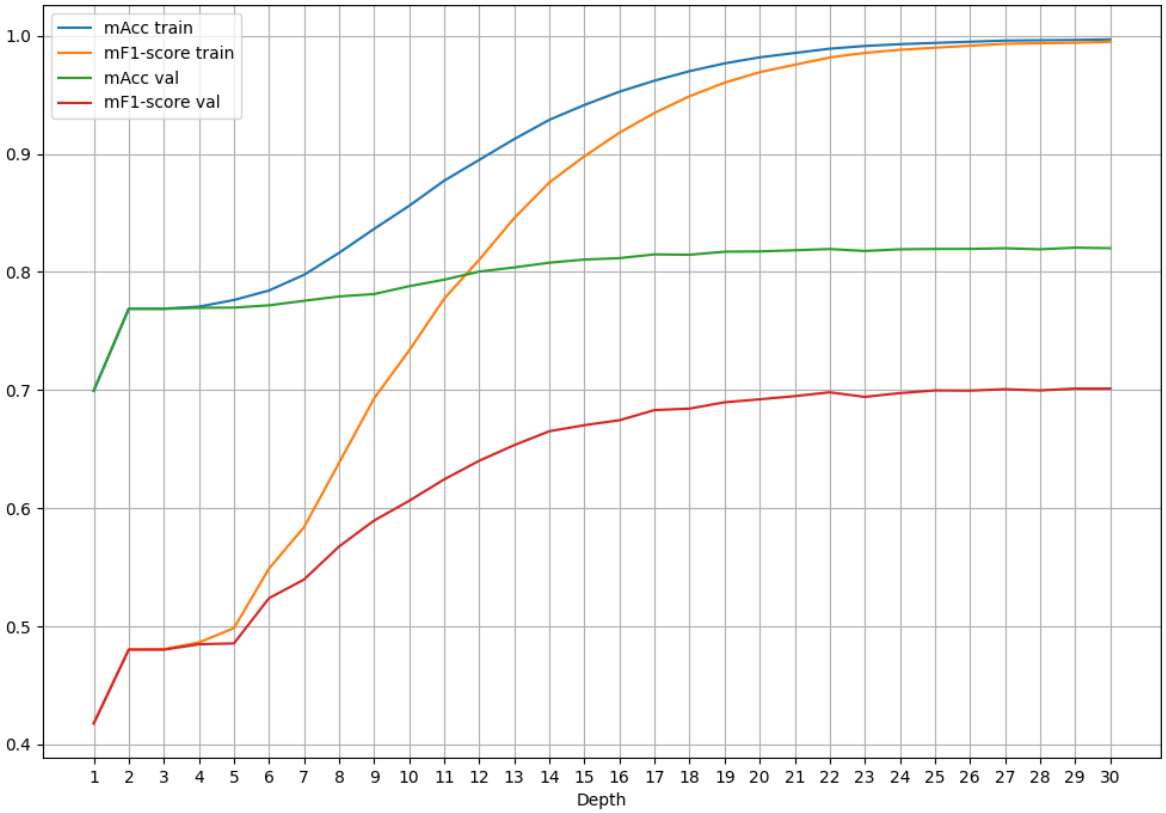
Bảng 3-4 Bảng kết quả trên bộ dữ liệu Yelp với nhiều mô hình khác nhau

Dữ liệu / Mô hình	Tập train		Tập validate		Tập test	
	mAcc	mF1-score	mAcc	mF1-score	mAcc	mF1-score
SVM	0.930	0.930	0.926	0.926	0.931	0.931
Decision Tree	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Random Forest	0.677	0.677	0.675	0.677	0.651	0.651
AdaBoost.MH	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Từ bảng kết quả trên, có thể nhận thấy phương pháp Decesion Tree mang lại kết quả cao hơn so với phương pháp SVM. Điều này có thể là do bộ dữ liệu này phù hợp với mô hình Decesion Tree. Ngoài ra, kết quả từ phương pháp Random Forest lại thấp hơn so với kết quả từ bộ phân lớp cơ sở, điều này đi ngược lại với mong đợi và cần đi sâu hơn để tìm hiểu.

3.2.4. Kết quả thuật toán AdaBoost.MR trên bộ dữ liệu Yeast

Cũng thực hiện biến đổi bài toán đa nhãn theo phương pháp PT5, đưa bài toán về dạng bài bài nhị phân và thực hiện phân lớp với quy trình K-Fold, k=5. Dưới đây là kết quả thực nghiệm tìm kiếm tham số chiều sâu sao cho thuật toán Decision Tree đạt kết quả thích hợp nhất:



Hình 3.6 Kết quả thuật toán Decesion Tree với các tham số độ sâu tối đa khác nhau trên bộ dữ Yeast

Có thể thấy, lựa chọn max depth = 20 là một lựa chọn tốt, lý do cũng nằm ở việc nếu chọn max depth lớn thì sẽ dễ xảy ra hiện tượng over-fitting. Dưới đây là bảng kết quả đạt được khi thực hiện với nhiều mô hình khác nhau:

Bảng 3-5 Bảng kết quả trên bộ dữ liệu Yelp với nhiều mô hình khác nhau

Dữ liệu / Mô hình	Tập train		Tập validate		Tập test	
	mAcc	mF1-score	mAcc	mF1-score	mAcc	mF1-score
SVM	0.698	0.0	0.698	0.0	0.696	0.0
Decision Tree	0.981	0.969	0.719	0.529	**0.817**	0.692
Random Forest	0.973	0.954	0.772	0.557	0.771	0.492
AdaBoost.MR	**1.000**	**1.000**	**0.802**	**0.802**	0.800	**0.800**

Qua bảng trên, có thể dễ dàng thấy rằng thuật toán AdaBoost.MR mang lại kết quả vượt trội trên bộ dữ liệu Yeast. Kết quả của mô hình này cao hơn hẳn so với các mô hình còn lại nếu so trên thang đô F1-score. Trái lại kết quả cao, mô hình SVM mang lại kết quả thấp nhất, đặc biệt là đạt 0.0 điểm trên thang đo F1-score. Điều này chứng tỏ mô hình SVM không thích hợp với bộ dữ liệu này.

KẾT LUẬN

Nội dung của bản báo cáo này đã trình bày khá chi tiết về thuật toán, ý tưởng và cơ sở toán học cho AdaBoost. AdaBoost, thuộc nhóm phương pháp Boosting, với ý tưởng là thay đổi trọng số của các mẫu sai trong bộ dữ liệu, khiến mô hình phải "tập trung" vào các mẫu đó. Sau quá trình huấn luyện và thay đổi trọng số thì kết quả cuối cùng thu được sẽ tốt hơn.

Thuật toán AdaBoost ban đầu chỉ được thực hiện với bộ dữ liệu phân lớp nhị phân bởi Freund và Schapire. Dần dần, các biến thể của thuật toán được phát triển, sử dụng lên các bài toán lớn hơn là phân lớp đa lớp và phân lớp đa nhãn. Mặc dù vậy, ý tưởng cốt lõi vẫn không thay đổi.

Về phần ứng dụng thuật toán AdaBoost cho các bài toán phân lớp, có thể nhận thấy các kết quả của các mô hình AdaBoost nhìn chung mang lại kết quả cao hơn khi so với các mô hình khác như SVM, Random Forest, Decision Tree. Kết quả ứng dụng cho thấy độ hiệu quả của thuật toán AdaBoost cho bài toán phân lớp vẫn còn hữu hiệu.

Mặc dù đạt được các kết quả khả quan và tốt hơn so với các mô hình phổ biến khác, kết quả trên bộ dữ liệu Marketing and Sales thể hiện rằng việc mất cân bằng dữ liệu vẫn là một vấn đề mở chưa có giải pháp cụ thể. Và AdaBoost cũng không phải ngoại lệ cho vấn đề đó.

TÀI LIỆU THAM KHẢO

[1] J. G. R. S. M. a. T. M. M. Carbonell, "An overview of machine learning," *Machine learning*, pp. 3-23, 1983.

[2] J. A. N. a. A. K. Alzubi, "Machine learning from theory to algorithms: an overview," *Journal of physics: conference series*, 2018.

[3] F. e. a. Maleki, "Overview of machine learning part 1: fundamentals and classic approaches," *Neuroimaging*, pp. e17-e32, 2020.

[4] T. O. Ayodele, "Machine learning overview," *New Advances in Machine Learning 2*, pp. 9-18, 2010.

[5] Y. M. Cha Zhang, Ensemble Machine Learning, 2012.

[6] Robert E. Schapire, Yoav Freund, Boosting: Foundations and Algorithms, The MIT Press, 2012.

[7] Moro,S., Rita,P., and Cortez,P., *Bank Marketing.*, UCI Machine Learning Repository, 2012.

[8] KOKLU, M. and OZKAN, Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques., Computers and Electronics in Agriculture, 2020.

[9] "Yelp Dataset Challenge," Yelp, 2013. [Online]. Available: http://www.yelp.com/dataset_challenge/. [Accessed 1 5 2024].

[10] Nakai,Kenta, *Yeast*, UCI Machine Learning Repository, 1996.