TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN GIỮA KÌ**

**HỌC MÁY**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**TIỂU LUẬN GIỮA KÌ**

**HỌC MÁY**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những “lời cảm ơn” đã có.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của **PGS.TS. Lê Anh Cường**. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Hoàng Đình Quý Vũ*

*Lê Huỳnh Huyền Trang*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

MỤC LỤC

[MỤC LỤC 1](#_Toc169202778)

[CHƯƠNG 1 – Tổng Quan và Ứng Dụng Word2Vec trong Xử Lý Ngôn Ngữ Tự Nhiên 5](#_Toc169202779)

[1.1 Bootstrapping là gì? 5](#_Toc169202780)

[1.2 Bagging là gì? 6](#_Toc169202781)

[1.3. Giải thuật Random Forest 7](#_Toc169202782)

[1.3.1 Định nghĩa: 8](#_Toc169202783)

[1.3.2 Xây dựng thuật toán Random Forest 10](#_Toc169202784)

[1.4. AdaBoosting 11](#_Toc169202785)

[1.4.1 Các bước của thuật toán AdaBoosting 13](#_Toc169202786)

[1.5 Gradient Boosting 18](#_Toc169202787)

[Các bước của thuật toán Gradient Boosting 21](#_Toc169202788)

[CHƯƠNG 2 – KHÁM PHÁ THUẬT TOÁN VÀ KỸ THUẬT TỐI ƯU TRONG MẠNG NEURAL 23](#_Toc169202789)

[2.1 Feed Forward Neural Network 23](#_Toc169202790)

[2.1.1 Cấu trúc của FFNN 24](#_Toc169202791)

[2.1.1.1 Các tầng - Layers 24](#_Toc169202792)

[2.1.1.2 Nodes – Units 25](#_Toc169202793)

[2.1.1.3 Activation functions 26](#_Toc169202794)

[2.1.2 Quá trình hoạt động của FFNN 32](#_Toc169202795)

[2.1.3 Hàm mất mát (Loss Function): 33](#_Toc169202796)

[2.1.4 Lan truyền ngược (Backpropagation): 34](#_Toc169202797)

[2.1.5 Thuật toán tối ưu hóa 36](#_Toc169202798)

[2.1.6 Ví dụ chạy 36](#_Toc169202799)

[2.2 Các phương pháp optimization 41](#_Toc169202800)

[2.2.1 Gradient Descent 42](#_Toc169202801)

[2.2.2 Stochastic Gradient Descent 43](#_Toc169202802)

[2.2.2 Gradient Descent với Momentum 45](#_Toc169202803)

[2.2.4 Adagrad 46](#_Toc169202804)

[2.2.5 RMSprop 47](#_Toc169202805)

[2.2.5 Adam 47](#_Toc169202806)

[TÀI LIỆU THAM KHẢO 49](#_Toc169202807)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

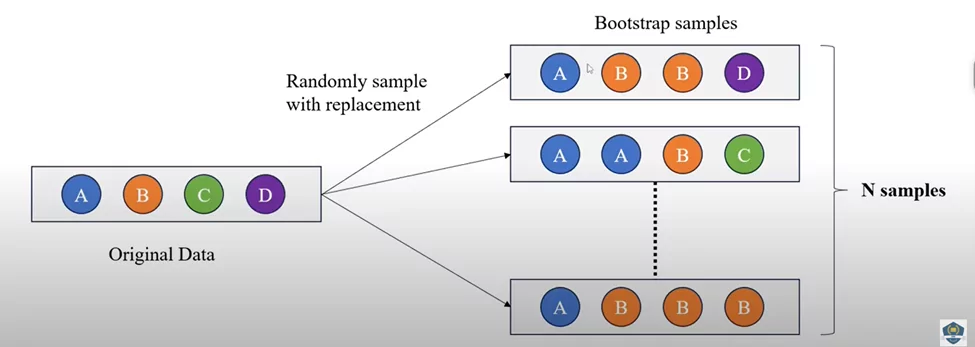
[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

1.1 Bootstrapping là gì?

Bootstrapping là kỹ thuật mà chúng ta tạo nên được những mẫu mới từ các bộ dữ liệu gốc, phương pháp này thường được sử dụng ở trong thống kê cũng như trong máy học. Kỹ thuật này giúp chúng ta có thể tạo nên được bộ dữ liệu đa dạng hơn từ bộ dữ liệu gốc duy nhất.



Ta có bộ dữ liệu gốc gồm 4 thành viên là A , B, C , D.

Tạo nên bộ dữ liệu mẫu bằng cách:

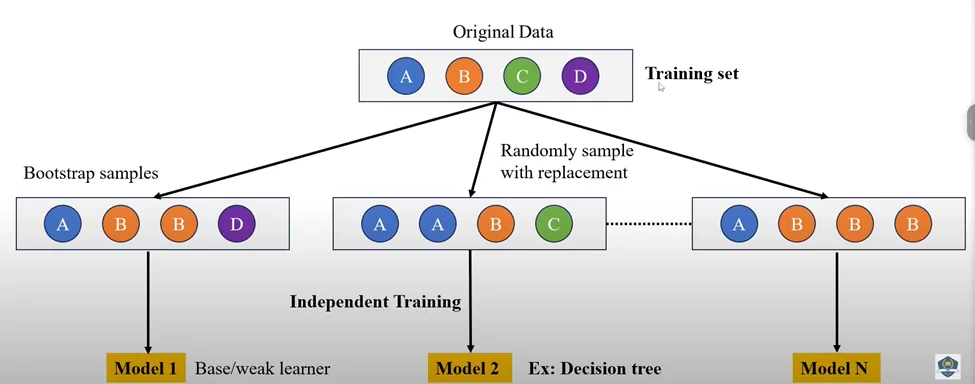
* Lấy ngẫu nhiên một dữ liệu ở trong bộ gốc.
* Mỗi thành viên của bộ dữ liệu gốc có thể được lấy nhiều lần.
* Lấy ngẫu nhiên, không nhất thiết phải có đủ tất cả các thành viên trong bộ dữ liệu gốc.
* Phương pháp Bootstrapping là phương pháp tạo mẫu cơ bản được sử dụng trong Bagging cũng như trong giải thuật Random Forest.

**Lợi ích của Bootstrapping**

Với giải thuật Bootstrapping, từ một bộ dữ liệu ban đầu chúng ta sẽ có N bộ dữ liệu khác nhau. Khi chúng ta sử dụng để huấn luyện chương trình, chúng ta sẽ có N mô hình khác nhau tương ứng với N bộ dữ liệu mới. Điều này giúp cho, dữ liệu đa dạng hơn và giúp giảm được sai số.

1.2 Bagging là gì?

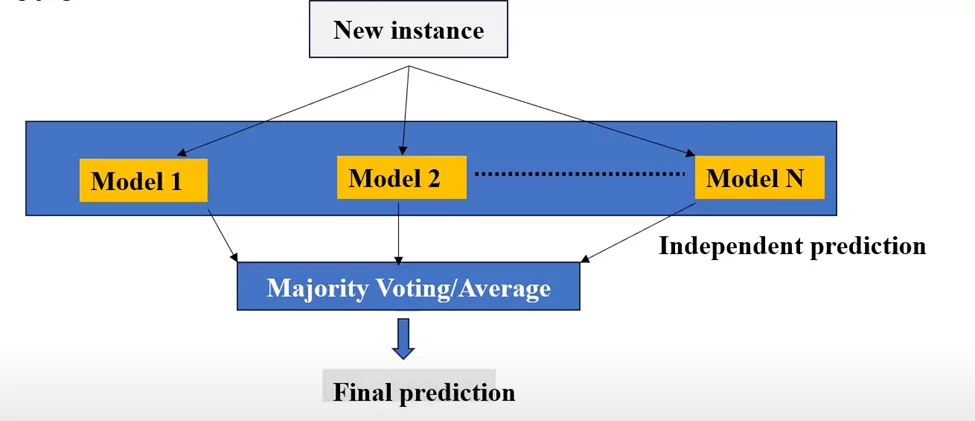
Bagging là viết tắt của Bootstrap Aggregating, có nghĩa là đầu tiên nó sẽ tạo nên các bộ dữ liệu mới theo phương pháp Bootstrapping. Tiếp theo, sẽ huấn luyện ra N model khác nhau dựa trên N bộ dữ liệu mới. Cuối cùng, khi thực hiện dự đoán, nó sẽ tổng hợp N kết quả lại.



Các model trong Bagging là những model độc lập, hoàn toàn không liên quan gì tới nhau. Tuy nhiên các Model trong bagging có chung một loại giải thuật, các model khác nhau là do dữ liệu huấn luyện khác nhau.

Những giải thuật này thường được gọi là Base/Weak learner. Bởi vì nó không đủ mạng nên ta phải cần nhiều model. Kết quả cuối cùng sẽ tổng hợp từ các Base/Weak learner, nên người ta tin rằng kết quả sẽ tốt hơn, sẽ mạnh hơn.

Như vậy, sau khi huấn luyện xong, ta sẽ có N model khác nhau.



Nếu bạn đưa một input mới vào, thì N model sẽ dự đoán ra N kết quả khác nhau. Kết quả cuối cùng được tính dựa trên nguyên tắc Voting hoặc Average từ N kết quả trên.

* Đối với giải thuật hồi quy thì sẽ dùng nguyên tắc Average.
* Đối với giải thuật phân loại thì sẽ dùng nguyên tắc Voting.

1.3. Giải thuật Random Forest

Là giải thuật rất đặc trưng của phương pháp Bagging đã đề cập ở phần trước, và nó được xây dựng dựa trên giải thuật Cây quyết định.

Sau đây, mình sẽ tóm tắt một số ưu khuyết điểm chính của Cây quyết định.

Ưu điểm lớn nhất của cây quyết định là dễ tạo nên, dễ sử dụng và khá dễ để giải thích, nó không có những biểu thức toán học phức tạp. Tuy nhiên, một nhược điểm rất lớn đối với giải thuật cây quyết định là nó rất nhạy cảm đối với Training data. Nếu ta thay đổi Training data thì kết quả của cây quyết định nó sẽ bị khác đi, do đó, nó có sai số phương sai khá lớn, dẫn đến hiện tượng Overfitting.

Điều này có thể được khắc phục bởi giải thuật Rừng ngẫu nhiên.

1.3.1 Định nghĩa:

Random forest là một phương pháp thống kê mô hình hóa bằng máy (machine learning statistic) dùng để phục vụ các mục đích phân loại, tính hồi quy và các nhiệm vụ khác bằng cách xây dựng nhiều cây quyết định (Decision tree). Một cây quyết định là một cách đơn giản để biểu diễn một giao thức (Protocol).

Nói cách khác, cây quyết định biểu diễn một kế hoạch, trả lời câu hỏi phải làm gì trong một hoàn cảnh nhất định. Mỗi Node của cây sẽ là các thuộc tính, và các nhánh là giá trị lựa chọn của thuộc tính đó. Bằng cách đi theo các giá trị thuộc tính trên cây, cây quyết định sẽ cho ta biết giá trị dự đoán. Nhóm thuật toán cây quyết định có một điểm mạnh đó là có thể sử dụng cho cả bài toán Phân loại (Classification) và Hồi quy (Regression). Random Forest có khả năng tìm ra thuộc tính nào quan trọng hơn so với những thuộc tính khác.

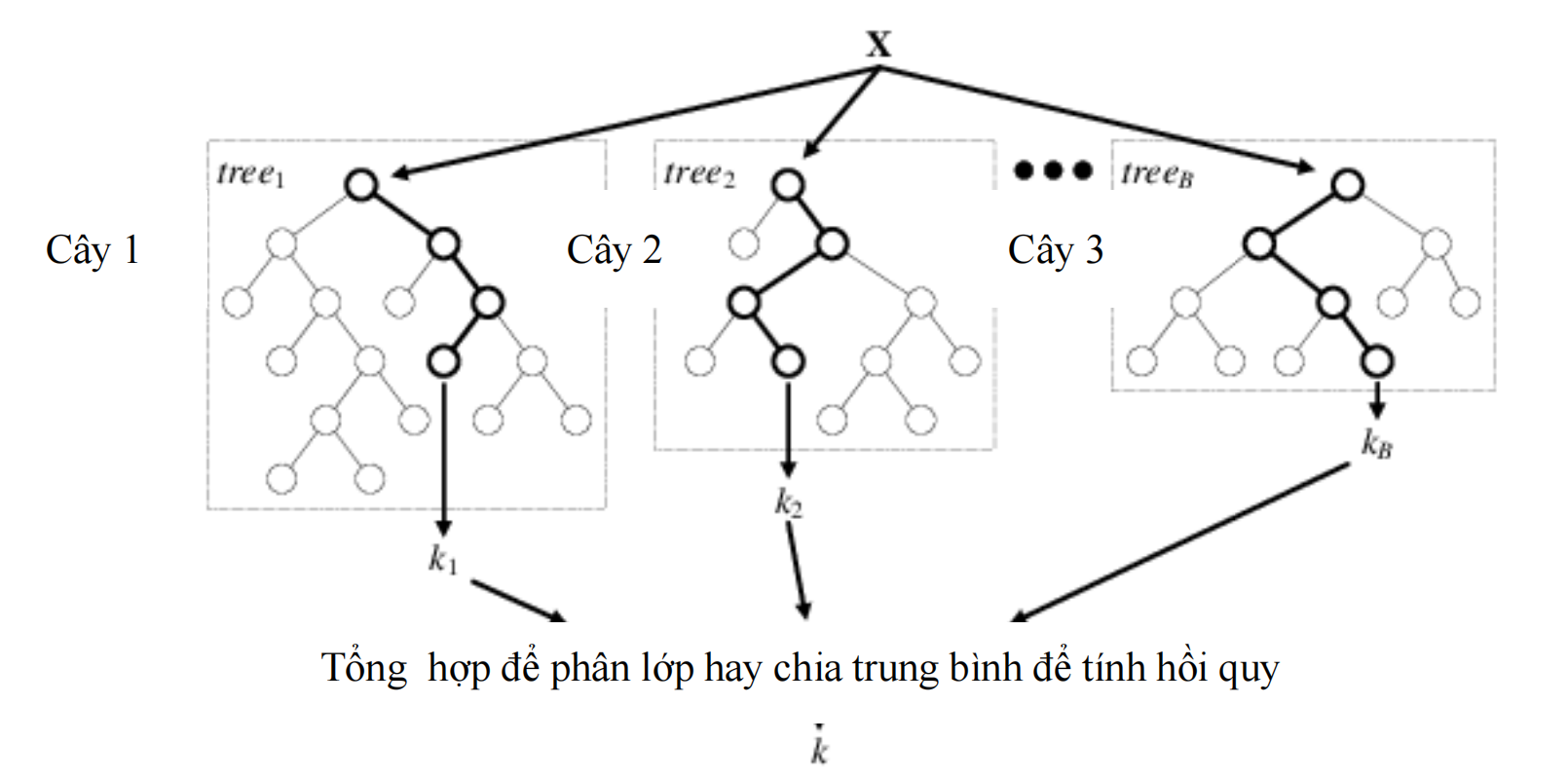
Trên thực tế, nó còn có thể chỉ ra rằng một số thuộc tính là không có tác dụng trong cây quyết định.

Từ hình 1 chúng ta thấy rằng Random Forest được cấu thành bởi một số cây quyết định.

Thuật toán lấy mẫu cho phương pháp random forest ứng dụng cho các phương pháp sử dụng thuật toán mô tả thống kê để ước lượng số lượng từ một mẫu dữ liệu (bagging).

Một tập mẫu X = x1, ..., xn với các câu trả lời Y = y1, ..., yn, lấy giá trị trung bình (B lần), chọn một mẫu ngẫu nhiên từ bộ mẫu phù hợp với cây quyết định:

Lặp b = 1,…, B; n mẫu từ giá trị tọa độ (X, Y); gọi là (Xb, Yb); lớp dữ liệu hay kết quả hồi quy fb của biến Xb, Yb;



Sau khi lấy mẫu, các phép tính toán cho các mẫu là ẩn số x’ có thể được thực hiện bằng cách lấy trung bình các giá trị nội suy từ tất cả các cây hồi quy riêng lẻ của biến x' hoặc lấy giá trị từ đa số của các mẫu trong cây quyết định:

Random forest có thể sắp xếp sự quan trọng của các biến trong các bài toán phân loại hay hồi quy, được mô tả trong các nghiên cứu của Breiman, xác định các biến quan trọng trong 1 tập dữ liệu là làm phù hợp phương pháp random forest với tập dữ liệu:

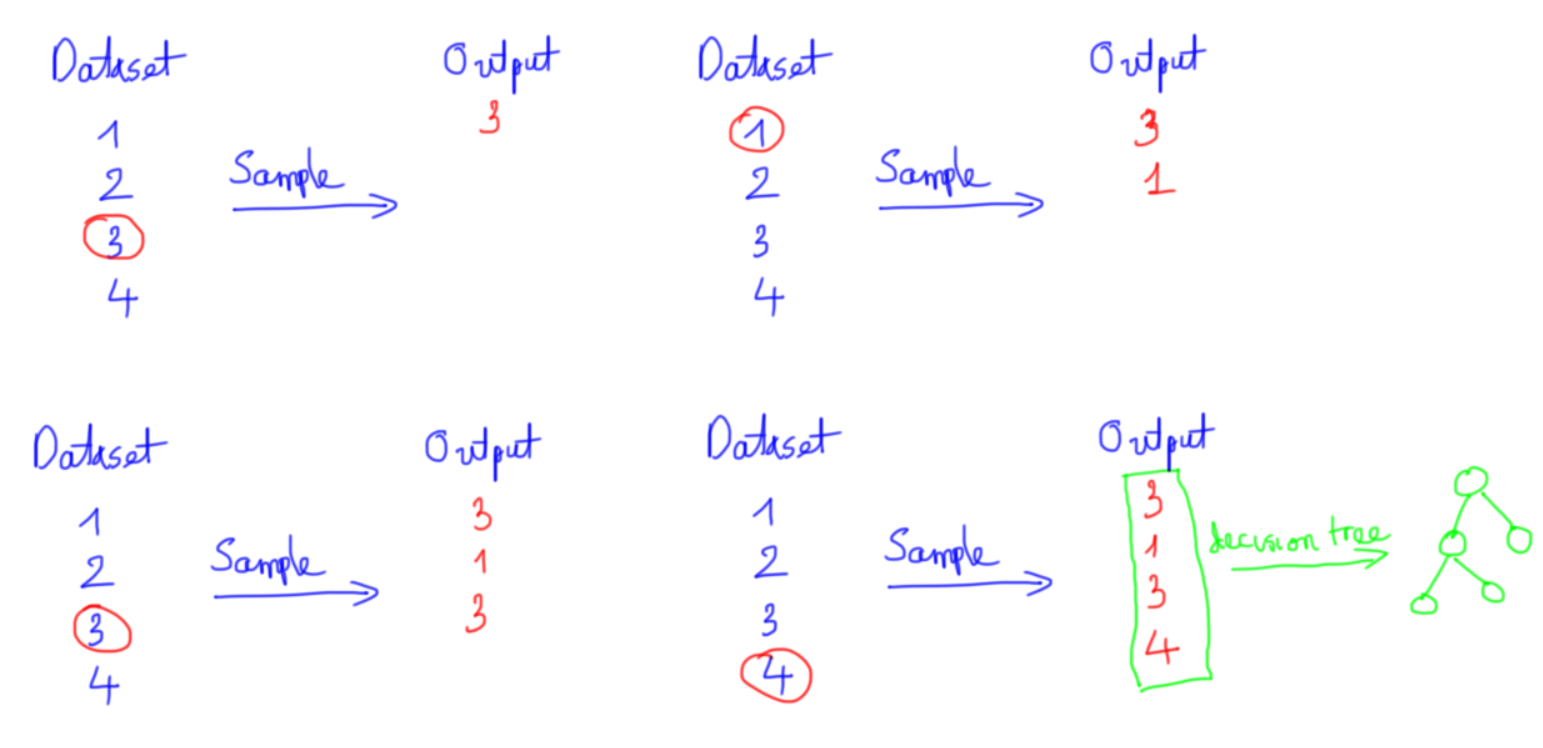
với i = 1

Để xác định được tính quan trọng của đối tượng thứ i sau khi lấy mẫu, các giá trị của mẫu i được hoán vị trong tập mẫu và các lỗi dự báo được tính toán lại trong tập dữ liệu. Độ quan trọng của đối tượng được tính bằng điểm, các điểm được tính toán bằng cách lấy trung bình của độ chênh lệch giữa các lỗi dự báo trước và sau khi hoán vị. Các đối tượng có giá trị lớn được xếp quan trọng hơn các điểm có giá trị nhỏ.

1.3.2 Xây dựng thuật toán Random Forest

Giả sử bộ dữ liệu có n dữ liệu (sample) và mỗi dữ liệu có d thuộc tính (feature). Để xây dựng mỗi cây quyết định sẽ làm như sau:

1. Lấy ngẫu nhiên n dữ liệu từ bộ dữ liệu với kỹ thuật Bootstrapping, hay còn gọi là random sampling with replacement. Tức khi sample được 1 dữ liệu thì không bỏ dữ liệu đấy ra mà vẫn giữ lại trong tập dữ liệu ban đầu, rồi tiếp tục sample cho tới khi sample đủ n dữ liệu. Khi dùng kĩ thuật này thì tập n dữ liệu mới có thể có những dữ liệu bị trùng nhau.



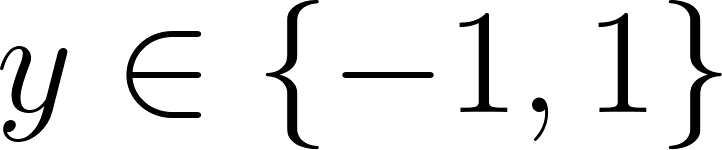
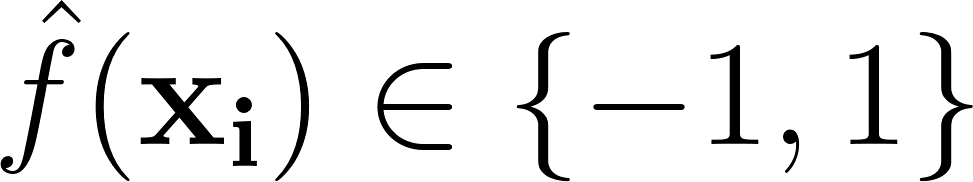
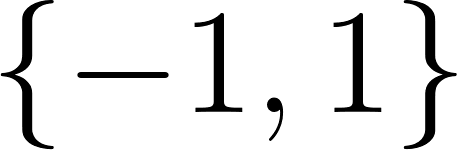
1. Sau khi sample được n dữ liệu từ bước 1 thì chọn ngẫu nhiên ở k thuộc tính (k < n). Giờ thu được bộ dữ liệu mới gồm n dữ liệu và mỗi dữ liệu có k thuộc tính.
2. Dùng thuật toán Decision Tree để xây dựng cây quyết định với bộ dữ liệu ở bước 2.

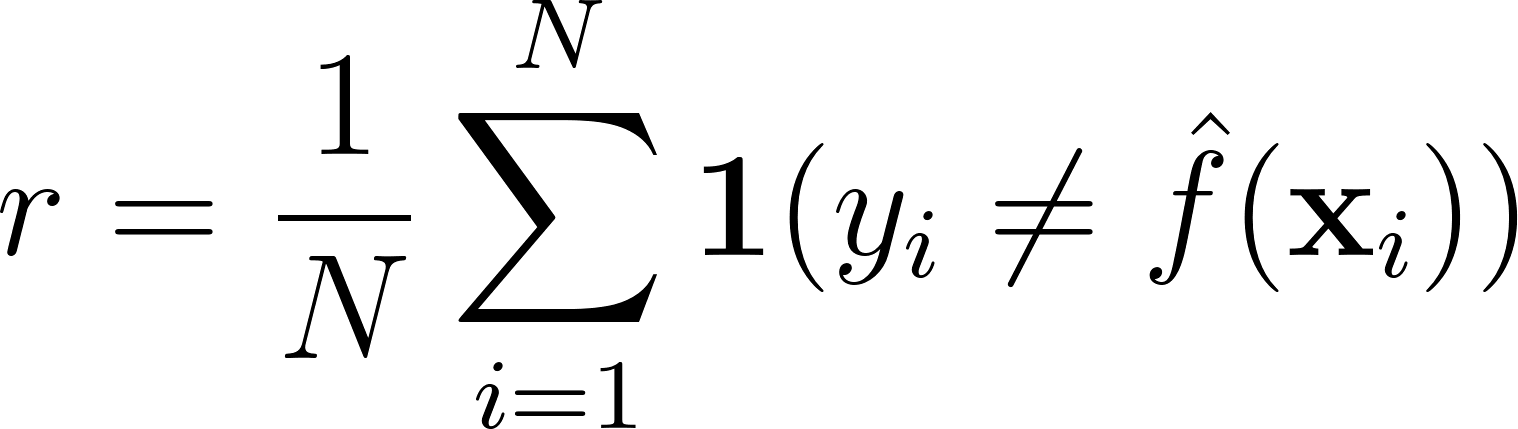
Do quá trình xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau.

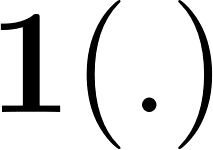
Thuật toán Random Forest sẽ bao gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau. Sau đó kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ các cây quyết định.

Khi dùng thuật toán Random Forest, để ý các thuộc tính như: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây. Ngoài ra, vẫn có các thuộc tính của thuật toán Decision Tree để xây dựng cây như độ sâu tối đa, số phần tử tối thiểu trong 1 node để có thể tách.

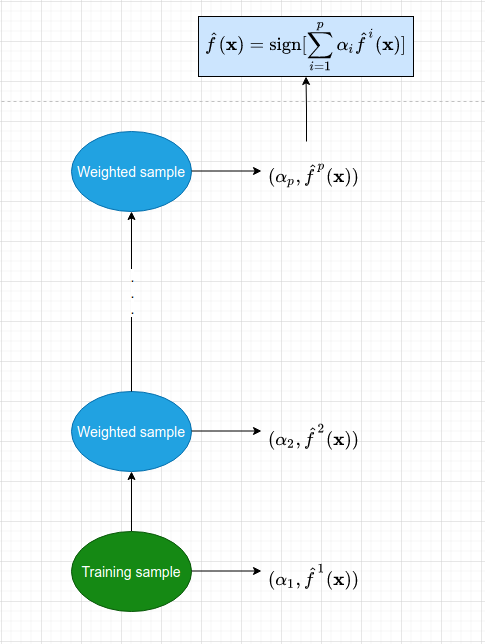
1.4. AdaBoosting

Giả định rằng bài toán phân loại nhị phân với biến mục tiêu gồm hai nhãn [](https://www.codecogs.com/eqnedit.php?latex=y%20%5Cin%20%5C%7B-1%2C%201%5C%7D#0). Giả định theo phương pháp tăng cường thì hàm dự báo đối với một biến đầu vào  là . Đồng thời biến mục tiêu [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7By%7D#0) nhận một trong hai giá trị . Khi đó sai số trên tập huấn luyện là:



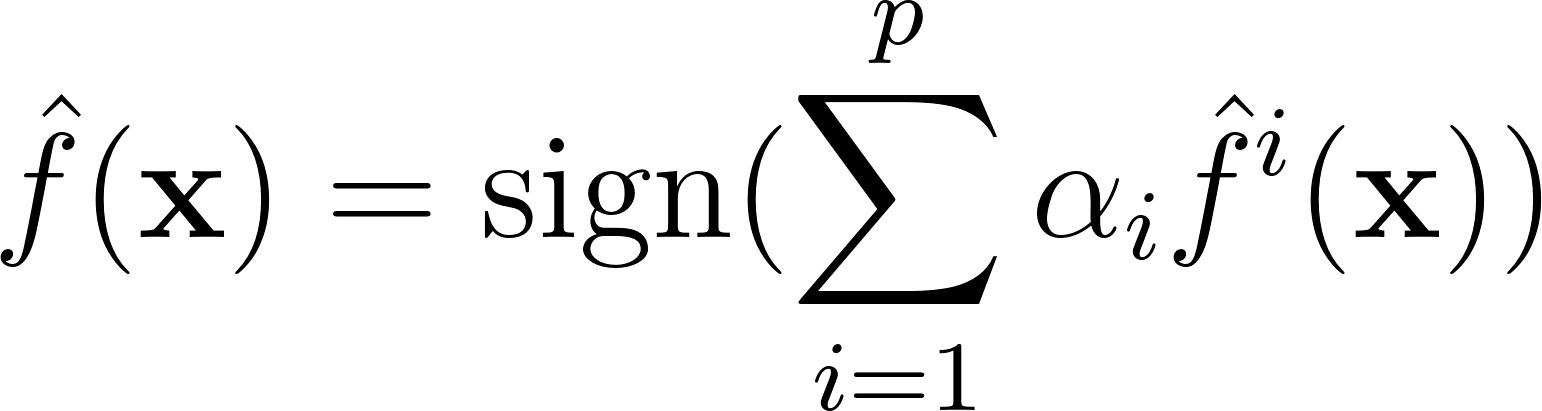
Trong đó hàm  là một hàm logic nhận giá trị 1 nếu như điều kiện bên trong hàm trả về là đúng, trái lại thì nhận giá trị 0.

Một mô hình phân loại yếu (weak classifier) có tỷ lệ dự báo sai lớn và giả định nó chỉ tốt hơn so với phân loại ngẫu nhiên một chút. Mục tiêu của phương pháp tăng cường là áp dụng liên tiếp các mô hình phân loại yếu để điều chỉnh lại trọng số cho các quan sát, qua đó ở mô hình sau sẽ ưu tiên phân loại đúng những quan sát đã phân loại sai từ mô hình trước đó. Kết thúc ta thu được một mô hình dự báo được kết hợp từ các mô hình phân loại yếu trong chuỗi. Mô hình kết hợp này thường có hiệu suất cao.

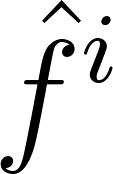


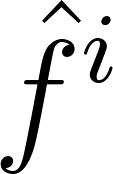
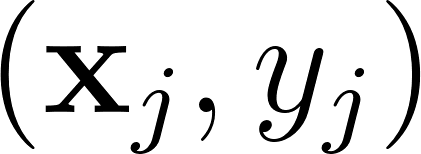
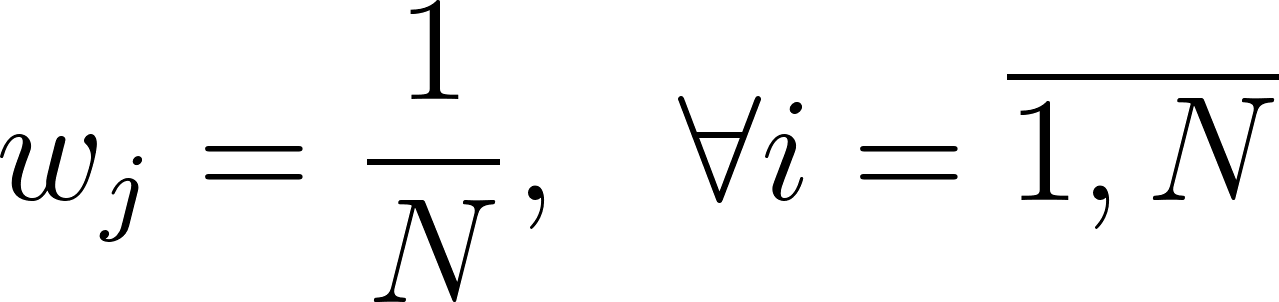
**Hình 1:** Sơ đồ của mô hình AdaBoosting. Mỗi một mô hình con được huấn luyện từ bộ dữ liệu được đánh trọng số theo tính toán từ mô hình tiền nhiệm. Dữ liệu có trọng số sau đó được đưa vào huấn luyện mô hình tiếp theo. Đồng thời ta cũng tính ra một trọng số quyết định  thể hiện vai trò của mỗi mô hình ở từng bước huấn luyện. Cứ tiếp tục như vậy cho tới khi số lượng mô hình đạt ngưỡng hoặc tập huấn luyện hoàn toàn được phân loại đúng thì dừng quá trình.

Kết quả dự báo từ mô hình cuối cùng là một kết hợp từ những mô hình với trọng số :

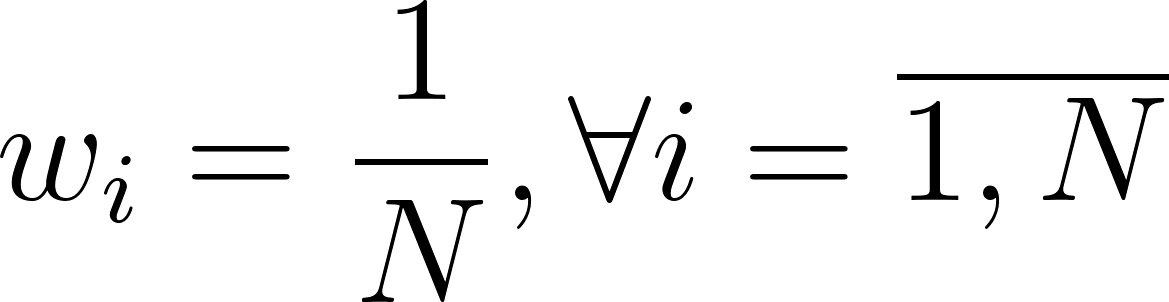
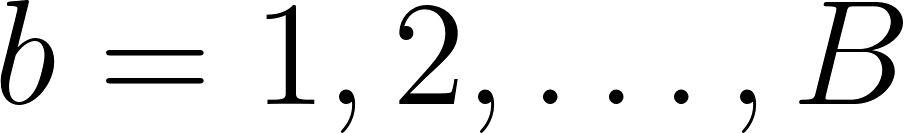
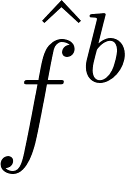
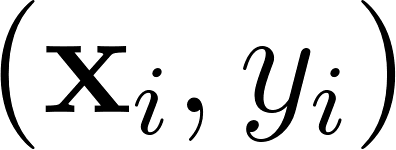
[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%20%3D%20%5Ctext%7Bsign%7D%20(%5Csum_%7Bi%3D1%7D%5E%7Bp%7D%20%5Calpha_i%20%5Chat%7Bf%7D%5E%7Bi%7D(%5Cmathbf%7Bx%7D))#0)

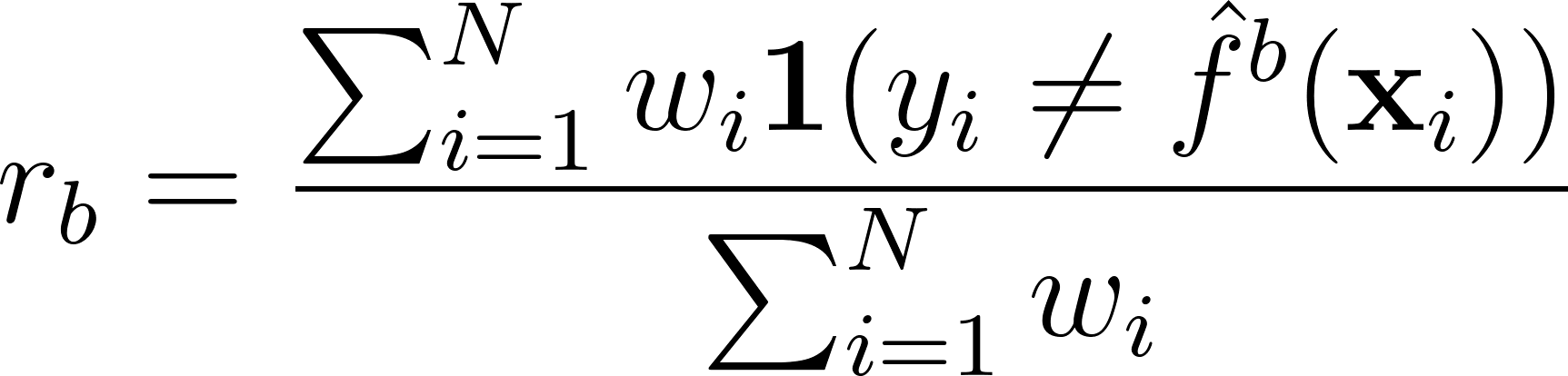
Trong phương trình trên hàm  là hàm nhận giá trị 1 nếu dấu của x là dương và nhận giá trị -1 nếu ngược lại.

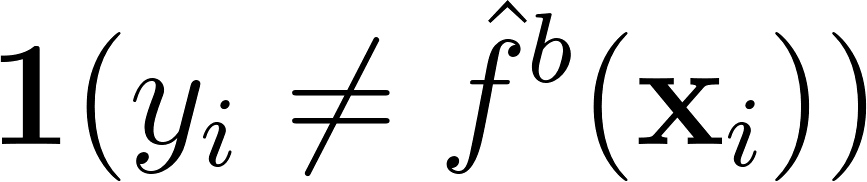
Các hệ số  được tính từ và phương pháp tăng cường, chúng được sử dụng để đánh trọng số mức độ đóng góp từ mỗi một mô hình con [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D%5E%7Bi%7D#0) trong chuỗi nhằm phân bổ vai trò quyết định trên từng mô hình khác nhau tùy thuộc vào mức độ chính xác của chúng. Điều này được phân tích kĩ hơn bên dưới.

Khi huấn luyện một mô hình con  thì chúng ta áp dụng một trọng số [](https://www.codecogs.com/eqnedit.php?latex=w_j#0) lên từng quan sát [](https://www.codecogs.com/eqnedit.php?latex=(%5Cmathbf%7Bx%7D_j%2C%20y_j)#0) sao cho đối với những quan sát bị dự báo sai thì trọng số của nó sẽ lớn hơn. Như vậy ở mô hình tiếp theo sẽ ưu tiên dự báo đúng những quan sát này hơn so với những quan sát đã được dự báo đúng. Ở thời điểm khởi đầu thì chúng ta gán [](https://www.codecogs.com/eqnedit.php?latex=w_j%20%3D%20%5Cfrac%7B1%7D%7BN%7D%2C%20~~%20%5Cforall%20i%20%3D%20%5Coverline%7B1%2C%20N%7D#0).

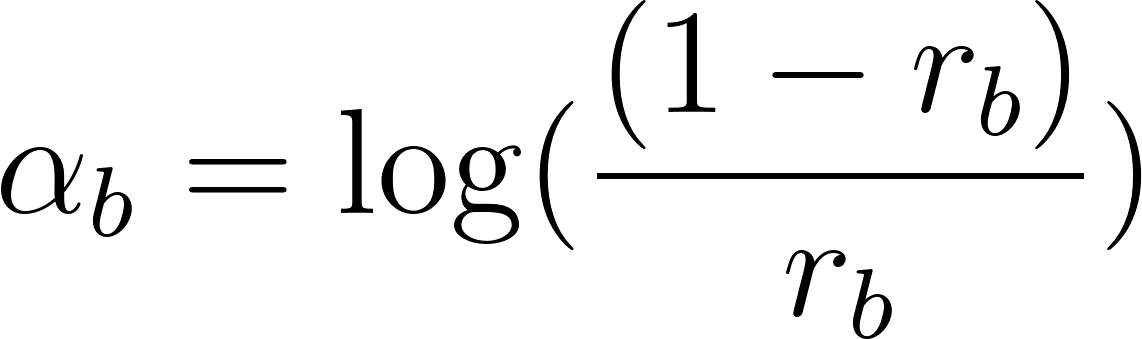
1.4.1 Các bước của thuật toán AdaBoosting

1. Khởi tạo trọng số quan sát [](https://www.codecogs.com/eqnedit.php?latex=w_i%20%3D%20%5Cfrac%7B1%7D%7BN%7D%2C%20%5Cforall%20i%20%3D%20%5Coverline%7B1%2C%20N%7D#0).
2. Lặp lại quá trình huấn luyện chuỗi mô hình ở mỗi bước b, [](https://www.codecogs.com/eqnedit.php?latex=b%20%3D%201%2C2%2C%20%5Cdots%2C%20B#0) gồm các bước con:
3. Khớp mô hình [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D%5E%7Bb%7D#0) cho tập huấn luyện sử dụng trọng số [](https://www.codecogs.com/eqnedit.php?latex=w_i#0) cho mỗi quan sát [](https://www.codecogs.com/eqnedit.php?latex=(%5Cmathbf%7Bx%7D_i%2C%20y_i)#0).
4. Tính sai số huấn luyện:

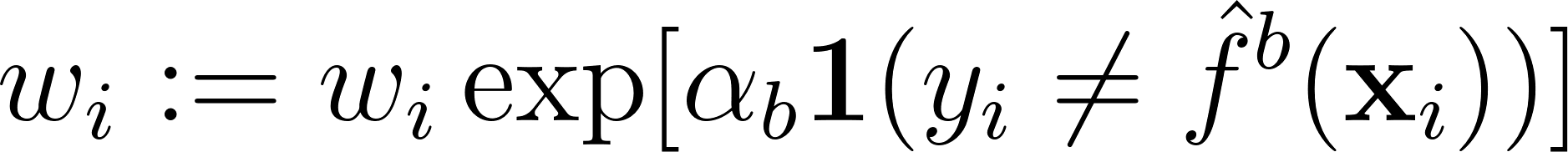
[](https://www.codecogs.com/eqnedit.php?latex=r_b%20%3D%20%5Cfrac%7B%5Csum_%7Bi%3D1%7D%5E%7BN%7D%20w_i%20%5Cmathbf%7B1%7D(y_i%20%5Cneq%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D_i))%7D%7B%5Csum_%7Bi%3D1%7D%5E%7BN%7D%20w_i%7D#0)

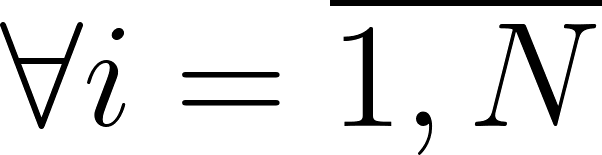
Ở đây [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7B1%7D(y_i%20%5Cneq%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D_i))#0) chính là những quan sát bị dự báo sai ở mô hình thứ b. Giá trị [](https://www.codecogs.com/eqnedit.php?latex=r_b%20%5Cin%20%5B0%2C%201%5D#0).

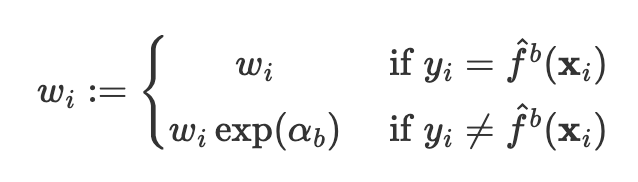
1. Tính trọng số quyết định cho từng mô hình:

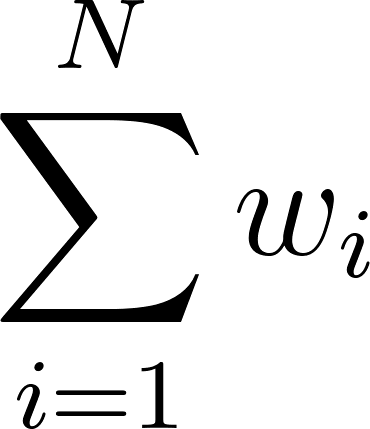
[](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%3D%20%5Clog(%5Cfrac%7B(1-r_b)%7D%7Br_b%7D)#0)

1. Cập nhật trọng số cho từng quan sát:

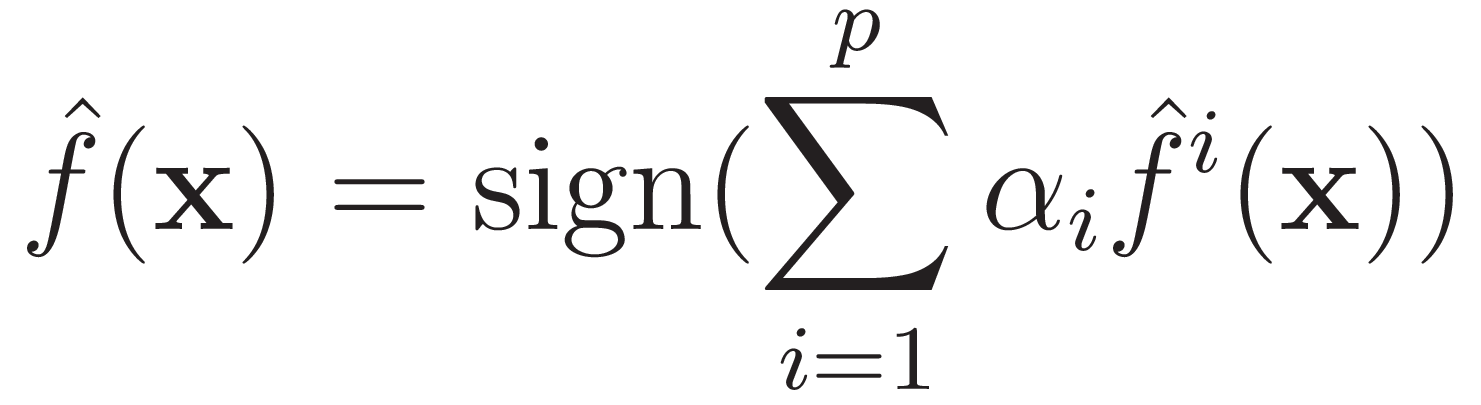
[](https://www.codecogs.com/eqnedit.php?latex=%20w_i%20%3A%3D%20w_i%20%5Cexp%5B%5Calpha_b%20%5Cmathbf%7B1%7D(y_i%20%5Cneq%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D_i))%5D%20#0)

với [](https://www.codecogs.com/eqnedit.php?latex=%5Cforall%20i%20%3D%20%5Coverline%7B1%2C%20N%7D#0). Như vậy ta có thể nhận thấy rằng:



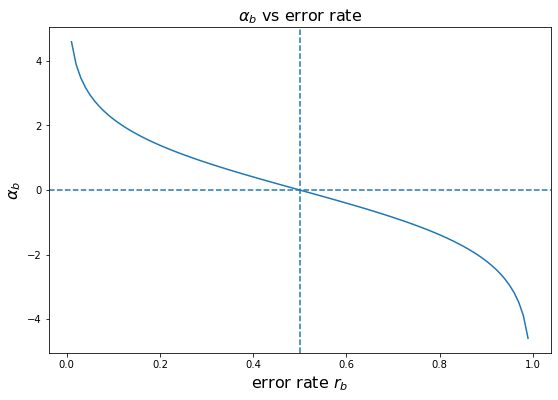
Sau khi tính xong các trọng số [](https://www.codecogs.com/eqnedit.php?latex=w_i#0) thì giá trị của chúng sẽ được chuẩn hoá bằng cách chia cho tổng [](https://www.codecogs.com/eqnedit.php?latex=%5Csum_%7Bi%3D1%7D%5E%7BN%7D%20w_i#0).

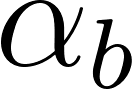
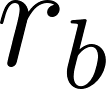
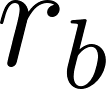
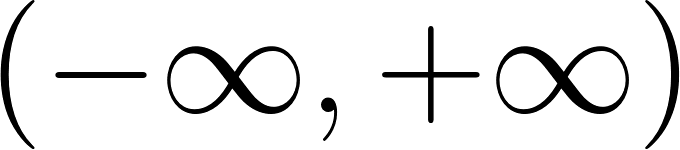
1. Cập nhật dự báo cuối cùng:

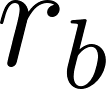
[](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%20%3D%20%5Ctext%7Bsign%7D%20(%5Csum_%7Bi%3D1%7D%5E%7Bp%7D%20%5Calpha_i%20%5Chat%7Bf%7D%5E%7Bi%7D(%5Cmathbf%7Bx%7D))%20%5Ctag%7B1%7D#0)

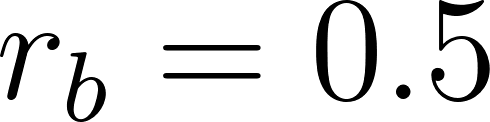
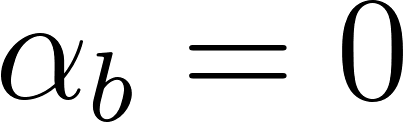
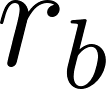
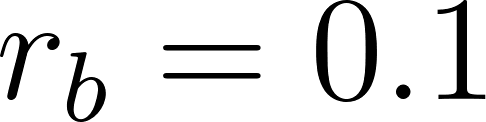
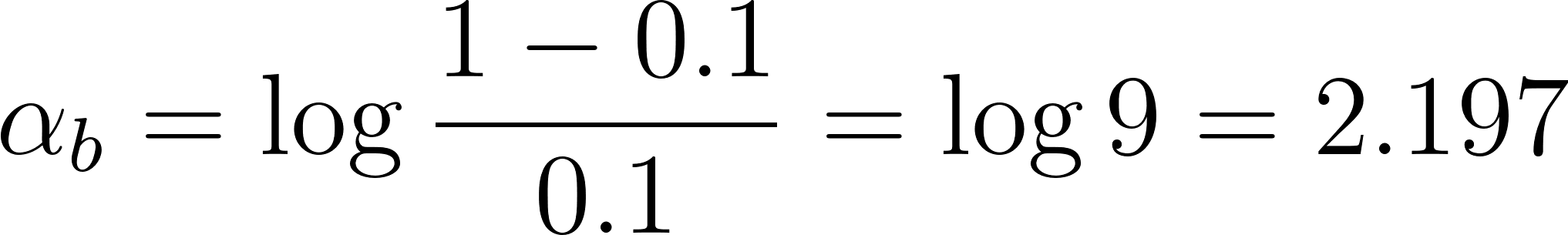
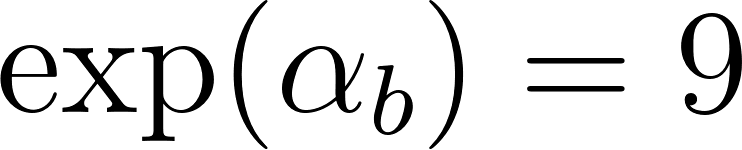
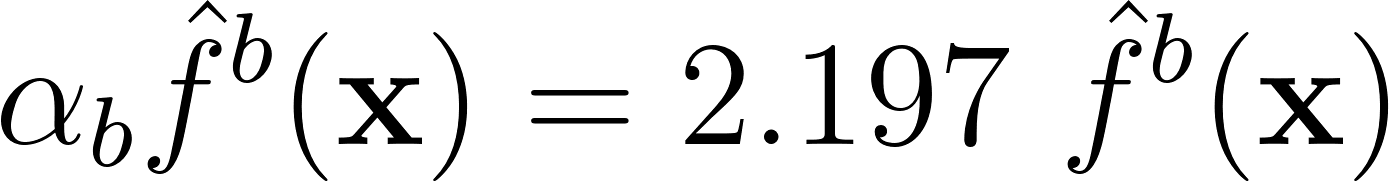
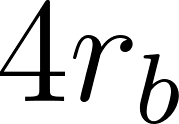
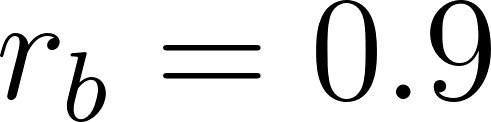
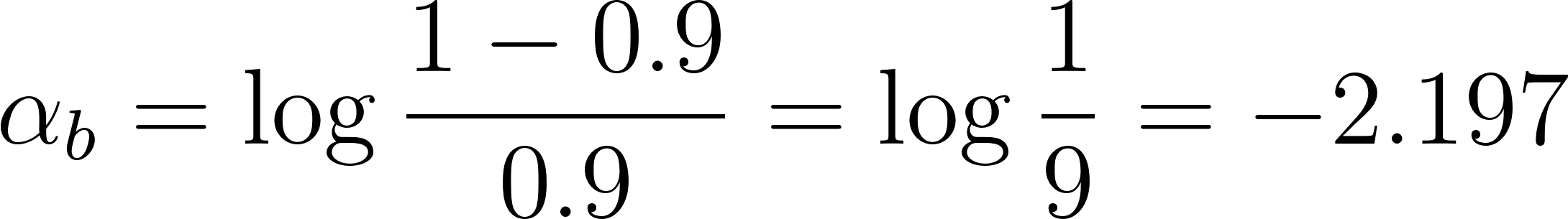
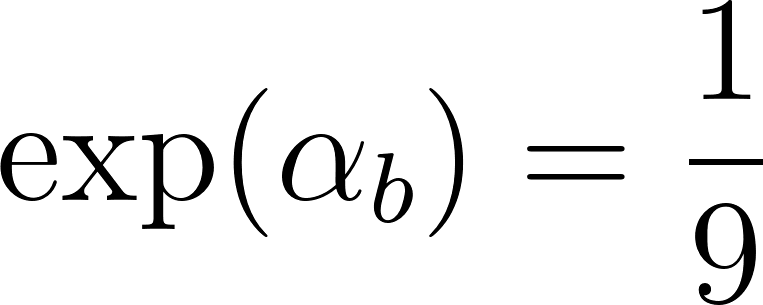
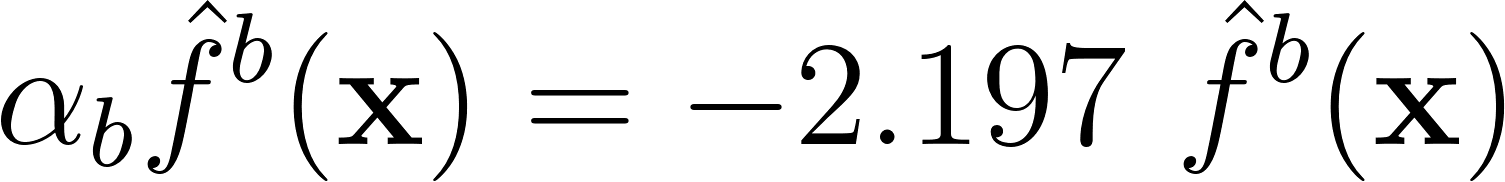
Trọng số [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_i#0) được tính ở bước thứ 2 thể hiện vai trò quan trọng trong việc ra quyết định của mô hình thứ i. Giá trị này được tính theo một hàm nghịch biến với sai số của mô hình. Chúng ta cùng phân tích hàm này bên dưới:

|  |
| --- |
| import numpy as np import matplotlib.pyplot as plt  plt.figure(figsize = (9, 6)) r = np.linspace(0.01, 0.99, 100)  def alpha(z):  return np.log((1-z)/z)  y = alpha(r) plt.plot(r, y) plt.xlabel(r'error rate $r\_b$', fontsize=16) plt.ylabel(r'$\alpha\_b$', fontsize=16) plt.axvline(0.5, linestyle='--') plt.axhline(0, linestyle='--') plt.title(r'$\alpha\_b$ vs error rate', fontsize=16) plt.show() |



**Hình 2:** Giá trị của trọng số quyết định [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b#0) theo sai số [](https://www.codecogs.com/eqnedit.php?latex=r_b#0). Đây là một hàm nghịch biến theo [](https://www.codecogs.com/eqnedit.php?latex=r_b#0) và có giá trị từ [](https://www.codecogs.com/eqnedit.php?latex=(-%5Cinfty%2C%20%2B%20%5Cinfty)#0)

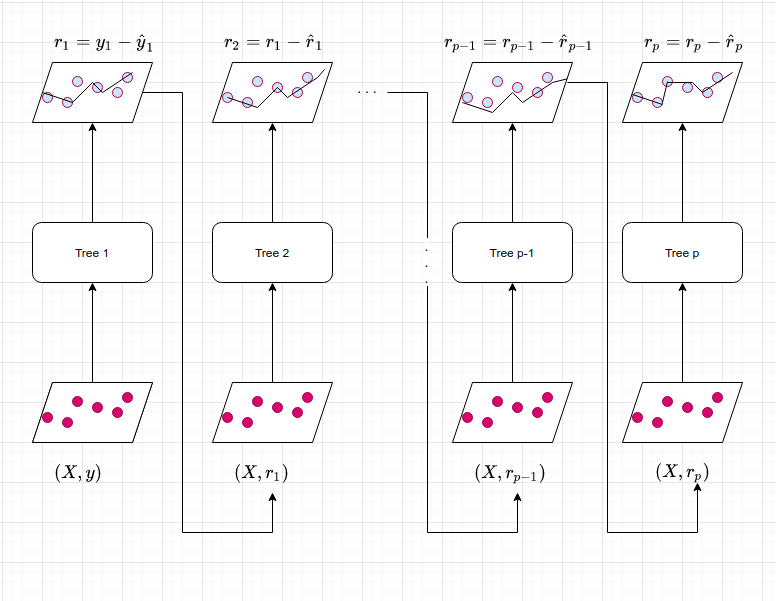
Bên dưới ta sẽ xét 3 trường hợp đối với sai số dự báo [](https://www.codecogs.com/eqnedit.php?latex=r_b#0):

* Khi [](https://www.codecogs.com/eqnedit.php?latex=r_b%20%3D%200.5#0) tương ứng với kết quả từ một mô hình dự báo ngẫu nhiên. Trường hợp này có [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%3D%200#0). Khi đó mô hình không có đóng góp gì vào hàm dự báo được thể hiện ở công thức (1). Điều này là hợp lý vì một giá trị dự báo ngẫu nhiên thì không có ích cho việc phân loại. Đồng thời trọng số sau cập nhật [](https://www.codecogs.com/eqnedit.php?latex=w_i%20%5Cexp(%5Calpha_b)%20%3D%20w_i#0), tức là vai trò của các quan sát được giữ cố định.
* Khi [](https://www.codecogs.com/eqnedit.php?latex=r_b#0) tiến dần tới 0, chẳng hạn [](https://www.codecogs.com/eqnedit.php?latex=r_b%20%3D%200.1#0), tương ứng với mô hình dự báo có tỷ lệ sai số thấp và đây là một mô hình khá mạnh. Khi đó [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%3D%20%5Clog%20%5Cfrac%7B1-0.1%7D%7B0.1%7D%20%3D%20%5Clog%7B9%7D%20%3D%202.197#0) và [](https://www.codecogs.com/eqnedit.php?latex=%5Cexp(%5Calpha_b)%20%3D%209#0). Như vậy đối với những quan sát bị dự báo sai thì trọng số của nó được gấp lên 9 lần, điều này giúp cho những mô hình sau sẽ điều chỉnh lại cây quyết định sao cho tập trung vào dự báo đúng những quan sát này. Đồng thời [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)%20%3D%202.197~%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)#0) cho thấy các dự báo từ mô hình này được đánh giá rất cao và góp phần gia tăng điểm số dự báo cuối cùng theo như công thức (1).
* Khi [](https://www.codecogs.com/eqnedit.php?latex=4r_b#0) tiến dần tới 1, chẳng hạn [](https://www.codecogs.com/eqnedit.php?latex=r_b%20%3D%200.9#0) cho thấy đây là một mô hình rất yếu vì có tỷ lệ sai số dự báo cao. Khi đó [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%3D%20%5Clog%20%5Cfrac%7B1-0.9%7D%7B0.9%7D%20%3D%20%5Clog%20%5Cfrac%7B1%7D%7B9%7D%20%3D%20-2.197#0) là một giá trị âm tương đối nhỏ và [](https://www.codecogs.com/eqnedit.php?latex=%5Cexp(%5Calpha_b)%20%3D%20%5Cfrac%7B1%7D%7B9%7D#0) là một giá trị gần 0. Như vậy trọng số [](https://www.codecogs.com/eqnedit.php?latex=w_i%20%5Cexp(%5Calpha_b)#0) sẽ bị giảm gấp 9 lần so với [](https://www.codecogs.com/eqnedit.php?latex=w_i#0). Lưu ý rằng trong trường hợp này mô hình đang dự báo hầu hết là sai nên nếu mô hình dự báo sai thì dường như những quan sát đó lại dễ được dự báo đúng và ít quan trọng. Điều này cũng giống như một người dự báo sai tới 90% thì khả năng ta lấy kết quả ngược lại của anh ta sẽ được mô hình dự báo đúng 90% và những trường hợp anh ta dự báo sai thường dễ dàng được phân loại đúng nhờ làm ngược lại. Do đó ta cần giảm trọng số [](https://www.codecogs.com/eqnedit.php?latex=w_i#0) cho những quan sát mà mô hình dự báo sai, trong trường hợp này là giảm đi 9 lần. Đồng thời đóng góp từ kết quả dự báo vào mô hình là [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha_b%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)%20%3D%20-2.197~%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)#0) cho thấy kết quả từ mô hình này sẽ được cập nhật ngược chiều vào điểm số cuối cùng. Điều này cũng giống như chúng ta làm ngược lại gợi ý của một người hay phán đoán sai để thu được phán đoán đúng.

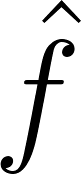
Quá trình tăng cường mô hình sẽ tiếp tục như vậy cho đến khi mô hình đạt số lượng tối đa hoặc toàn bộ các quan sát trên tập kiểm tra được phân loại đúng. Một lưu ý đó là các mô hình cây quyết định con trong phương pháp tăng cường thường có độ sâu thấp, thông thường chỉ gồm 1 node gốc với hai node lá, trường hợp cây quyết định chỉ gồm một node gốc được gọi là mô hình gốc cây (stump). Sở dĩ chúng ta không cần yêu cầu các cây quyết định phải quá phức tạp là để ngăn ngừa hiện tượng quá khớp có thể xảy ra và đồng thời tăng khả năng giải thích cho mô hình.

1.5 Gradient Boosting

Phương pháp Gradient Boosting cũng có ý tưởng tương tự như AdaBoosting đó là huấn luyện liên tiếp các mô hình yếu. Nhưng chúng ta không sử dụng sai số của mô hình để tính toán trọng số cho dữ liệu huấn luyện mà sử dụng phần dư. Xuất phát từ mô hình hiện tại, chúng ta cố gắng xây dựng một cây quyết định cố gắng khớp phần dư từ mô hình liền trước. Điểm đặc biệt của mô hình này đó là thay vì chúng ta cố gắng khớp giá trị biến mục tiêu là y thì chúng ta sẽ tìm cách khớp giá trị sai số của mô hình trước đó. Sau đó chúng ta sẽ đưa thêm mô hình huấn luyện vào hàm dự báo để cập nhật dần dần phần dư. Mỗi một cây quyết định trong chuỗi mô hình có kích thước rất nhỏ với chỉ một vài nodes quyết định được xác định bởi tham số độ sâu d trong mô hình. Hình bên dưới sẽ minh hoạ cụ thể hơn quá trình này:



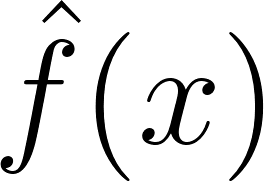
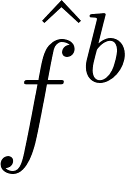
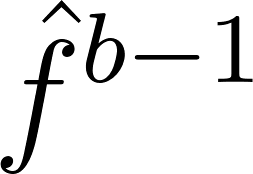
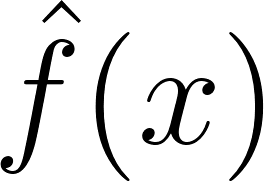
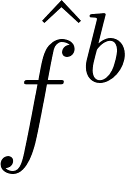
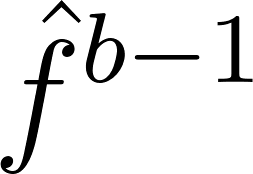
**Hình 3:** Phương pháp huấn luyện mô hình theo Gradient Boosting. Các mô hình cây quyết định được sắp xếp theo chuỗi. Mỗi cây quyết định sẽ được thành lập phụ thuộc vào kết quả dự báo của cây quyết định liền trước. Tại một cây quyết định mô hình sẽ tìm cách khớp phần dư từ cây quyết định trước đó.

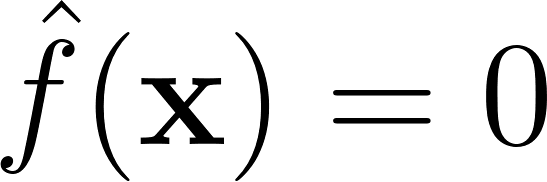
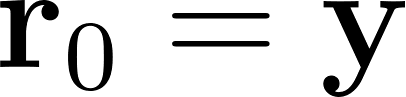
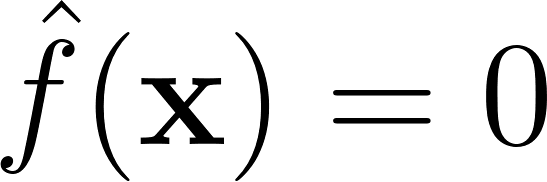
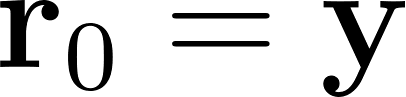
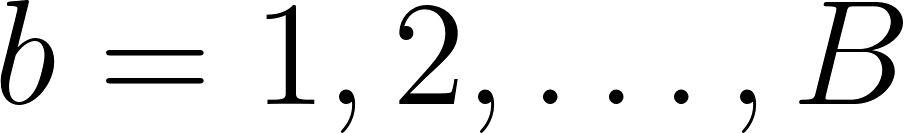
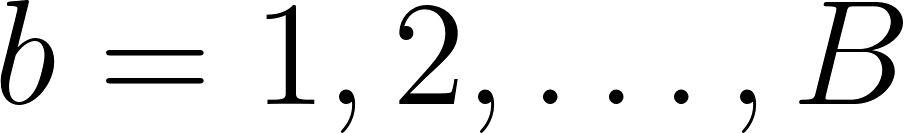
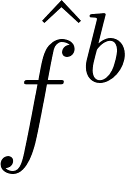
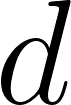
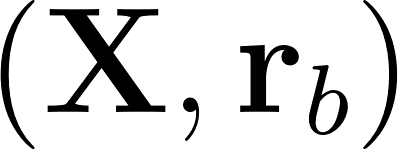
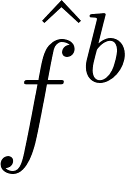
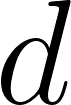
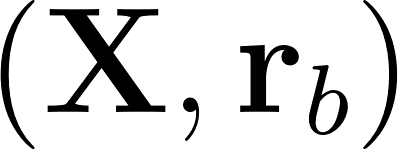
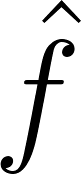
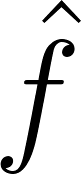
Bằng cách khớp trên những cây quyết định có kích thước rất nhỏ trên những phần dư, chúng ta sẽ từ từ cải hiện hàm dự báo  trong vùng mà nó không được dự báo tốt. Tham số co (shrinkage parameter) [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda#0) cũng giống như hệ số học tập (learning rate) có tác dụng làm chậm quá trình tiếp cận tới mô hình tốt hơn, điều này cho phép tạo ra nhiều các cây quyết định với hình dạng khác nhau để khớp phần dư. Theo phương pháp tiếp cận chậm bằng cách lấp đầy từ từ phần dư, mô hình thường có hiệu suất cao và vượt trội so với phương pháp \_bỏtúi\_ khi xây dựng một cây quyết định sâu ngay từ đầu. Lưu ý rằng, theo phương pháp tăng cường thì sự thành lập của mỗi cây quyết định phụ thuộc mạnh mẽ vào những cây quyết định đã được phát triển trước đó.

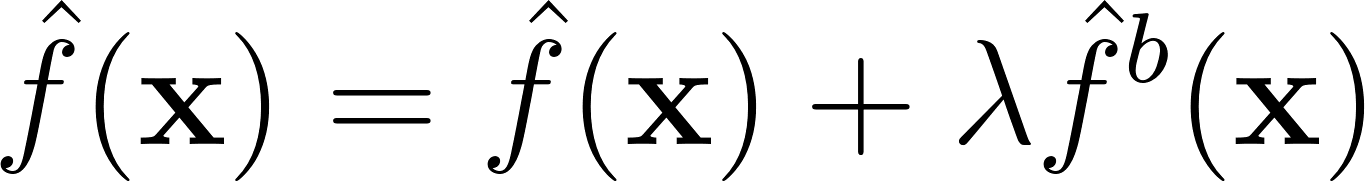
Ở hình trên chúng ta vừa mô tả quá trình tăng cường đối với một cây hồi quy (regression tree) áp dụng trên bài toán dự báo. Các tiếp cận đối với bài toán phân loại tương tự như phương pháp AdaBoosting. Như vậy trong phương pháp tăng cường sẽ có ba tham số hiệu chỉnh chính:

1. Số lượng cây [](https://www.codecogs.com/eqnedit.php?latex=B#0). Không giống như phương pháp rừng cây, phương pháp tăng cường có thể gặp hiện tượng quá khớp nếu [](https://www.codecogs.com/eqnedit.php?latex=B#0) lớn, mặc dù hiện tượng quá khớp này có xu hướng xảy ra từ từ nếu chúng xuất hiện. Để lựa chọn ra số lượng cây [](https://www.codecogs.com/eqnedit.php?latex=B#0) phù hợp chúng ta có thể sử dụng đánh giá chéo (cross validation).
2. Hệ số co [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda#0) là một số dương nhỏ. Hệ số này cũng gần giống như learning rate có tác dụng kiểm soát tỷ lệ mà phương pháp tăng cường cập nhật số dư. Các giá trị của hệ số cos thường là 0.01 hoặc 0.001, tuỳ thuộc vào từng bài toán và từng bộ dữ liệu cụ thể. Thông thường khi [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda#0) rất nhỏ có thể cần sử dụng một giá trị rất lớn của [](https://www.codecogs.com/eqnedit.php?latex=B#0) để đạt được hiệu suất tốt.
3. Độ sâu d của cây quyết định đại diện cho số lần phân chia tối đa trong mỗi cây quyết định. Thường thì trong phương pháp tăng cường thì chúng ta không cần yêu cầu d quá lớn. Điều này nhằm kiểm soát mức độ phức tạp của mô hình và tránh hiện tượng quá khớp. Trường hợp phổ biến là [](https://www.codecogs.com/eqnedit.php?latex=d%20%3D%201#0) cho thấy mô hình huấn luyện theo phương pháp tăng cường (gọi là mô hình tăng cường) đã có thể hoạt động tốt, khi đó mỗi cây được gọi là gốc cây (stump) chỉ gồm một node phân chia. Trong trường hợp này, mô hình tăng cường tìm cách khớp một xác suất cộng dồn mà mỗi một phần tử là một mô hình gốc cây chỉ gồm một câu hỏi.

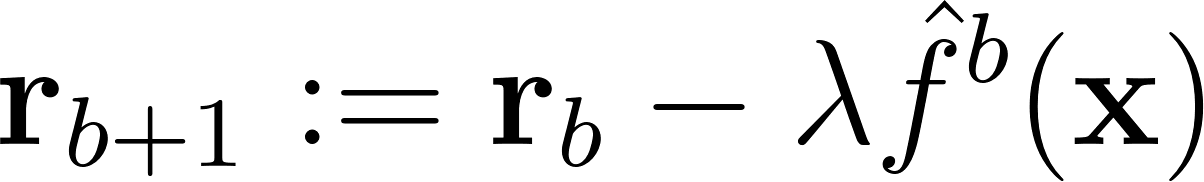
## Các bước của thuật toán Gradient Boosting

Giả định [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D(x)#0) là hàm dự báo từ phương pháp tăng cường được áp dụng trên một tác vụ dự báo với ma trận đầu vào [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7BX%7D#0) và biến mục tiêu là véc tơ [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7By%7D#0). Tại mô hình thứ [](https://www.codecogs.com/eqnedit.php?latex=b#0) trong chuỗi mô hình dự báo, kí hiệu là [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D%5E%7Bb%7D#0), ta tìm cách khớp một giá trị phần dư [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7Br%7D_i#0) từ cây quyết định tiền nhiệm [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D%5E%7Bb-1%7D#0). Các bước trong quá trình huấn luyện mô hình theo phương pháp tăng cường được tóm tắt như sau: là hàm dự báo từ phương pháp tăng cường được áp dụng trên một tác vụ dự báo với ma trận đầu vào  và biến mục tiêu là véc tơ . Tại mô hình thứ  trong chuỗi mô hình dự báo, kí hiệu là , ta tìm cách khớp một giá trị phần dư  từ cây quyết định tiền nhiệm . Các bước trong quá trình huấn luyện mô hình theo phương pháp tăng cường được tóm tắt như sau:

1. Ban đầu ta thiết lập hàm dự báo [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%20%3D%200#0) và số dư [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7Br%7D_0%20%3D%20%5Cmathbf%7By%7D#0) cho toàn bộ quan sát trong tập huấn luyện. và số dư  cho toàn bộ quan sát trong tập huấn luyện.
2. Lặp lại quá trình huấn luyện cây quyết định theo chuỗi tương ứng với [](https://www.codecogs.com/eqnedit.php?latex=b%20%3D%201%2C2%2C%20%5Cdots%2C%20B#0). Với một lượt huấn luyện gồm các bước con sau đây:. Với một lượt huấn luyện gồm các bước con sau đây:
3. Khớp một cây quyết định [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D%5E%7Bb%7D#0) có độ sâu là [](https://www.codecogs.com/eqnedit.php?latex=d#0) trên tập huấn luyện [](https://www.codecogs.com/eqnedit.php?latex=(%5Cmathbf%7BX%7D%2C%20%5Cmathbf%7Br%7D_b)#0). có độ sâu là  trên tập huấn luyện .
4. Cập nhật [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D#0) bằng cách cộng thêm vào giá trị dự báo của một cây quyết định, giá trị này được nhân với hệ số co [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda#0): bằng cách cộng thêm vào giá trị dự báo của một cây quyết định, giá trị này được nhân với hệ số co :

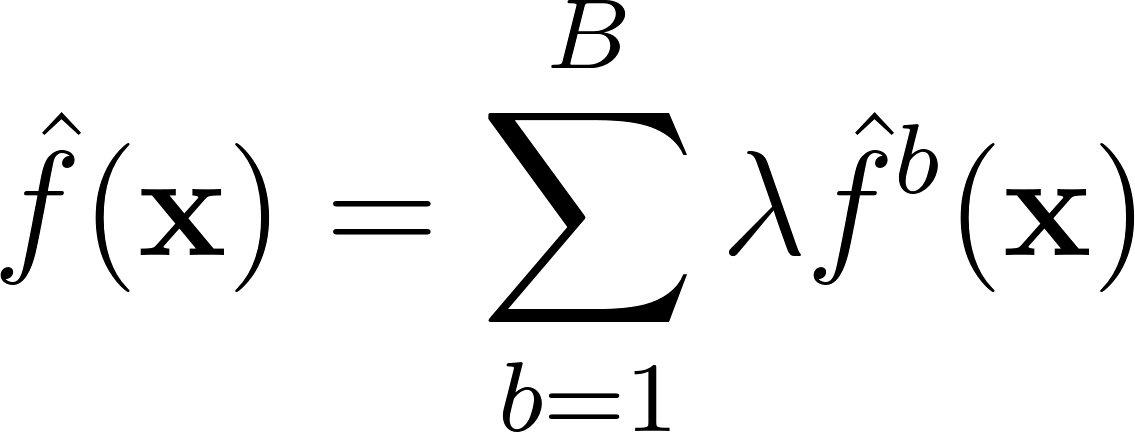
[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%20%3D%20%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%2B%5Clambda%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)#0)

1. Cập nhật phần dư cho mô hình:

[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathbf%7Br%7D_%7Bb%2B1%7D%20%3A%3D%20%5Cmathbf%7Br%7D_b%20-%20%5Clambda%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)#0)

Thuật toán sẽ dừng cập nhật khi số lượng cây quyết định đạt ngưỡng tối đa [](https://www.codecogs.com/eqnedit.php?latex=B#0) hoặc toàn bộ các quan sát trên tập huấn luyện được dự báo đúng. hoặc toàn bộ các quan sát trên tập huấn luyện được dự báo đúng.

3. Kết quả dự báo từ chuỗi mô hình sẽ là kết hợp của các mô hình con:

[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bf%7D(%5Cmathbf%7Bx%7D)%20%3D%20%5Csum_%7Bb%3D1%7D%5E%7BB%7D%20%5Clambda%20%5Chat%7Bf%7D%5E%7Bb%7D(%5Cmathbf%7Bx%7D)#0)

CHƯƠNG 2 – KHÁM PHÁ THUẬT TOÁN VÀ KỸ THUẬT TỐI ƯU TRONG MẠNG NEURAL

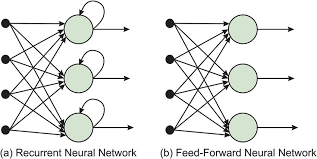
* 1. Feed Forward Neural Network

Trong bài viết này, chúng ta sẽ tìm hiểu về Mạng nơ-ron tiếp nối (feedforward Neural Networks- FFNN), còn được gọi là Mạng tiếp nối sâu hoặc Perceptron nhiều lớp (Deep feedforward Networks, Multi-layer Perceptrons - MLP). Chúng tạo thành nền tảng của nhiều Mạng thần kinh quan trọng đang được sử dụng trong thời gian gần đây, chẳng hạn như CNN (được sử dụng rộng rãi trong các ứng dụng thị giác máy tính), RNN (được sử dụng rộng rãi trong hiểu ngôn ngữ tự nhiên và học theo trình tự).

Feed Forward Neural Network (FFNN) là một loại mạng neural cơ bản và phổ biến trong học sâu. Đây là loại mạng neural đơn giản nhất, trong đó dữ liệu di chuyển theo một hướng từ đầu vào qua các lớp ẩn đến đầu ra mà không có chu kỳ (loop).

A diagram of a machine

Description automatically generated



* + 1. Cấu trúc của FFNN

2.1.1.1 Các tầng - Layers

FFNN bao gồm ba loại lớp chính:

* **Lớp đầu vào (Input Layer):** Nhận dữ liệu đầu vào. Số lượng nút trong lớp này bằng với số lượng tính năng trong dữ liệu đầu vào.
* **Lớp ẩn (Hidden Layers):** Các lớp giữa đầu vào và đầu ra. Chúng thực hiện các phép biến đổi và tính toán phức tạp. Một FFNN có thể có một hoặc nhiều lớp ẩn.
* **Lớp đầu ra (Output Layer):** Cung cấp đầu ra của mạng. Số lượng nút trong lớp này tùy thuộc vào bài toán cụ thể (ví dụ: một nút cho bài toán hồi quy, nhiều nút cho bài toán phân loại đa lớp).

A group of blue circles with black arrows

Description automatically generated

Số lượng layer trong một MLP được tính bằng số hidden layers + output layer. Như hình trên số lượng layers trong MLP: L = 3.

2.1.1.2 Nodes – Units

Một node hình tròn trong một layer được gọi là một unit. Đầu vào của các hidden layer được ký hiệu bởi z, đầu ra của mỗi unit thường được ký hiệu là a.

A diagram of a network

Description automatically generated

* Trong đó:
  + Đầu ra của unit thứ i trong layer thứ được ký hiệu
  + là ma trận trọng số thể thể hiện liên kết layer từ đến
  + b là độ lệch. Khi bỏ bias, phương trình giờ có dạng: w1\*x + w2∗y=0, sẽ luôn đi qua gốc tọa độ và nó không tổng quát hóa phương trình đường thẳng nên có thể không tìm được phương trình mong muốn

2.1.1.3 Activation functions

Các hàm kích hoạt có thể là các hàm phi tuyến khác nhau, như là hàm sigmoid hoặc hàm tanh. Điều quan trọng là các hàm kích hoạt phải là phi tuyến(non-linear), vì nếu không, nhiều layer hoặc một layer cũng sẽ không có sự khác biệt. Ví dụ, nếu hàm kích hoạt là một hàm tuyến tính (giả sử hàm f(s) = s), thì cả hai layer có thể được thay thế bằng một layer duy nhất với ma trận hệ số . Ở phía dưới, chúng mình sẽ giới thiệu:

* Sigmoid
* Tanh
* Relu
* Softmax

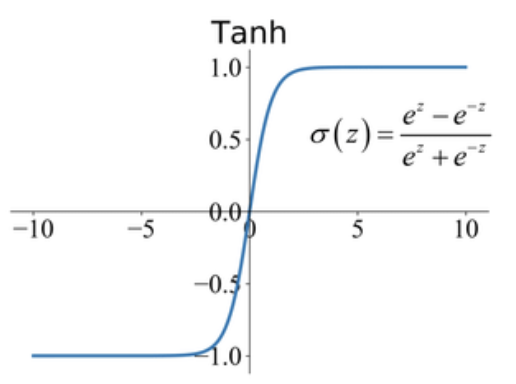
Hàm sigmoid được sử dụng cho hồi quy logistic hai lớp. Nó được sử dụng để phân loại xem một mẫu nhất định là Đúng hay Sai dựa trên xác suất được tính toán. Mẫu được phân loại là True nếu giá trị lớn hơn 0,5 và ngược lại. Nếu đầu vào lớn, hàm số sẽ cho đầu ra gần với 1. Với đầu vào nhỏ (rất âm), hàm số sẽ cho đầu ra gần với 0. Một vấn đề dễ nhận thấy là khi giá trị đầu vào rất lớn (âm hoặc dương), gradient của hàm số sẽ gần bằng 0. Điều này có nghĩa là các hệ số tương ứng với đơn vị đang xét sẽ gần như không được cập nhật (**vanishing gradient problem**).

A graph of a function

Description automatically generated

Khác với Sigmoid giá trị chạy trong khỏi từ (0,1) thì tanh chạy trong khỏi (-1, 1).

Tanh: tanh(z) =



ReLU (Rectified Linear Unit) được sử dụng rộng rãi gần đây vì tính đơn giản của nó. Công thức:

ReLU(z)=max(0,z)

* Với = z
* Với = 0

A graph of a function

Description automatically generated A graph of a function

Description automatically generated with medium confidence

Ở 2 hình trên so sánh sự hội tụ của SGD khi sử dụng hai activation function khác nhau: ReLU và tanh. Cải thiện tốc độ tính toán được cho là do ReLU có thể tính toán gần như ngay lập tức và gradient của nó cũng được tính toán nhanh chóng, với giá trị gradient bằng 1 nếu đầu vào lớn hơn 0 và bằng 0 nếu đầu vào nhỏ hơn 0. Hàm ReLU có nhiều biến thể khác như Noisy ReLU, Leaky ReLu, ELUs.

Tiếp theo, softmax activation function thường được sử dụng trong lớp đầu ra của các mạng neural để giải quyết các bài toán phân loại đa lớp (multiclass classification). Softmax chuyển đổi một vector các giá trị thành một vector các xác suất, trong đó tổng tất cả các xác suất là 1.

Cho 1 vector đầu vào:

Công thức Softmax:

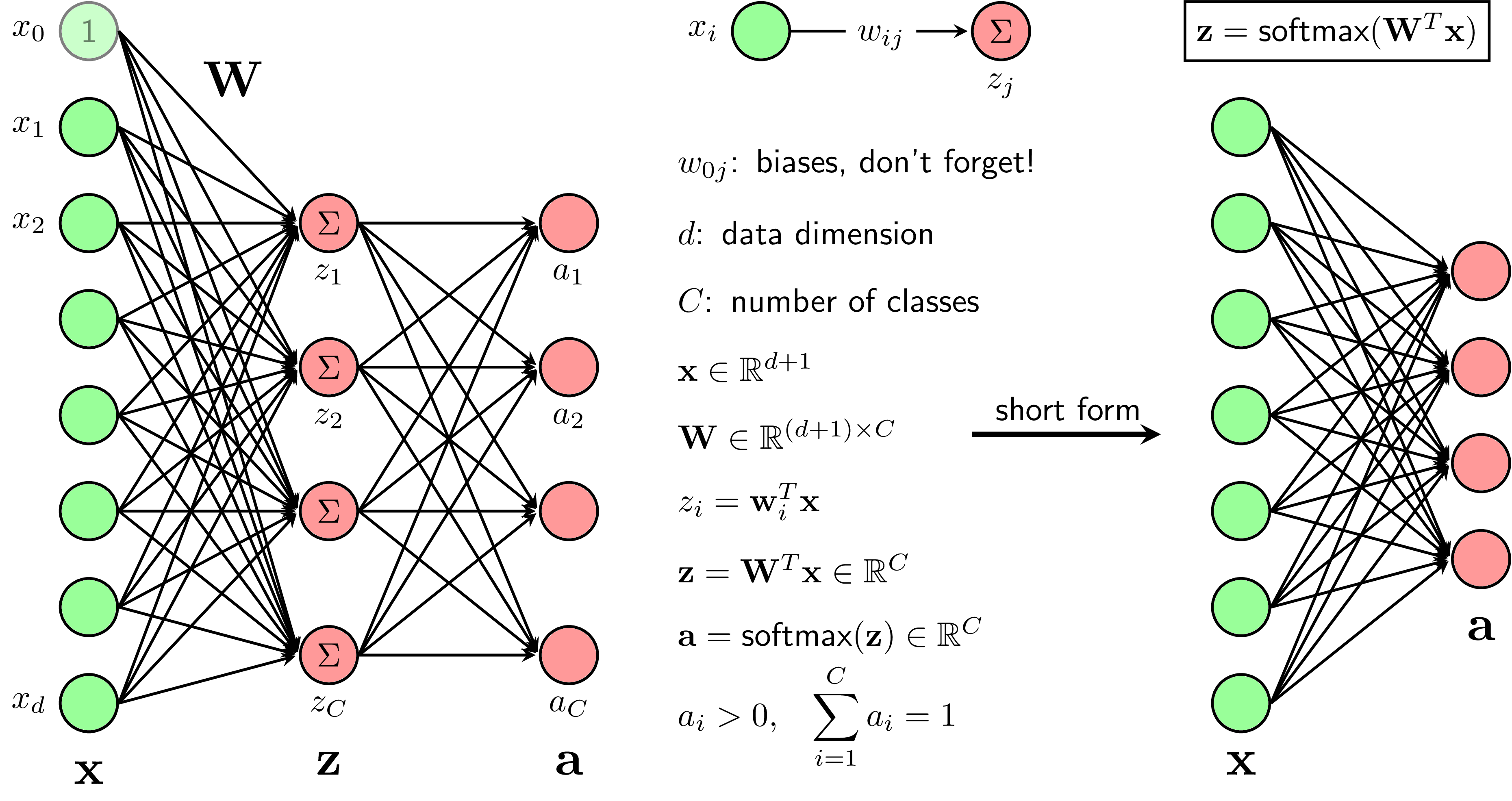
Trong đó :

* thể hiện xác suất để input đó rơi vào class i, với lớn hơn 0 và có tổng bằng 1
* , càng lớn thì xác suất dữ liệu rơi vào class i càng cao. có thể nhận giá trị âm hoặc dương.
* là một hàm mượt, có nghĩa là nó liên tục và khả vi vô hạn. Nghĩ là là 1 giá trị dương thì sẽ trở thành 1 hàm đồng biến(đơn điệu tăng) với tăng thì tăng.
* A graph of a function

  Description automatically generated

Ta có thể viết thành

* được hiểu là xác suất để một điểm dữ liệu x rơi vào class thứ i nếu biết tham số mô hình (ma trận trọng số) là W

Hình vẽ dưới đây thể hiện mạng Softmax Regression dưới dạng neural network: 

2.1.2 Quá trình hoạt động của FFNN

**Lan truyền tiến (Forward Propagation):** Dữ liệu đầu vào di chuyển qua các lớp của mạng, mỗi lớp áp dụng một phép biến đổi tuyến tính và sau đó là một hàm kích hoạt (activation function). Kết quả của mỗi lớp trở thành đầu vào cho lớp tiếp theo.

A math equations on a white background

Description automatically generated

* Trong đó:
* 𝑥 là vector đầu vào.
* 𝑊 là ma trận trọng số.
* 𝑏 là vector hệ số chặn (bias).
* 𝜎 là hàm kích hoạt.

**Hàm kích hoạt (Activation Function):** Áp dụng một hàm phi tuyến để giới thiệu tính phi tuyến vào mô hình, giúp mạng học được các mối quan hệ phi tuyến trong dữ liệu.

Các hàm kích hoạt phổ biến:

* Sigmoid: ​
* Tanh: tanh(z) =
* ReLU: ReLU(z)=max(0,z)

2.1.3 Hàm mất mát (Loss Function):

Hàm mất mát đo lường sự khác biệt giữa giá trị dự đoán của mạng và giá trị thực tế. Mục tiêu của việc huấn luyện là tối thiểu hóa hàm mất mát.

Một số hàm mất mát phổ biến:

* Mean Squared Error (MSE) cho bài toán hồi quy.
* A math equations and symbols

  Description automatically generated with medium confidence
* Với (X, Y) là cặp dữ liệu trong training
* Cross-Entropy Loss cho bài toán phân loại: Như vậy, Entropy của tín hiệu này có phân phối P (tính kì vọng dựa vào P) nhưng lại được mã hóa dựa vào phân phối Q.
* A black text on a white background

  Description automatically generated
* Tính chất của CrossEntropy
  + - Đạt giá trị minimu P == Q
    - Rất nhạy cảm với sự sai khác giữa và . Khi càng khác nhau thì cross entropy càng tăng. Vì vậy bài toán cần xây dựng output càng gần target càng tốt.
    - Công thức Binary cross entropy loss – BCE

A black text on a white background

Description automatically generated

Ví dụ: tính BCE, hình 1 với output và target gần nhau và hình 2 thì ngược lại.

A math equations with numbers

Description automatically generated

A math equations with numbers

Description automatically generated

2.1.4 Lan truyền ngược (Backpropagation):

Lan truyền ngược là quá trình điều chỉnh trọng số của mạng dựa trên lỗi của dự đoán. Nó bao gồm hai bước chính:

* **Lan truyền ngược gradient:** Tính gradient của hàm mất mát đối với từng trọng số bằng cách sử dụng quy tắc chuỗi. Gradient này cho biết mức độ ảnh hưởng của từng trọng số đến lỗi.
* **Cập nhật trọng số:** Điều chỉnh trọng số bằng cách sử dụng gradient đã tính toán. Quá trình này thường sử dụng một thuật toán tối ưu hóa như Gradient Descent.
* Quy tắc cập nhật trọng số:

A black and white math symbols

Description automatically generated with medium confidence

* Trong đó:
* η là tốc độ học (learning rate).

* ​ là gradient của hàm mất mát đối với trọng số.

Cụ thể ta cần tính được 2 thứ đó, tách nhỏ ra từ công thức ở trên. Với W, b là tập hợp ma trận trọng số giữa các layers và biases của mỗi layer:

A black and white math symbols

Description automatically generated with medium confidence

Trong đó có cần biết đến đạo hàm hợp Chain rule:



2.1.5 Thuật toán tối ưu hóa

Các thuật toán tối ưu hóa được sử dụng để cập nhật trọng số nhằm giảm thiểu hàm mất mát. Các thuật toán phổ biến bao gồm:

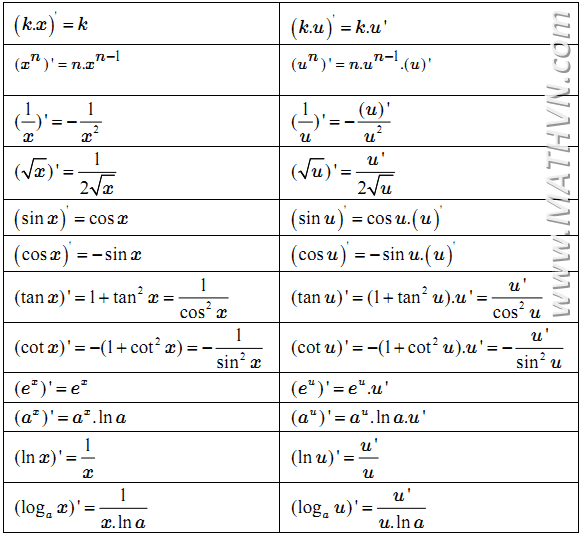
* **Gradient Descent**
* **Stochastic Gradient Descent (SGD)**
* **Adagrad**
* **RMSprop**
* **Adam**

2.1.6 Ví dụ chạy

Ở ví dụ này xin mượn ví dụ của trang nttuan8 để nói về backpropagation.

Link: https://nttuan8.com/bai-4-backpropagation/

Công thức đạo hàm:



Mô hình

* A diagram of a network

  Description automatically generated
* Với L = 2
* Mô hình: 2-2-1, nghĩa là 2 node trong input layer, 1 hidden layer có 2 node và output layer có 1 node.

Tiếp theo là công thức tính chạy trong feedforward:

* A math equations on a white background

  Description automatically generated

Nó sẽ có dạng tổng quát:

* A group of math equations

  Description automatically generated

Từ đó chúng ta dùng hàm loss: Binary cross entropy loss – BCE

* A math equations on a white background

  Description automatically generated
* A number and symbol on a white background

  Description automatically generated
* là giá trị model dự đoán (train), còn là giá trị thực tế

Dựa vào mô hình ở trên, tính toán W, b của tầng 1 và 2. Vì khi backpropagration thì tầng 2 sẽ tính trước rồi tới tầng 1. Ở trên đã giới thiệu đến đạo hàm hợp Chain rule.

Trước khi vô đạo hàm W, b của từng tầng thì có chúng mình có để giải thích thêm hình ở bên dưới:

Giải thích từ dấu = thứ nhất, thì nó là công thức sigmoid, sau khi đạo hàm thì cộng và trừ 1 với tử. Nhóm nhân tử chung lại.

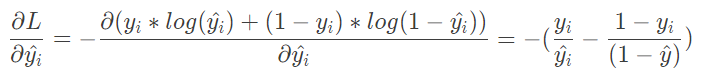
Dựa vào đây có thể giải thích được đến bước ảnh ở dưới.

Tiếp theo chúng ta sẽ đạo hàm Loss với W, b tầng 2.

Áp dụng chain rule: A math equation with black text

Description automatically generated with medium confidence

Ta tính được đạo hàm loss với kết quả dự đoán.

* + 

A diagram of a graph

Description automatically generated

Dựa vào tính toán ở trên kia.

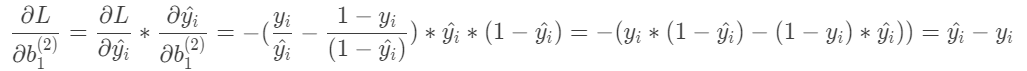
A number and symbol on a white background

Description automatically generated

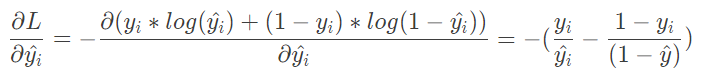
A math equations on a white background

Description automatically generated

Khi đó:



Tương tự với ở trên thế vào là được: tương tự với đạo L với a.

* 
* A number one and a symbol

  Description automatically generated with medium confidence

= -

A math equations with numbers and symbols

Description automatically generated with medium confidence

Tiếp theo chúng ta sẽ đạo hàm Loss với W, b tầng 1.



Áp dụng chain rule: A mathematical equation with numbers and lines

Description automatically generated

Ta có:

A math equations on a white background

Description automatically generated

Công thức đạo hàm a1 với b1 (l=1) thì tương tự, với B.

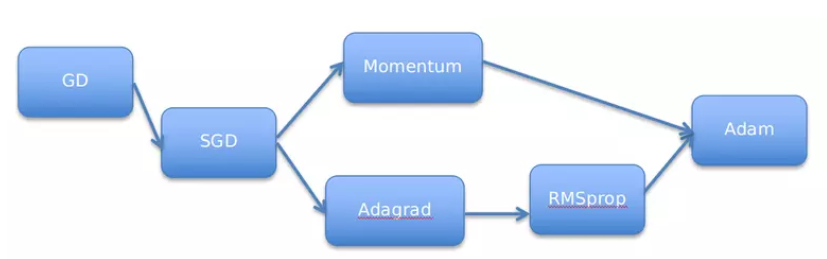
A group of math equations

Description automatically generated

* 1. Các phương pháp optimization

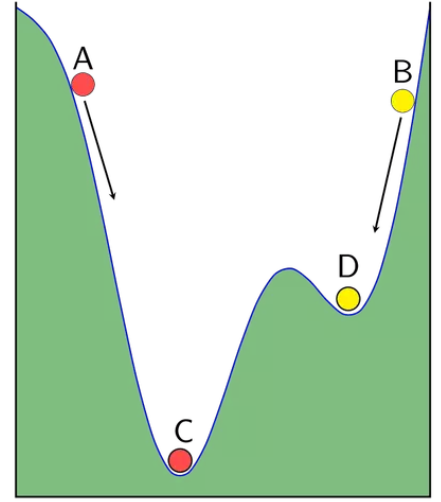
Trước khi đi sâu vào vấn đề, chúng ta cần hiểu thuật toán tối ưu (optimizers) là gì và chúng được sử dụng để làm gì. Thuật toán tối ưu là công cụ quan trọng trong việc xây dựng mô hình neural network. Chúng giúp mô hình **"học"** được các **đặc trưng (features) và mẫu (patterns)** từ dữ liệu đầu vào. Mục tiêu là tìm ra một cặp **weights** và **bias** phù hợp để tối ưu hóa hiệu suất của mô hình.

Nhưng làm thế nào để "học" được weights và bias? Không thể chỉ đơn giản là chọn ngẫu nhiên một số lần hữu hạn và hy vọng rằng ở một bước nào đó sẽ tìm được lời giải. Cách làm này không khả thi và lãng phí tài nguyên. Do đó, chúng ta cần một phương pháp hệ thống để cải thiện weights và bias theo từng bước. Đây chính là lý do mà các thuật toán tối ưu ra đời: để hướng dẫn quá trình điều chỉnh weights và bias sao cho mô hình ngày càng chính xác hơn trong việc dự đoán.



2.2.1 Gradient Descent

Gradient Descent cổ điển: Cập nhật tham số sau khi tính toán gradient dựa trên toàn bộ tập dữ liệu.

* Dự đoán 1 điểm khởi tạo:
* Cập nhật đến khi đạt được kết quả chấp nhận được:
* là đạo hàm của hàm mất mát tại
* Ưu Điểm:
* Đơn Giản và Dễ Hiểu: Thuật toán Gradient Descent cơ bản rất dễ hiểu và triển khai. Nó cập nhật trọng số sau mỗi vòng lặp dựa trên gradient của hàm loss, giúp tối ưu hóa mô hình neural network.
* Hiệu Quả: Bằng cách cập nhật trọng số dần dần, Gradient Descent giúp mô hình học được từ dữ liệu và cải thiện độ chính xác của dự đoán.
* Nhược Điểm:
* Phụ Thuộc Vào Điểm Khởi Tạo Ban Đầu: Vị trí khởi tạo của điểm ban đầu có thể ảnh hưởng lớn đến quá trình hội tụ. Nếu chọn điểm x ban đầu khác nhau, kết quả cuối cùng có thể khác nhau, đặc biệt là trong các hàm có nhiều cực trị.
* Phụ Thuộc Vào Learning Rate:
  + - Learning Rate Quá Nhỏ: Nếu tốc độ học quá nhỏ, quá trình hội tụ sẽ rất chậm, làm ảnh hưởng đến thời gian huấn luyện mô hình. Mô hình có thể mất nhiều thời gian để đạt được kết quả tốt.
    - Learning Rate Quá Lớn: Nếu tốc độ học quá lớn, mô hình có thể tiến nhanh đến gần điểm tối ưu chỉ sau vài vòng lặp. Tuy nhiên, nó có thể không hội tụ và dao động xung quanh điểm tối ưu mà không đạt được giá trị tối ưu chính xác, do bước nhảy quá lớn.
* Phụ Thuộc Vào Nghiệm Khởi Tạo: Ví dụ, nếu hàm số có nhiều điểm cực trị toàn cục (global minimum), tùy thuộc vào điểm khởi tạo ban đầu, thuật toán có thể hội tụ đến các nghiệm cuối cùng khác nhau(local minimum). Điều này có thể dẫn đến các kết quả không nhất quán.
* 

2.2.2 Stochastic Gradient Descent

Cập nhật tham số dựa trên một mẫu ngẫu nhiên (mini-batch) từ tập dữ liệu.

* Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là một **epoch**. Với **GD thông thường** thì mỗi **epoch** ứng với **1 lần** cập nhật θ, với **SGD** thì mỗi epoch ứng với **N lần cập nhật** θ với N là số điểm dữ liệu (sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên).
* Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt).
* là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là

A black and white image of a black circle with a red line

Description automatically generated

GD có hạn chế đối với cơ sở dữ liệu lớn (vài triệu dữ liệu) thì việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh. Bên cạnh đó GD không phù hợp với **online learning**.

* Online learning: là khi dữ liệu cập nhật liên tục (ví dụ như thêm người dùng đăng kí) thì mỗi lần thêm dữ liệu ta phải tính lại đạo hàm trên toàn bộ dữ liệu => thời gian tính toán lâu, thuật toán không online nữa. Vì thế SGD ra đời để giải quyết vấn đề đó, vì mỗi lần thêm dữ liệu mới vào chỉ cần cập nhật trên 1 điểm dữ liệu đó thôi, phù hợp với online learning.
* Ưu điểm :
* Thuật toán giải quyết được đối với cơ sở dữ liệu lớn mà GD không làm được. Thuật toán tối ưu này hiên nay vẫn hay được sử dụng.
* Nhược điểm :
* Thuật toán vẫn chưa giải quyết được 2 nhược điểm lớn của gradient descent ( learning rate, điểm dữ liệu ban đầu )

2.2.2 Gradient Descent với Momentum

Trong Gradient Descent (GD), chúng ta cần tính lượng thay đổi tại thời điểm t để cập nhật vị trí mới của nghiệm. Nếu coi đại lượng này như vận tốc trong vật lý, vị trí mới sẽ là:

Dấu trừ biểu thị việc di chuyển ngược với gradient. Nhiệm vụ của chúng ta là tính sao cho nó chứa cả thông tin về độ dốc (gradient) và đà (momentum) tức vận tốc trước (với vận tốc ban đầu ). .

A diagram of a graph

Description automatically generated

Ở hình b, với GD thì nếu chạy từ A thì có thể đến C là (global minimum) còn nếu từ B thì nó chỉ chạy đến D(local minimum). Còn với hình c, có 1 điểm khác là khi chạy từ B thì nó sẽ dựa vào công thức (vận tốc trước đó và đà ) sẽ giúp nó vượt qua đỉnh gần đó sẽ tới điểm E rồi tới C.

* Ưu điểm : Thuật toán tối ưu giải quyết được vấn đề: Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.
* Nhược điểm: Tuy momentum giúp hòn bi vượt dốc tiến tới điểm đích, tuy nhiên khi tới gần đích, nó vẫn mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn, điều này được giải thích vì viên bi có đà.

2.2.4 Adagrad

Khác với các thuật toán trước đó, trong quá trình huấn luyện, learning rate thường không thay đổi (được xem là một hằng số). Tuy nhiên, Adagrad coi learning rate là một tham số và cho phép nó biến thiên sau mỗi thời điểm t.

A square root of a square root of a square

Description automatically generated

* Trong đó :
* n : hằng số
* gt : gradient tại thời điểm t
* ϵ : hệ số tránh lỗi ( chia cho mẫu bằng 0), default 1e-8
* G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.
  + A diagram of a mathematical equation

    Description automatically generated
* Ưu điểm: Một lơi ích dễ thấy của Adagrad là tránh việc điều chỉnh learning rate bằng tay, chỉ cần để tốc độ học default là 0.01 thì thuật toán sẽ tự động điều chỉnh.
* Nhược điểm: Yếu điểm của Adagrad là tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

2.2.5 RMSprop

RMSprop giải quyết vấn đề giảm tốc độ học của Adagrad bằng cách chia tốc độ học cho trung bình của bình phương gradient.

A mathematical equation with numbers and symbols

Description automatically generated

* Ưu điểm: Giải quyết được vấn đề tốc độ học giảm dần của Adagrad (vấn đề tốc độ học giảm dần theo thời gian sẽ làm quá trình đào tạo chậm lại, có khả năng dẫn đến trì trệ).
* Nhược điểm: Thuật toán RMSprop có thể chỉ tìm ra các giải pháp tối thiểu cục bộ thay vì đạt đến mức tối thiểu toàn cầu như Momentum. Do đó, người ta kết hợp cả thuật toán Momentum và RMSprop để tạo ra thuật toán tối ưu mang tên Adam. Chúng ta sẽ thảo luận về nó trong phần sau.

2.2.5 Adam

Như đã đề cập ở phía trên, Adam là sự kết hợp của Momentum và RMSprop. Nếu diễn giải theo hiện tượng vật lí, Momentum tương tự như quả cầu lao xuống dốc, trong khi Adam giống như quả cầu rất nặng có ma sát (friction). Do đó, Adam dễ dàng vượt qua local minimum để đạt tới global minimum và khi đạt tới global minimum, nó không mất nhiều thời gian dao động quanh điểm cuối vì có ma sát, dễ dừng lại hơn.

A graph of a ball with a number of points

Description automatically generated

Công thức update:

* A math equations on a white background

  Description automatically generated

TÀI LIỆU THAM KHẢO

1. <http://vap.ac.vn/Portals/0/TuyenTap/2020/4/27/e28bc8dd2a3f4e77967c747c7a3c4e18/81.pdf>
2. <https://phamdinhkhanh.github.io/deepai-book/ch_ml/Boosting.html>
3. <https://ant.ncc.asia/series-bagging-va-random-forest-phan-2/>
4. https://machinelearningcoban.com/2017/02/24/mlp/#-cac-ky-hieu-va-khai-niem
5. https://machinelearningcoban.com/2017/01/16/gradientdescent2/#gradient-descent-voi-momentum
6. https://nttuan8.com/bai-2-logistic-regression/#Loss\_function
7. https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn