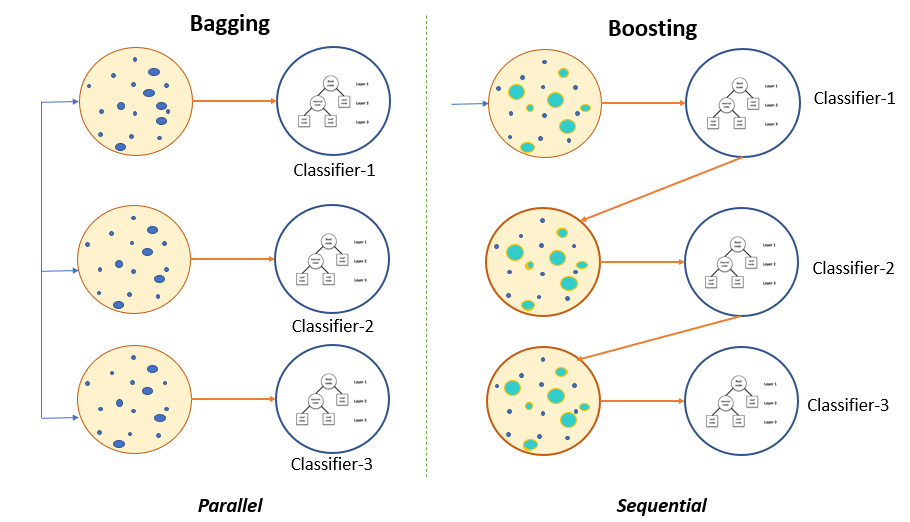
**Class 9 : Boosting techniques ( XGBoost & Ada Boost & Gradient Tree boosting)**

1. **Lý Thuyết**

Ở thuật toán random forest, chúng ta đã biết đến phương pháp bagging ( là chạy song song nhiều mô hình decision tree sau đó lấy xác suất đa số để tối đa hóa sự chính xác, mỗi mô hình là độc lập và chạy song song với nhau, kết quả của mô hình này không ảnh hưởng mô hình kia ).

Thì boosting cũng là một phương pháp để tối đa hóa sự chính xác , tuy nhiên , phương pháp boosting sẽ chạy từng mô hình



Graphical user interface

Description automatically generated with medium confidence

Trong đó ,

Bagging : Xây dựng nhiều model trên những subsamples lấy từ data sample , những model được train độc lập và song song . Sau đó kết quả của chúng sẽ được lấy trung bình cộng để cho ra kết quả cuối cùng.

Boosting : Cũng xây nhiều model, tuy nhiên model sau sẽ học cách sửa những erros của những model trước => **Tạo thành 1 chuỗi các model mà model sau sẽ tốt hơn model trước,** lấy kết quả cuối cùng của model cuối cùng làm kết quả.

Bagging và Boosting còn được gọi chung là ensemble learning ( tức là học tập thể )

Ngoài ra còn có phương pháp Stacking ( cái này bạn có thể tra google nếu muốn tìm hiểu thêm)

Nhìn chung , các phương pháp này đều hướng đến việc cải thiện kết quả ( Variance & bias )

Tất cả các thao tác trên bản chất là xoay quanh để giải quyết vấn đề giữa bias và variance ([bias-variance trade off](https://medium.com/@mp32445/understanding-bias-variance-tradeoff-ca59a22e2a83)). Hiểu đơn giản về bias-variance trade off tức là "chúng ta mong muốn mô hình khi fit vào dữ liệu sẽ có bias thấp và variance thấp, tuy nhiên, bias và variance thường có xu hướng nghịch đảo với nhau. - bias thấp nhưng variance cao hoặc variance thấp nhưng bias cao, và chúng ta chỉ có thể lựa chọn tăng cái này và chấp nhận giảm cái kia". Bui Tien Dung, viblo asia.

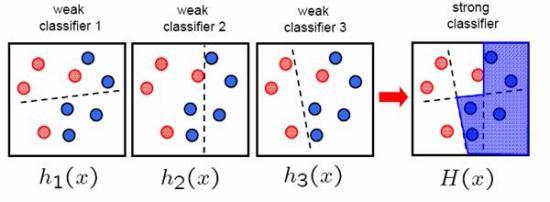
A picture containing circle

Description automatically generated

* 1. **Các thuật toán phổ biến**

Text

Description automatically generated with medium confidence



Các thuật toán boosting đều nhằm để giải quyết bài toán tối ưu Loss Function. ( Ở đây chúng ta không nhắc lại khái niệm loss function nữa, mà sẽ đi thẳng vào thuật toán).

Xây dựng nhiều thuật toán cùng lúc ( ví dụ như Decision Tree ) và kết hợp chúng lại, mục đích là để có 1 cụm hoặc 1 nhóm các weak learner rồi kết hợp chúng lại để tạo ra 1 strong learner.

Weak learner : phân loại với độ chính xác không cao

Strong learner : phân loại với độ chính xác cao hơn

Nguyên lý hoạt động : để tìm ra các điểm yếu của các model ( weak rule ) , sử dụng thuật toán cơ sở ( base learning algorithm ) – tức thuật toán mà ta đang dùng với một phân phối ( distribution ) khác.

* Với mỗi distribution mới áp dụng cho base learning algorithm , nó tạo ra một weak prediction rule mới.
* Lặp đi lặp lại nhiều lần để kếtt hợp các weak rules này thành một prediction rule duy nhất
  + 1. **Adaptive Boosting (AdaBoost)**

AdaBoost tiến hành train các mô hình mới dựa trên việc đánh lại trọng số cho các điểm dữ liệu hiện tại, nhằm giúp các mô hình mới có thể tập trung hơn vào các mẫu dữ liệu đang bị học sai, từ đó làm giảm giá trị loss của mô hình.

ADABoost là một classifier , do vậy hoạt động tốt nhất với các bài toàn classification

Các bước triển khai như sau :

* Xác định một base weak learner, có thể là bất cứ kiểu classification nào, các loại thuật toán phân loại đơn giản như KNN ,logistic regression , hoạt động tốt như là 1 weak learner.
* Bắt đầu với trọng số bằng nhau cho mỗi điểm dữ liệu = 1/N ( N là tổng số điểm dữ liệu – records).
* Fit mô hình classification đầu tiên, sau đó cho training data sử dụng các trọng số ban đầu.
* Sau đó, tính toán tổng lỗi phát sinh sau khi chạy mô hình
* Tiếp đó, tính trọng số mới với giá trị tỉ lệ lỗi phát sinh

Text

Description automatically generated

* Update trọng số mới của model,

Với những kết quả dự đoán sai , trọng số sẽ bằng :



Với những kết quả dự đoán đúng , trọng số sẽ bằng :



 (Các điểm dữ liệu bị đoán sai --> tăng trọng số, các điểm dữ liệu đoán đúng --> giảm trọng số)

Trọng số càng gần mức 0.5 thì Total Error càng gần = 0

* Dựa vào những trọng số mới , một dataset mới sẽ được tạo ra với tần suất xuất hiện của những records bị classify sai sẽ nhiều hơn những records được classify đúng.
* Lặp lại quá trình trên

Notes : AdaBoost có thể được áp dụng mà không cần dựa vào việc đánh trọng số lại các điểm dữ liệu, thay vào đó, chúng ta có thể re-sample lại dataset để lấy dữ liệu train cho các model tiếp theo dựa vào xác suất được xác định bới các trọng số.

Xem thêm tại : [Link](https://www.youtube.com/watch?v=GM3CDQfQ4sw)

* + 1. **Gradient Boosting**

Gradient boosting là 1 dạng tổng quát hóa của ADABOOST. Tuy nhiên, gradient boosting được dùng cho những bài toán phức tạp hơn mà ADABoost không làm tốt được. ( ví dụ ở bài toán classification yes/no ở dạng binary thì ADAboost làm tốt, nhưng có những bài toán có quá nhiều class, hoặc nhiều features phức tạp ).

Gradient boost sẽ sinh ra dataset mới để train model dựa trên residual giữa errors/ correct , => gradient boost cố giảm thiểu residuals.

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Xem thêm tại : [Link](https://www.youtube.com/watch?v=0ARLObJJ3y4)

* + 1. **XGBoost (Extreme Gradient Boosting)**

XGB là một thuật toán dựa trên Gradient Boosting, có thể hiểu XGB là phiên bản cải tiến của Gradient Boosting.

XGB boost là một trong những thuật toán phổ biến nhất hiện nay, có thể thực hiện cả 2 công việc là regression và classification.

Là thuật toán được sử dụng ở hầu hết các cuộc thi trên Kaggle hoặc Hackathon

Về những thuật toán boosting này ,rất may mắn là chúng đều đã được support bằng những thư viện Sklearn , XGBoost classifier nên chúng ta không cần phải hiểu quá sâu và tường tận, chỉ cần nắm vững nguyên lý hoạt động của chúng và apply được là được.

Practice with Loan Prediction

Practice : [Colab](https://colab.research.google.com/drive/1NZk0aubDpFpFwELY2bo88t85n6jkOkUf?usp=sharing)