I. Kỹ thuật ensemble

Kỹ thuật esemble là kết hợp nhiều mô hình học máy để tạo ra dựa đoán tối ưu nhất cho bài toán.

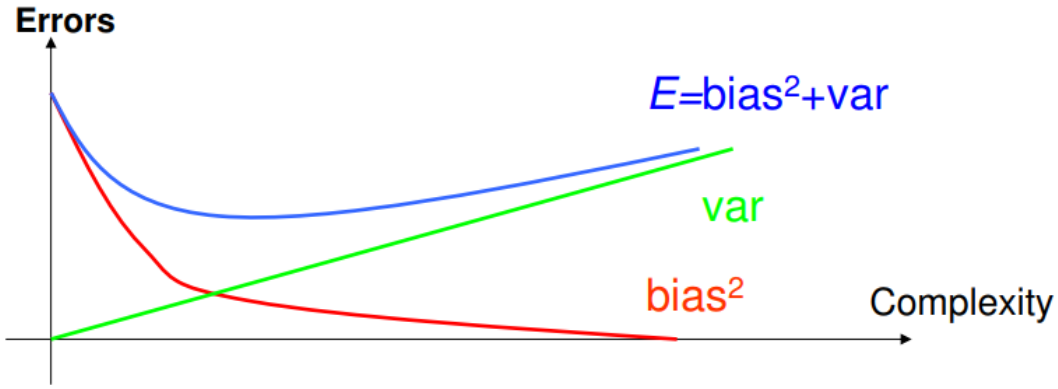
Một số kỹ thuật ensemble:

+ Bagging
+ Boosting

* Tính chất:

-Xây dựng tập hợp các mô hình cơ sở dựa trên tập train.
-Kết hợp các mô hình khi phân loại cho độ chính xác cao.
-Dựa trên cơ sở bias/ variance
-Áp dụng cho nhiều giải thuật cơ sở như cây quyết định, SVM, Naive Bayes
-Giải quyết các vấn đề về phân loại, hồi quy, phân cụm,..
-Cho kết quả tốt, tuy nhiên không thể dịch được kết quả sinh ra
-Được ứng dụng thành công trong các lĩnh vực tìm kiếm thông tin, nhận dạng, phân tích dữ liệu

* Cơ sở bias/variance:

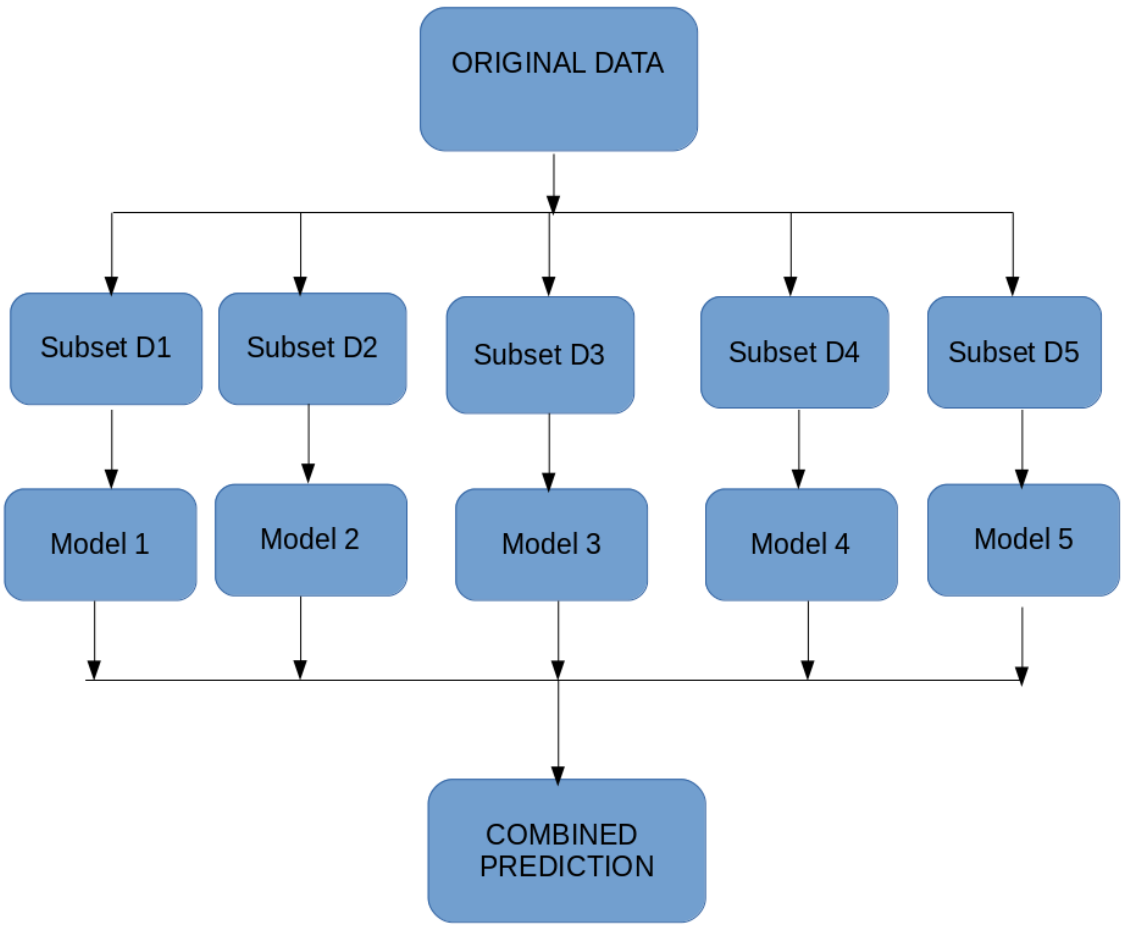


- Bias: thành phần lỗi độc lập với train data
- Variance: thành phần lỗi do biến động liên quan đến sự ngẫu nhiên của train data

1. Bagging

Bagging (viết tắt của Boostrap Aggregating), là kỹ thuật kết hợp nhiều mô hình song song độc lập nhau trên các tập con của tập dữ liệu, các tập con có thể có phần tử lặp, dựu đoán cuối cùng dựa vào sự kết hợp dự đoán của tất cả các mô hình. Bagging dùng để giảm overfitting trong những mô hình phức tạp.

Bagging giảm variance.



Một số thuật toán bagging: Bagging meta-estimator, Random Forest,...

1.1. Bagging meta-estimator

Bao gồm BaggingClassifier và BaggingRegressor. Các bước:

1. Tạo các tập con ngẫu nhiên từ tập gốc (Boostrapping).
2. Mỗi tập con được fit trên base estimator.
3. Kết hợp dự đoán của mỗi model để cho ra kết quả cuối cùng.

Một số tham số quan trọng:

- base_estimator: mô hình cơ sở để fit các tập con ngẫu nhiên, mặc định là decision tree.
- n_estimators: số estimator được tạo ra.
- max_samples: quy định kích thước của tập con.
- max_features: số feature chạy trọng toàn bộ dữ liệu.
- n_jobs: số công việc chạy đồng thời , nên đặt bằng số nhân cpu, nếu =-1 thì bằng số nhân.

1.2. Random Forest

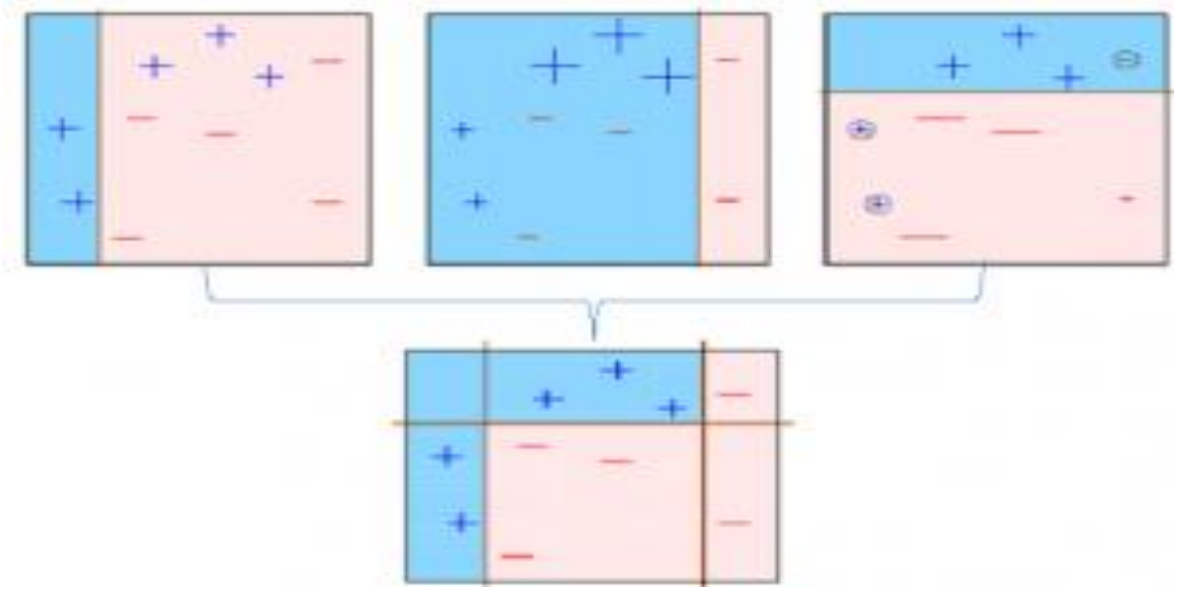
Là một thuật toán mở rộng của Bagging, có base_estimator là decision tree. Khác với Bagging, Random Forest chọn ngẫu nhiên một tập các feature để quyết định phân chia tốt nhất tại mỗi nút của decision tree. Dự đoán cuối cùng được tính bằng cách tính trung bình cộng của các dự đoán của từng decision tree.

Bao gồm : RandomForestClassifier và RandomForestRegressor

2. Boosting

Boosting là kỹ thuật kết hợp các mô hình một cách tuần tự, mô hình sau sẽ cố gắng sửa các lỗi của mô hình trước đó. Boosting kết hợp các weak learner để tạo ra strong learner. Boosting dùng để tăng sự linh hoạt cho những mô hình đơn giản.

Boostring giảm bias.



Các thuật toán Boosting bao gồm: AdaBoost, GBM, XGBM, Light GBM, Cat-Boost,

3. So sánh Bagging và Boosting

Giống nhau	Khác nhau
Đều là mô hình tập hợp N learner từ 1 learner	- Bagging: các mô hình chạy độc lập - Boosting: mô hình sau cố gắng sửa lỗi của mô hình trước
Đều sinh ra một số tập con ngẫu nhiên	Chỉ có Boosting xác định trọng số cho dữ liệu, dữ liệu phân loại sai được tăng trọng số để môh ình sau tập trung vào các trường hợp khó nhất.
Đều sử dụng quyết định cuối bằng cách tínhtrung bình N learner	-Bagging: TRung bình với trọng số bàng nhau -Boosting: Tính trung bình có trọng số

Đều giảm phương sai và cung cấp sự ổn định hơn	-Bagging: Giải quyết over-fitting - Boosting: Giảm bias, có thể làm tăng overfitting.

II. Lan truyền ngược

(Backpropagtion algorithm)

Là phương pháp giúp tính gradient ngược từ layer cuối cùng đến layer đầu tiên. Layer cuối được tính trước vì gần với predicted outputs và loss function. Việc tính toán dựa trên quy tắc “chain rule” (đạo hàm của hàm hợp).

Với lớp cuối cùng:

$$\frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = e_j^{(L)} a_i^{(L-1)}$$

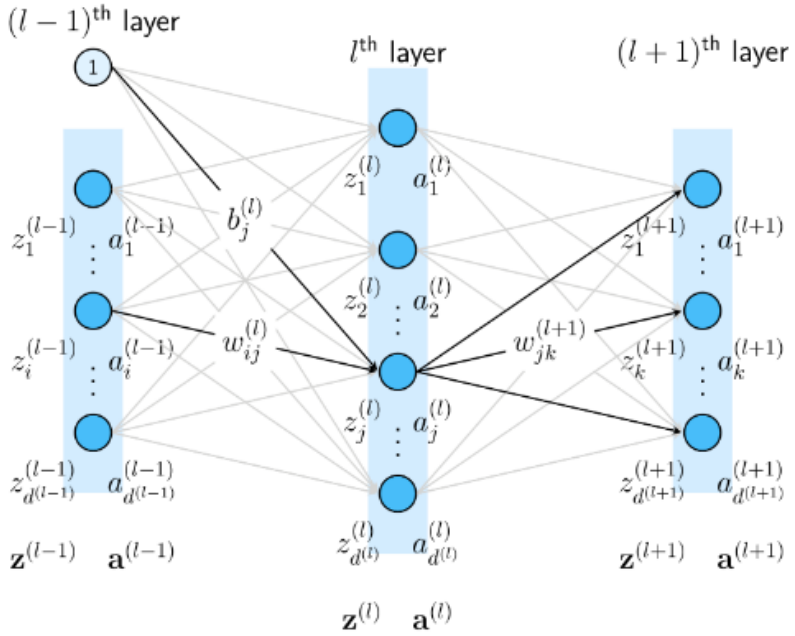
Với :

$$\frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = a_i^{(L-1)} \quad \text{do} \quad z_j^{(L)} = \mathbf{w}_j^{(L)T} \mathbf{a}^{(L-1)} + b_j^{(L)}$$

Ta lại có:

$$\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)}$$

Ở các lớp l thấp hơn:



$$\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$$

$$\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$$

$$z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

Ta tính được:

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} . \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$$
$$= e_j^{(l)} a_i^{(l-1)}$$

Với:

$$e_j^{(l)} = \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} . \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}$$
$$= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} . \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \right) f'(z_j^{(l)})$$
$$= \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)} \right) f'(z_j^{(l)})$$
$$= \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)} \right) f'(z_j^{(l)})$$

III. Stochastic Gradient Descent

Thuật toán Gradient Descent:

1. Dự đoán một điểm khởi tạo: $\theta = \theta_0$
2. Cập nhật θ cho đến khi đạt được kết quả chấp nhận được:

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

Trong đó:

+ θ là tập các tham số của mô hình.
+ $J(\theta)$ là hàm mất mát.
+ η là learning rate.

Trong thuật toán SGD, tại một thời điểm chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu $\mathbf{x}_i$ rồi cập nhật θ dựa trên đạo hàm này. Thực hiện với từng điểm trên toàn bộ tập dữ liệu, sau đó lặp lại quá trình trên.

Mỗi lần duyệt 1 lượt tất cả các phần tử trong tập dữ liệu gọi là epoch. Với GD thì mỗi epoch cập nhật θ 1 lần, còn SGD thì cập nhật N là với N là số điểm dữ liệu. Việc cập nhật từng điểm một có thể làm giảm đi tốc độ thực hiện một epoch, tuy nhiên SGD 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.

*** *Thứ tự chọn điểm dữ liệu:*

Cần lưu ý: sau mỗi epoch, cần xáo trộn thứ tự dữ liệu để đảm bảo tính ngẫu nhiên, việc này ảnh hưởng tới hiệu năng của SGD.

Quy tắc cập nhật của SGD:

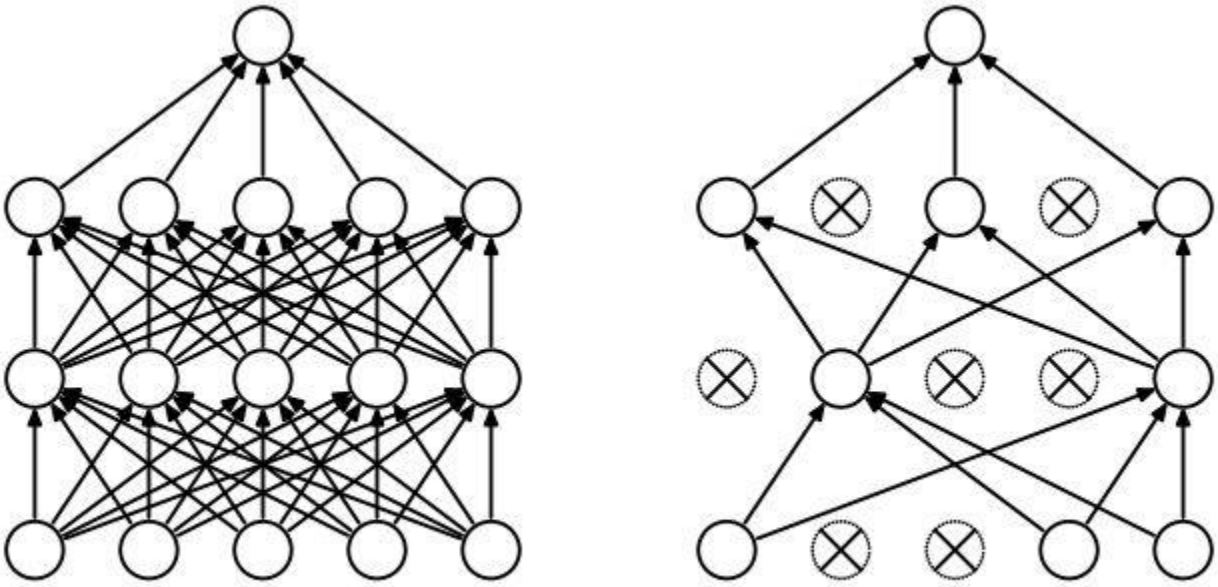
$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}_i; \mathbf{y}_i)$$

Trong đó $J(\theta; \mathbf{x}_i, \mathbf{y}_i)$ là hàm mất mát với chỉ một cặp dữ liệu là $(\mathbf{x}_i, \mathbf{y}_i)$.

Ưu điểm của SGD là hội tụ rất nhanh.

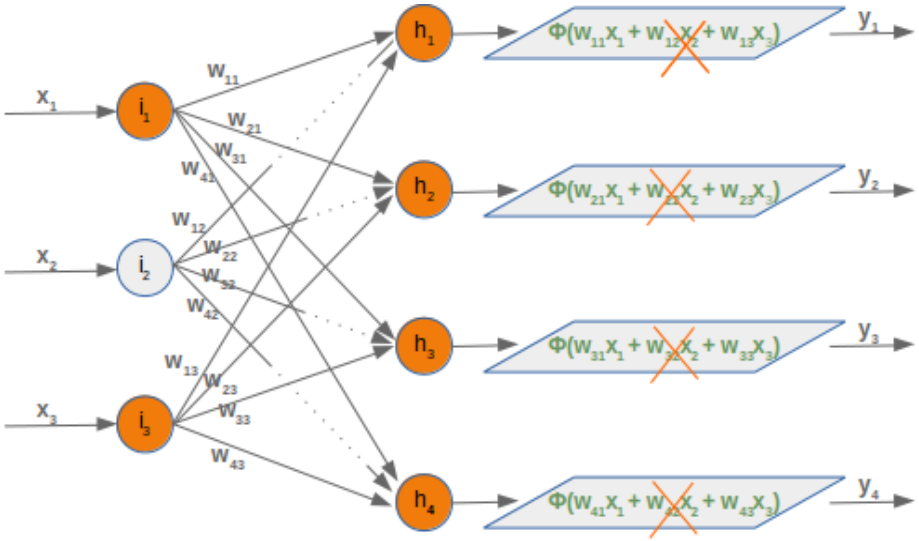
IV. Drop out

- Trong neural network, việc cuối cùng là tối ưu các tham số để giảm los function, nhưng đôi khi có unit thay đổi theo cách sửa lại lỗi của các unit khác dẫn đến việc hoà trộn làm giảm tính dự đoán của model, tức là overfitting.
- Dropout là cách thức giả định một phần của unit bị ẩn đi trong quá trình training, để giảm tích hoà trộn (1 hidden unit không thể dựa vào một unit khác để sửa lỗi của mình)



(a) Standard Neural Net (b) After applying dropout.

- Tại mỗi step trong quá trình training, khi thực hiện Lan truyền xuôi đến layer sửa dụng Dropout, thay vì tính toán tất cả unit có trên layer, tại mỗi unit ta sẽ random xem unit đó có được tính hay không dựa trên xác suất p.



- Dropout dùng để đạt được kết quả trung bình cảu việc train nhiều mạng con trong network.