**MODELs &**

**FEATURE ENGINEERING**

# **PART 1: SUPERVISED MODEL**

### **CLASSIFICATION**

### **REGRESSION**

# **PART 2: UN-SUPERVISED MODEL**

# **PART 3: ENSEMBLE LEARNING**

Ensemble nghĩa là “nhóm”. Cái tên thể hiện phần nào đặc trưng của mô hình Ensemble Learning, đó là nó kết hợp nhiều mô hình học máy yếu (high bias hoặc week learner) khác nhau để tạo ra một mô hình tổng hợp mạnh mẽ hơn (high variance hoặc strong learner). Thay vì dựa vào một mô hình duy nhất, ensemble learning tận dụng sự đa dạng và sự đồng thuận của các mô hình khác nhau để tạo ra dự đoán chính xác và ổn định hơn.

Có nhiều phương pháp ensemble learning khác nhau, bao gồm Bagging, Boosting, và Stacking. Mỗi phương pháp này có cách tiếp cận riêng để kết hợp các mô hình, nhưng mục tiêu chung là tạo ra một mô hình tổng hợp có khả năng dự đoán tốt hơn so với bất kỳ mô hình nào trong ensemble. Ensemble learning thường được áp dụng trong các bài toán phân loại và dự báo để cải thiện độ chính xác và độ ổn định của dự đoán.

Trước khi đi sâu vào mô hình, hãy cùng tìm hiểu về học máy yếu, còn được gọi là high variance hoặc strong learner.

**Weak learner**

Máy học yếu là thuật ngữ được sử dụng trong Học máy để chỉ một mô hình dự đoán đơn giản, không có khả năng dự đoán tốt một cách độc lập trên tập dữ liệu huấn luyện. Một weak learner thường có độ chính xác thấp hoặc có khả năng dự đoán không tốt trên một phần của dữ liệu.

Mặc dù weak learners không đủ mạnh để tạo ra dự đoán chính xác độc lập, nhưng chúng vẫn có ích trong ensemble learning, nơi chúng được kết hợp với nhau để tạo ra một mô hình tổng hợp mạnh mẽ hơn.

A diagram of a weak learner

Description automatically generated

***Lựa chọn các mô hình yếu để tạo ra mô hình mạnh***

Trước hết, ta chọn một mô hình cơ sở, thường là một mô hình đơn giản như Decision Tree. Tuy nhiên, vì cần nhiều mô hình "yếu" để tạo thành một mô hình "mạnh", ta tăng số lượng mô hình cơ sở lên, ví dụ có thể có nhiều Decision Tree khác nhau. Việc lựa chọn mô hình cơ sở cần phù hợp với chiến thuật ban đầu đã chọn. Ví dụ, nếu chọn một mô hình có bias thấp nhưng variance cao, ta cần tạo ra một mô hình tổ hợp có xu hướng giảm variance. Ngược lại, nếu chọn một mô hình có bias cao nhưng variance thấp, mô hình tổ hợp cần giảm bias. Điều này giúp cân bằng giữa bias và variance, từ đó tạo ra một mô hình tổ hợp có khả năng dự đoán tốt hơn trên dữ liệu mới.

**Bias cao**: Một mô hình có bias cao có xu hướng dự đoán kém chính xác trên cả tập huấn luyện và tập kiểm tra. Nó có thể do mô hình quá đơn giản và không thể biểu diễn được mối quan hệ phức tạp trong dữ liệu. Ví dụ: Một mô hình hồi quy tuyến tính được sử dụng để dự đoán giá nhà. Nếu mô hình chỉ sử dụng một biến độc lập (ví dụ: diện tích của căn nhà) để dự đoán giá nhà, mô hình có thể có bias cao vì nó không thể biểu diễn được các yếu tố khác như vị trí, tiện ích xung quanh, v.v.

**Variance cao**: Một mô hình có variance cao dự đoán tốt trên tập huấn luyện nhưng dự đoán kém trên tập kiểm tra. Nó có thể do mô hình quá phức tạp và quá nhạy cảm với nhiễu trong dữ liệu huấn luyện. Ví dụ: Một mô hình cây quyết định rất sâu được sử dụng để dự đoán việc sinh trưởng cây dựa trên nhiều yếu tố khác nhau như nhiệt độ, độ ẩm, loại đất, v.v. Mô hình này có thể có variance cao nếu nó quá phức tạp và dễ bị overfitting, tức là nó học những điều tốt nhất từ dữ liệu huấn luyện nhưng không tổng quát hóa được cho dữ liệu mới.

Một mô hình có thể có cả bias cao và variance cao, khi đó nó sẽ dự đoán kém chính xác trên cả tập huấn luyện và tập kiểm tra. Điều này thường xảy ra khi mô hình quá đơn giản để biểu diễn được mối quan hệ phức tạp trong dữ liệu, nhưng cũng quá nhạy cảm với nhiễu trong dữ liệu.

Ngược lại, **một mô hình tốt sẽ có cả bias thấp và variance thấp**, dự đoán tốt trên cả tập huấn luyện và tập kiểm tra. Điều này xảy ra khi mô hình có độ phức tạp phù hợp để biểu diễn được mối quan hệ trong dữ liệu và không bị overfitting hoặc underfitting

Trong thực tế, việc tối ưu hóa bias và variance là một quá trình phức tạp trong Machine Learning. Đôi khi, cần phải làm việc để giảm bias mà không tăng variance, hoặc ngược lại, tùy thuộc vào bài toán cụ thể và tính chất của dữ liệu.

**Biến thể của Ensemble learning**[[1]](#footnote-1)

* **Bagging**: Giảm variance
* **Boosting:** Tập trung vào việc giảm bias (cũng giảm cả variance)
* **Stacking**:*Tập trung vào việc giảm bias (cũng giảm cả variance)*

## **BAGGING**

Với Bagging, chúng ta tạo ra nhiều mô hình dự đoán khác nhau trên các mẫu con khác nhau từ tập dữ liệu huấn luyện. Mỗi mô hình này được huấn luyện độc lập và đồng thời với nhau. Khi dự đoán, đầu ra của tất cả các mô hình được trung bình cộng lại để tạo ra kết quả cuối cùng. Điều này giúp giảm thiểu sự biến động và tăng tính ổn định của dự đoán.

Xây dựng một lượng lớn các model (thường là cùng loại) trên những subsamples khác nhau từ tập training dataset (random sample trong 1 dataset để tạo 1 dataset mới). Những model này sẽ được train độc lập và song song với nhau nhưng đầu ra của chúng sẽ được trung bình cộng để cho ra kết quả cuối cùng.

Thuật toán **bagging** còn được gọi là **Boostrap Aggregating** sẽ xây dựng nhiều cây quyết định độc lập từ tập dữ liệu gốc với các mẫu khác nhau và tổng hợp kết quả dựa trên từng cây quyết định. Thuật toán diễn ra như sau:

* Bước 1: Lấy ngẫu nhiên có hoàn lại (boostrap) các quan sát từ tập dữ liệu gốc. Số lượng quan sát của mỗi bootstrap thông thường bằng số lượng quan sát từ tập dữ liệu ban đầu.
* Bước 2: Với mỗi tập dữ liệu boostrap, xây dựng 1 cây quyết định
* Bước 3: Tổng hợp mô hình & kết quả dự báo từ tất cả các cây quyết định đã xây dựng

**Điểm mạnh của Bagging:**

* Giảm variance: Bagging giúp giảm variance của mô hình tổ hợp bằng cách tạo ra nhiều mô hình "yếu" dựa trên các mẫu con khác nhau từ tập dữ liệu huấn luyện và sau đó kết hợp chúng. Bằng cách này, Bagging giảm nguy cơ overfitting và làm cho mô hình tổ hợp tổng thể ổn định hơn trên dữ liệu mới.
* Tính đa dạng: Bằng cách tạo ra các mẫu con khác nhau từ dữ liệu huấn luyện, Bagging tạo ra các mô hình đa dạng. Điều này giúp mô hình tổ hợp biểu diễn được nhiều khía cạnh khác nhau của dữ liệu hơn và cải thiện khả năng tổng quát hóa.
* Dễ triển khai và hiệu quả tính toán: Bagging không đòi hỏi nhiều điều kiện về loại mô hình cơ sở và thường dễ triển khai. Nó cũng có thể được tính toán song song trên nhiều CPU hoặc máy tính, làm tăng hiệu suất tính toán.

**Điểm yếu của Bagging:**

* Khả năng giảm bias không lớn: Các model trong Bagging đều là học một cách riêng rẽ, không liên quan hay ảnh hưởng gì đến nhau, điều này trong một số trường hợp có thể dẫn đến kết quả tệ khi các model có thể học cùng ra 1 kết quả. Chúng ta không thể kiểm soát được hướng phát triển của các model con thêm vào bagging. Do đó, dù Bagging giúp giảm variance, nhưng nó có thể không giảm bias nhiều. Nếu mô hình cơ sở có bias cao, thì việc kết hợp chúng bằng Bagging có thể không cải thiện hiệu suất dự đoán.
* Tăng chi phí tính toán: Tạo ra nhiều mô hình từ các mẫu con khác nhau tăng chi phí tính toán, đặc biệt là khi có nhiều dữ liệu hoặc khi mô hình cơ sở phức tạp.
* Không phù hợp cho mô hình cơ sở không ổn định: Nếu mô hình cơ sở quá đơn giản hoặc quá phức tạp, hoặc dễ bị nhiễu, Bagging có thể không cải thiện hiệu suất dự đoán và có thể dẫn đến kết quả kém hơn trên tập kiểm tra.

A diagram of a model

Description automatically generated

### **Bootstrapping**

Đây là một kỹ thuật thống kê từ 1 bộ dữ liệu N sinh ra B bộ dữ liệu mới (bootstrap samples) (thường có kích thước với bộ dữ liệu ban đầu). Mình sẽ không đi sâu vào kỹ thuật này vì thời lượng có hạn. Dưới đây là một ví dụ ngắn về kỹ thuật này, áp dụng trên bộ dữ liệu Sonar

A screenshot of a computer code

Description automatically generated

### **Bagging**

Khi train một model, bất kể đó là classification hay regression, ta đều phải định nghĩa 1 hàm nhận đầu vào và trả về đầu ra xác định bởi kết quả training. Như thế, với đầu vào khác nhau chúng ta sẽ có nhiều model khác nhau, sơ sơ là thế.

Ý tưởng về bagging khá đơn giản: mình fit một đống model trên B bootstrap samples và tính trung bình cộng kết quả dự đoán của các model đó nhằm kiếm được một model có variance thấp. NHƯNG việc fit độc lập từng model cần nhiều dữ liệu mà bạn biết đấy dữ liệu thì đào đâu ra. Cho nên chúng ta đành phải tạo bộ dữ liệu mới dựa trên bộ dữ liệu cũ bằng kỹ thuật bootstrapping. Những dữ liệu mới này có phân phối giống nhau và gần như độc lập nên cũng không ảnh hưởng tới kết quả cuối cùng tổng hợp từ các model "yếu", việc tính trung bình cộng đầu ra của các base model này cũng sẽ giảm variance.

Sơ lược dưới dạng toán học:

Ta có L bootstrap samples (tương ứng với L bộ dữ liệu) có kích thước B.

A black text on a white background

Description automatically generated A black text on a white background

Description automatically generated

Tương ứng với L bộ dữ liệu là L model "yếu".

A black text with dots and circles

Description automatically generated

Kết hợp các model này lại, ta được một model mới mạnh hơn. Với những vấn đề khác nhau, như regression, đầu ra của các model "yếu" sẽ được trung bình cộng, kết quả này sẽ là đầu ra của model "mạnh". Còn với classification, class đầu ra của mỗi một model "yếu" sẽ được coi là 1 vote và class mà nhận được số vote nhiều nhất sẽ là đầu ra của model "mạnh" (cách này gọi là hard-voting). Trong trường hợp model "yếu" dự đoán xác suất của tất cả class thì ta sẽ tính trung bình cộng của xác suất của từng class rồi lấy xác suất có giá trị lớn nhất (cách này gọi là soft-voting).

A close-up of words

Description automatically generated

Cuối cùng, để chốt phần lý thuyết và sang phần code, tôi sẽ chỉ ra một trong những lợi ích mà bagging mang lại, đó là tính song song. Như hình dưới, bạn sẽ thấy phần core của bagging đều là tiến trình song song nên nếu bạn có con máy khỏe, bạn có thể train từng model song song với nhau và cuối cùng tổng hợp đầu ra của các model này lại.

### **Code of Bagging**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

### **Random forests**

Model này là một ví dụ điển hình của phương thức Bagging. Tôi sẽ giả định các bạn đã hiểu khái niệm của random forests bởi giải thích nó cũng tốn kha khá thời gian (tôi sẽ chỉ tập trung vào ensemble và các biến thể của nó nên thông cảm nhé ).

Random forests là một tổ hợp của một đống Decision Tree, trong khi phát triển Tree, model này dùng một trick khác trong việc sinh dữ liệu: thay vì chỉ lấy mẫu qua observation trong tập dữ liệu để sinh mẫu, ta sẽ lấy mẫu trên tất cả features và chỉ lấy ngẫu nhiên 1 tập con để xây dựng tree.

Ví dụ

A screenshot of a computer

Description automatically generated

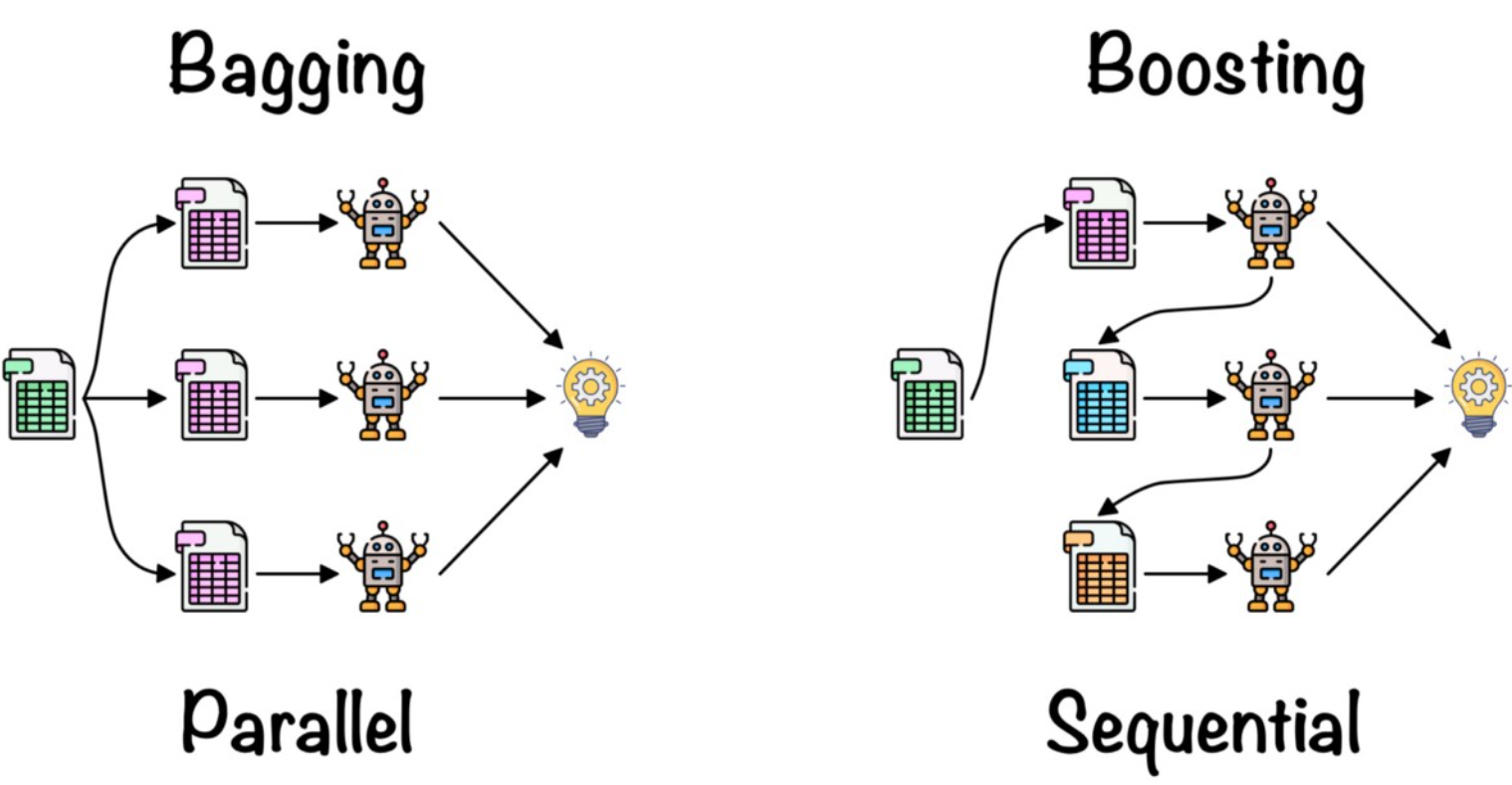
Với cách xây dựng dữ liệu của random forests, tất cả các trees sẽ không sợ đầu vào giống nhau aka đầu ra của các trees khác nhau. Một lợi ích khác của trick này sẽ giúp Decision Tree hoạt động hiệu quả hơn khi thiếu dữ liệu aka observations.

## **BOOSTING**

Trong quá trình huấn luyện, mỗi mô hình sau sẽ cố gắng sửa các lỗi mà mô hình trước đã thực hiện trên dữ liệu. Cụ thể, nếu một điểm dữ liệu đã được dự đoán sai bởi mô hình trước, thì trong quá trình huấn luyện mô hình sau, trọng số của điểm dữ liệu đó sẽ được tăng lên để mô hình sau tập trung vào việc dự đoán chính xác các điểm dữ liệu đó. Quá trình này tạo ra một chuỗi các mô hình, mỗi mô hình cố gắng cải thiện từ mô hình trước bằng cách tập trung vào những điểm dữ liệu mà mô hình trước dự đoán sai. Cuối cùng, kết quả trả về được quyết định bởi mô hình cuối cùng trong chuỗi. Vì mỗi mô hình sau cố gắng cải thiện từ mô hình trước, nên kết quả của mô hình cuối cùng thường được coi là tốt hơn và được sử dụng cho dự đoán cuối cùng.

**Giải quyết được điểm yếu của Bagging**!

Như vậy các model yếu của thể hỗ trợ lẫn nhau, học được từ nhau để tránh đi vào các sai lầm của model trước đó. Đây là điều Bagging không làm được.



**Hàm mục tiêu**

Mục tiêu của quá trình tối ưu hóa là tìm ra các hệ số ​ và các trọng số ​ sao cho hàm mất mát này đạt giá trị nhỏ nhất. Điều này giúp đảm bảo rằng ensemble của các weak learner sẽ tạo ra dự đoán tốt nhất có thể

Trong đó:

* là số lượng các weak learners trong ensemble.
* : là hệ số được áp dụng cho weak learner thứ n (confidence score)
* : là trọng số được áp dụng cho weak learner thứ n
* : hàm mất mát (Loss function) với giá trị nhãn thực tế và là dự đoán được tạo ra bằng cách kết hợp các weak learner

Sau khi mỗi week learning tìm ra kết quả, kết quả sẽ được truyền vào week learner tiếp sau đó.

Ví dụ:

* Bước 1: Weak learner đầu tiên được huấn luyện trên tập dữ liệu. Kết quả: 0.7
* Bước 2: Weak learner thứ hai được huấn luyện trên tập dữ liệu, nhưng với việc tập trung vào các điểm dữ liệu mà weak learner đầu tiên dự đoán sai. Kết quả: 0.5
* Bước 3: Cả hai weak learner được kết hợp lại với các trọng số (ví dụ: C1​=0.5, C2​=0.5).

Tổng kết quả: 0.7×0.5+0.5×0.5=0.60.7×0.5+0.5×0.5=0.6

Thay vì cố gắn quét tìm tất cả các giá trị để tìm nghiệm tối ưu toàn cục – một công việc tốn nhiều thời gian và tài nguyên, chúng ta sẽ cố gắn tìm các giá trị nghiệm cục bộ sau khi thêm mỗi một mô hình mới vào chuỗi mô hình với mong muốn dần đi đến nghiệm toàn cục.

### **ADABOOST**

### **GRADIENT BOOSTING**

Tham khảo tại: [Link1](https://en.wikipedia.org/wiki/Gradient_boosting)

“Khi gặp một bài toán bất kì, dù là phân lớp (classification) hay hồi quy (regression) thì việc chọn ra một mô hình đủ tốt luôn là một quyết định quan trọng và khó khăn nhất. Khác với Deep Learning, việc tìm ra mô hình tốt là việc cố gắng thay đổi số layer hay thay đổi cấu trúc mạng, ở Machine Learning, việc lựa chọn mô hình là việc tối ưu tham số, quan sát các đặc điểm về số chiều của không gian dữ liệu, đặt ra các giả thiết về phân phối dữ liệu, ...”

*Link2*

**Bias-variance trade-off[[2]](#footnote-2)**

Khác với Bagging là các model thực hiện song song và đọc lập, Boosting là theo phương pháp model sau thực hiện dựa trên kinh nghiệm của model trước. Để hạn chế được sai lầm từ các model trước, Boosting tiến hành đánh trọng số cho các mô hình mới được thêm vào dựa trên các cách tối ưu khác nhau. Tùy theo cách đánh trọng số (cách để các model được fit một cách tuần tự) và cách tổng hợp lại các model, từ đó hình thành nên 2 loại Boosting:

* Adaptive Boosting (AdaBoost)
* Gradient Boosting

Một số tính chất chung của Boosting là:

* Boosting là một quá trình tuần tự, không thể xử lí song song, do đó, thời gian train mô hình có thể tương đối **lâu**.
* Sau mỗi vòng lặp, Boosting có khả năng làm **giảm error theo cấp số nhân**.
* Boosting sẽ hoạt động tốt nếu base learner của nó không quá phức tạp cũng như error không thay đổi quá nhanh.
* Boosting giúp làm giảm giá trị bias cho các model base learner.

#### **XGBOOST**

XGBoost (Extreme Gradient Boosting) là một giải thuật được base trên gradient boosting, tuy nhiên kèm theo đó là những cải tiến to lớn về mặt tối ưu thuật toán, về sự kết hợp hoàn hảo giữa sức mạnh phần mềm và phần cứng, giúp đạt được những kết quả vượt trội cả về thời gian training cũng như bộ nhớ sử dụng.

#### **LightGMB**

## **STACKING**

Xây dựng một số model (thường là khác loại) và một meta model (supervisor model), train những model này độc lập, sau đó meta model sẽ học cách kết hợp kết quả dự báo của một số mô hình một cách tốt nhất.



# **PART 4: FEATURE ENGINEERING**

## ***Regularization (Chính quy hóa)***

Kỹ thuật làm giảm độ biến động, đồng thời giảm các vấn đề về overfitting.

**L1 and L2 Regularization**

L1, L2 là hai phương pháp thêm các thuật ngữ phạt vào hàm chi phí để ngăn chặn sự phức tạp của các mô hình:

* L1 regularization (Lasso regularization) dẫn đến các mô hình thưa (sparse) bằng cách thêm một hình phạt dựa trên giá trị tuyệt đối của các hệ số. Trong L1 regularization, một số hệ số của mô hình sẽ trở thành 0, điều này có thể dẫn đến loại bỏ một số đặc trưng không quan trọng từ mô hình.
* L2 regularization (Ridge regularization) khuyến khích việc sử dụng trọng số nhỏ hơn và phân phối đồng đều hơn bằng cách thêm một hình phạt dựa trên bình phương của các hệ số. Trong L2 regularization, các hệ số có xu hướng được giảm độ lớn và phân phối đều hơn, điều này giúp tránh được overfitting và cải thiện khả năng tổng quát hóa của mô hình.

Regulation R(f) được thêm vào công thức tính loss của một hàm như sau:

Trong đó,

* là giá trị của L1 regularization trong hàm loss.
* là hệ số regularization, điều chỉnh mức độ của regularization.
* W­i là trọng số (weights) của các đặc trưng (features).
* n là số lượng đặc trưng.

L1 regularization trong least squares[[3]](#footnote-3):

L2 regularization on least squares:

Khi λ tăng, tức là hệ số regularization tăng lên, điều gì thay đổi và cách mà loss và overfitting phản ứng sẽ phụ thuộc vào loại regularization được sử dụng:

**L1 Regularization (Lasso Regularization):**

* Khi λ tăng, L1 regularization sẽ tăng cường việc loại bỏ các trọng số không quan trọng bằng cách đặt chúng thành 0.
* Loss function sẽ tăng lên do việc thêm giá trị của L1 regularization vào.
* Overfitting có thể giảm do mô hình trở nên đơn giản hơn với số lượng đặc trưng ít hơn.

**L2 Regularization (Ridge Regularization):**

* Khi λ tăng, L2 regularization sẽ ưa thích các trọng số nhỏ hơn và phân phối đồng đều hơn.
* Loss function sẽ tăng lên do việc thêm giá trị của L2 regularization vào.
* Overfitting có thể giảm do các trọng số được kiểm soát chặt chẽ hơn, ngăn chặn mô hình phát triển quá phức tạp và dễ bị overfitting.

Tóm lại, khi λ tăng, cả hai loại regularization đều tăng cường việc kiểm soát mức độ phức tạp của mô hình, dẫn đến việc giảm overfitting. Tuy nhiên, việc tăng λ cũng sẽ làm tăng loss function do sự gia tăng của giá trị regularization. Điều quan trọng là phải cân nhắc và lựa chọn giá trị λ phù hợp để đạt được sự cân bằng giữa việc kiểm soát overfitting và giảm thiểu loss function.

## ***Gradient Descent***

Gradient Descent là kỷ thuật được sử dụng trong việc tìm điểm nhỏ nhất hoặc lớn nhất. Ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering.

Để giải bài toán local minimum, ta tìm nghiệm của phương trình đạo hàm bằng 0. Với global minimum, nếu may mắn tìm được toàn bộ (hữu hạn) các điểm cực tiểu, thì ta có thể thực hiện thay thế từng local minimum để tìm ra giá trị nhỏ nhất trong số chúng. Tuy nhiên từ sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là **xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0**. Gradient Descent (viết gọn là GD) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

Tham khảo tại: [Link](https://machinelearningcoban.com/2017/01/12/gradientdescent/)

1. Link 1: <https://anhhoangduc.com/ds-book-r/p03-05-bagging-boosting>

   Link 2: <https://github.com/chrisalbon/MachineLearningFlashcards.com>

   Link 3: https://viblo.asia/p/ensemble-learning-va-cac-bien-the-p1-WAyK80AkKxX [↑](#footnote-ref-1)
2. 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 [↑](#footnote-ref-2)
3. Least squares regression: mô hình tuyến tính sao cho tổng bình phương sai số giữa các giá trị dự đoán và giá trị thực tế là nhỏ nhất [↑](#footnote-ref-3)