TRƯỜNG ĐẠI HỌC

**SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**University of Technology and Education**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

NGÀNH CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

**Môn: TRÍ TUỆ NHÂN TẠO**

**Nhóm sinh viên thực hiện: Nhóm 7**

1. **Lâm Hoàng An \_18110073**
2. **Lê Công Nghĩa \_16110165**

**GVHD: Trần Nhật Quang**

TRƯỜNG ĐẠI HỌC

**SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**University of Technology and Education**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

NGÀNH CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

**Môn: TRÍ TUỆ NHÂN TẠO**

**Nhóm sinh viên thực hiện: Nhóm 7**

1. **Lâm Hoàng An \_18110073**
2. **Lê Công Nghĩa \_16110165**

**GVHD: Trần Nhật Quang**

Mục Lục

[1 Tìm hiểu về đạo văn 6](#_Toc60243292)

[1.1 Lời cam kết 6](#_Toc60243293)

[2 Giới thiệu thư viện SKLEARN 7](#_Toc60243294)

[2.1 Giới thiệu 7](#_Toc60243295)

[2.2 Tại sao sử dụng sklearn 7](#_Toc60243296)

[2.3 Cài đặt sklearn 7](#_Toc60243297)

[2.4 Ưu điểm của thư viện 7](#_Toc60243298)

[**3** Machine Learning 8](#_Toc60243299)

[3.1 Đôi nét về Machine Learning 8](#_Toc60243300)

[3.1.1 Machine Learning là gì ? 8](#_Toc60243301)

[3.1.1.1 Khái niệm 8](#_Toc60243302)

[3.1.2 Tại sao sử dụng Machine Learning 8](#_Toc60243303)

[3.1.3 Phân Loại 9](#_Toc60243304)

[3.1.3.1 Supervised Learning (Học có giám sát) 10](#_Toc60243305)

[3.1.3.2 Unsupervised Learning (Học không giám sát) 12](#_Toc60243306)

[3.1.3.3 Semisupervised learning (Học bán giám sát ) 13](#_Toc60243307)

[3.1.3.4 Online learning 13](#_Toc60243308)

[3.1.3.5 Batch learning 13](#_Toc60243309)

[3.1.3.6 Instance-based learning 14](#_Toc60243310)

[3.1.3.7 Model-based learning 14](#_Toc60243311)

[3.1.4 Những thách thức trong Machine Learning 14](#_Toc60243312)

[3.1.4.1 Bad Data 14](#_Toc60243313)

[3.1.4.2 Bad algorithms 15](#_Toc60243314)

[3.1.5 Testing and Validating 16](#_Toc60243315)

[3.1.5.1 Testing 16](#_Toc60243316)

[3.1.5.2 Validating 16](#_Toc60243317)

[3.1.6 End to end machine learning project 17](#_Toc60243318)

[3.1.6.1 Look at the big picture ( Quan sát một cách tổng quát về dự án) 17](#_Toc60243319)

[3.1.6.2 Get the data (Thu thập và xử lý dữ liệu). 17](#_Toc60243320)

[3.1.6.3 Discover and visualize the data to gain insights (Khám phá và trực quan hóa dữ liệu). 17](#_Toc60243321)

[3.1.6.4 Prepare the data for machine learning algorithms (Chuẩn bị dữ liệu cho thuật toán). 17](#_Toc60243322)

[3.1.6.5 Train and evaluate models (Chọn model). 18](#_Toc60243323)

[3.1.6.6 Fine-tune your models. 18](#_Toc60243324)

[3.1.6.7 Analyze and test your solution. 18](#_Toc60243325)

[3.2 Classification 19](#_Toc60243326)

[3.2.1 MNIST dataset 19](#_Toc60243327)

[3.2.2 Binary Classifier 19](#_Toc60243328)

[3.3 Tranning model 19](#_Toc60243329)

[3.3.1 Linear Regression (Hồi quy tuyến tính) 19](#_Toc60243330)

[3.3.1.1 The Normal Equation 20](#_Toc60243331)

[3.3.1.2 Computational Complexity 21](#_Toc60243332)

[3.3.2 Gradient Descent 21](#_Toc60243333)

[3.3.2.1 Batch Gradient Descent 24](#_Toc60243334)

[3.3.2.2 Stochastic Gradient Descent 25](#_Toc60243335)

[3.3.2.3 Mini – batch Gradient Descent 25](#_Toc60243336)

[3.3.3 Polynomial Regression 26](#_Toc60243337)

[3.3.4 Learning Curves 27](#_Toc60243338)

[3.3.5 Regularized Linear Models 29](#_Toc60243339)

[3.3.5.1 Ridge Regression 29](#_Toc60243340)

[3.3.5.2 Lasso Regression 30](#_Toc60243341)

[3.3.5.3 Elastic Net 31](#_Toc60243342)

[3.3.5.4 Early Stopping 32](#_Toc60243343)

[3.3.6 Logistic Regression 32](#_Toc60243344)

[3.3.6.1 Estimating Probabilities 32](#_Toc60243345)

[3.3.6.2 Training and Cost Function 33](#_Toc60243346)

[3.3.6.3 Decision Boundaries 34](#_Toc60243347)

[3.3.6.4 Sorfmax Regression 34](#_Toc60243348)

[3.4 Support Vector Machines 34](#_Toc60243349)

[3.4.1 Linear SVM Classification 35](#_Toc60243350)

[3.4.1.1 Hard Margin Classification 35](#_Toc60243351)

[3.4.1.2 Sort Margin Classification 36](#_Toc60243352)

[3.4.2 Nonlinear SVM Classification 37](#_Toc60243353)

[3.4.2.1 Polynomial Kernel 37](#_Toc60243354)

[3.4.2.2 Adding Similarity Features 38](#_Toc60243355)

[3.4.2.3 Gaussian RBF Kernel 38](#_Toc60243356)

[3.4.2.3.1 Data transformation 39](#_Toc60243357)

[3.4.2.3.2 Select the landmarks 39](#_Toc60243358)

[3.4.2.4 Computational Complexity 40](#_Toc60243359)

[3.4.3 SVM Regression 40](#_Toc60243360)

[3.4.4 Under the Hood 41](#_Toc60243361)

[3.4.4.1 Decision Function and Predictions 41](#_Toc60243362)

[3.4.4.2 Training Objective 42](#_Toc60243363)

[3.4.4.3 Quadratic Programming 43](#_Toc60243364)

[3.4.4.4 The Dual Problem 44](#_Toc60243365)

[3.4.4.5 Kernelized SVM 45](#_Toc60243366)

[3.4.4.6 Online SVMs 45](#_Toc60243367)

[3.5 Decision Trees 45](#_Toc60243368)

[3.5.1 Trainning and Visualizing a Descision Tree 46](#_Toc60243369)

[3.5.2 Making Predictions 46](#_Toc60243370)

[3.5.2.1 Estimating Class Probabilities 46](#_Toc60243371)

[3.5.3 The Cart trainning Algorithm 46](#_Toc60243372)

[3.5.4 Computational Complexity 47](#_Toc60243373)

[3.5.5 Gini Impurity or Entropy? 47](#_Toc60243374)

[3.5.6 Regularization Hyperparameters 48](#_Toc60243375)

[3.5.7 Regresssion 48](#_Toc60243376)

[3.5.8 Instability 49](#_Toc60243377)

[Ensemble Learning and Random Forests 49](#_Toc60243378)

[3.6 49](#_Toc60243379)

[3.6.1 Votting Classifiers 49](#_Toc60243380)

[3.6.2 Bagging and Pasting 51](#_Toc60243381)

[3.6.3 Random Forests 52](#_Toc60243382)

[3.6.3.1 Extra-Trees 52](#_Toc60243383)

[3.6.3.2 Feature Importance 52](#_Toc60243384)

[3.6.4 Boosting 52](#_Toc60243385)

[3.6.4.1 AdaBoost 52](#_Toc60243386)

[3.6.4.2 Gradient Boosting 53](#_Toc60243387)

# Tìm hiểu về đạo văn

Đạo văn không chỉ đơn thuần là việc copy thành quả của một ai đó hay vay mượn các ý tưởng gốc, mà còn rất nhiều biến hóa khác của đạo văn. Tại các quốc gia phát triển họ coi ý tưởng hay sản phẩm trí thức là một tài sản được Pháp luật bảo vệ nên có thể nói đạo văn là một " **tội** ".

Để hiểu rõ về đạo văn một cách rõ ràng hơn ta sẽ tìm hiểu những hình thức đạo văn khác nhau :

“The Ghost Writer”: Sử dụng toàn bộ công trình của người khác thành của mình

“The Photocopy”: Sao chép bố cục của các đoạn văn từ một nguồn duy nhất, không hề sửa đổi lại.

“The Potluck Paper”: Cố gắng “trá hình” việc đạo văn của mình bằng cách sao chép từ nhiều nguồn khác nhau, biên tập lại các câu sao cho nội dung thật hợp lí.

“The Poor Disguise”: Giữ lại các nội dung quan trọng của nguồn, nhưng người đó vẫn sửa lại một chút về “diện mạo” của bài viết đó bằng cách thay đổi từ khóa hay câu cú.

“The Labor of Laziness”: Chú giải các nguồn khác nhau và nối chúng lại với nhau, thay vì dành nỗ lực tương tự cho công việc của mình.

“The Self-Stealer”: Sao chép đáng kể các thành quả trước đó của chính mình để phục vụ cho bài viết/nghiên cứu mới.

Vậy để tránh việc đạo văn ta có những cách sau :

Tìm hiểu kỹ về vấn đề mà bạn đang muốn nói tới

Diễn đạt lại nhiều lần bằng các cách khác nhau.

Nắm được những gì không cần trích dẫn.

Nêu tên tác giả của ý tưởng đó.

Trích dẫn nguồn bất cứ khi nào sử dụng lời trích, chú giải một cách chi tiết và cụ thể.

## Lời cam kết

“Chúng tôi xin cam đoan đồ án này do chính chúng tôi thực hiện. Chúng tôi không sao chép, sử dụng bất kỳ tài liệu, mã nguồn của người khác mà không ghi rõ nguồn gốc. Chúng tôi xin chịu hoàn toàn trách nhiệm nếu vi phạm.”

# Giới thiệu thư viện SKLEARN

## Giới thiệu

Scikit-learn (sklearn) là thư viện mã nguồn mở về machine learning phổ biến nhất trong cộng đồng Python

Sklearn chứa các thuật toán machine learning từ cơ bản cho đến phức tạp , ta chỉ cần khai báo vừa đưa dữ liệu và train thì sẽ có kết quả của thuật toán không cần cài đặt thuật toán một cách thủ công.

## Tại sao sử dụng sklearn

Độ chính xác thuật toán của Sklearn đem lại.

Sklearn hỗ trợ các thuật toán và doc của sklearn đầy đủ dễ hiểu để ta có thể sử dụng các thuật toán một các dễ dàng ngắn ngọn hiểu quả cao.

Scikit-learn còn là một trong những lựa chọn hàng đầu của các researchers và deverlopers.

Trang chủ của thư viện: <http://scikit-learn.org/>

## Cài đặt sklearn

Để phục vụ trong quá trình học hoặc ta làm việc vs sklearn ta cần cài những thư viện liên quan và đặc biệt là ngôn ngữ sử dụng là Python.

B1: Trước tiên máy các bạn phải được cài đặt Python ( phiên bản mình đang sử dụng là Python 3.7 ) nếu máy tính các bạn đã có Python có thể bỏ qua bước này.

B2: Để cài đủ thư viện sklearn ta cần sử dụng lệnh này:

**$ pip install numpy scipy matplotlib scikit-learn pandas**

## Ưu điểm của thư viện

Hàm ngắn ngọn, khai báo dễ dàng, dễ sử dụng.

Xử lý được lượng dữ liệu lớn.

Độ chính xác và độ tin cậy cao do cộng đồng sử dụng nhiều.

Tốc độ tính toán nhanh.

Hỗ trợ nhiều các thuật toán Machine Learning đơn giản và phức tạp.

# Machine Learning

## Đôi nét về Machine Learning

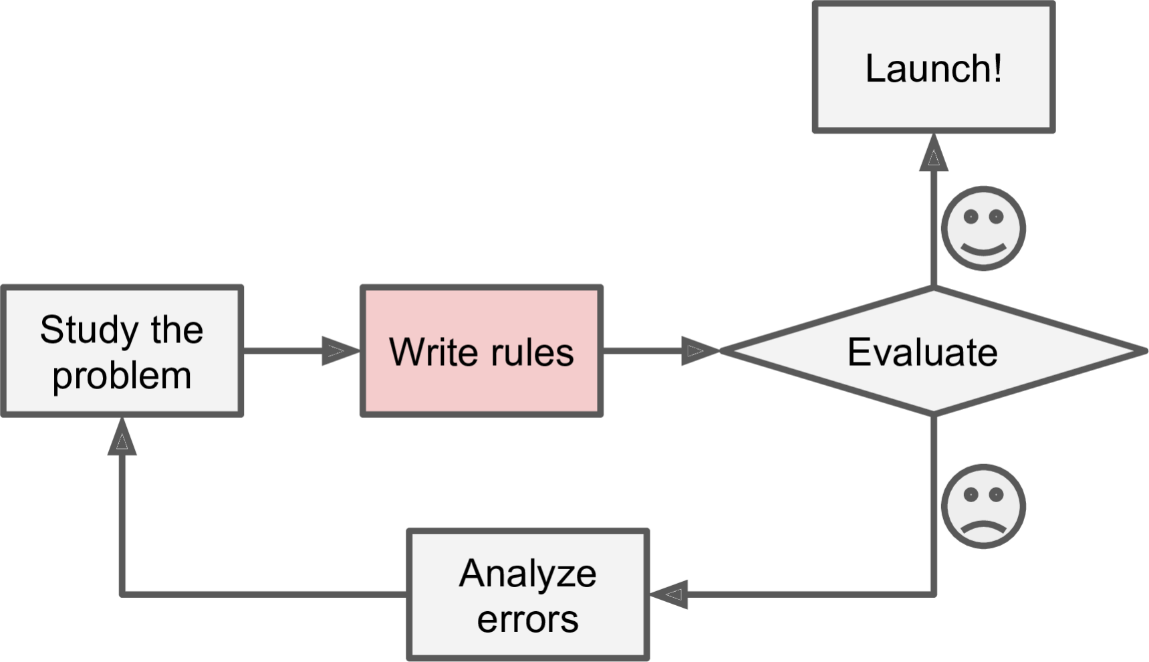
### Machine Learning là gì ?

#### Khái niệm

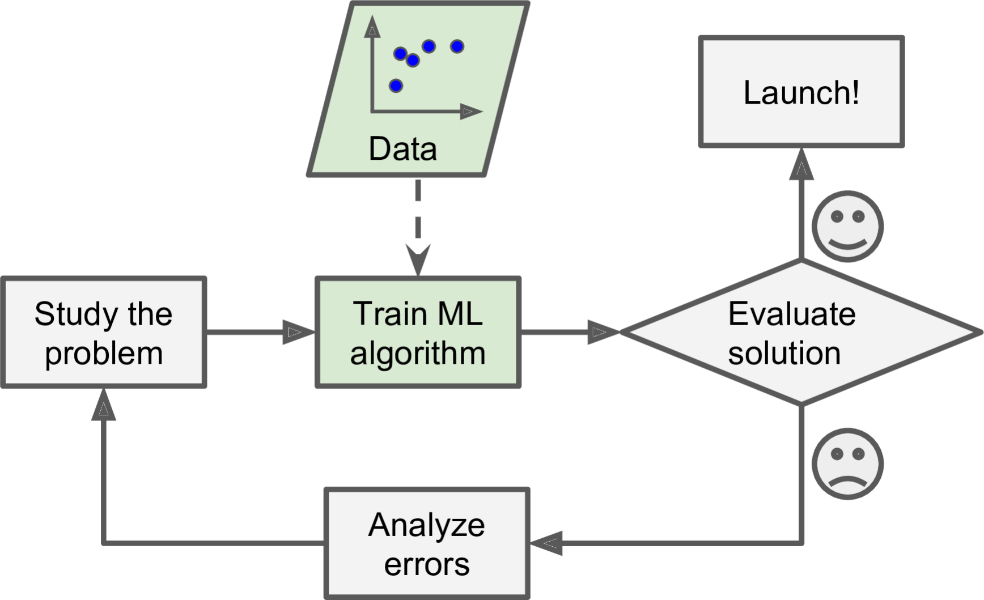
Theo Arthur Samuel đưa ra năm 1959 Maching learning là một ngành học thuộc khoa học máy tính, giúp máy tính có khả năng tự học mà không phải lập trình một cách rõ ràng

Theo Tom Mitchell Tom Mitchell coi Machine Learning như 1 chương trình, nhiệm vụ của nó là thưc hiện 1 task T nào đó, khi thực hiện xong, ta thu được experience E. Nhờ vào việc học hỏi experience E, ta có thể thay đổi (hoặc không) để tiến tới thực hiện task T+1, và nhằm cải thiện hiệu suất P

### Tại sao sử dụng Machine Learning

**Lập trình truyền thống**

**Lập trình có Machine Learning**



Lập trình truyền thống: Mình cần phải Tự học rất các vấn đề và viết ra các luật cho máy tính khi chạy chương trình mình có thể sẽ sinh ra lỗi vì mình sẽ thiếu một số trường hợp máy tính không xử lí được và mình tiếp tục tự học vấn đề và thử tiếp. Còn lập trình có xử dụng Machine Learning thì mình sẽ đưa ra vấn đề cho máy tính mà máy tính tự train cho máy tính và xử lý đưa ra kết quả.

Machine Learning có sự gắn bó chặt chẽ với khá nhiều ngành khác:

Big Data, AI, Statistics Learnig

Ứng dụng sâu rộng vào cuộc sống hàng ngày: trí tuệ nhân tạo AlphaGo, nhận diện khuôn mặt, gợi ý bạn bè từ faceboook, phân loại spam email từ google mail, chuẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, dự đoán kết quả trận đấu, nhận dạng giọng nói, phân loại các chuẩn DNA

### Phân Loại

Machine learning Algorithm được chia làm 2 loại chính là: Supervised Learning (Học có giám sát) và Unsupervised Learning (Học không giám sát).

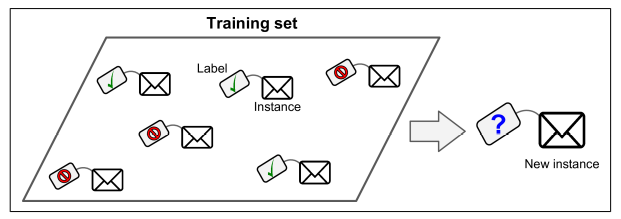
Ngoài ra còn 1 vài loại khác như SemiSupervised Learning, Reinforcement Learning, Learning to Learn, Developmental Learning, ...

#### Supervised Learning (Học có giám sát)

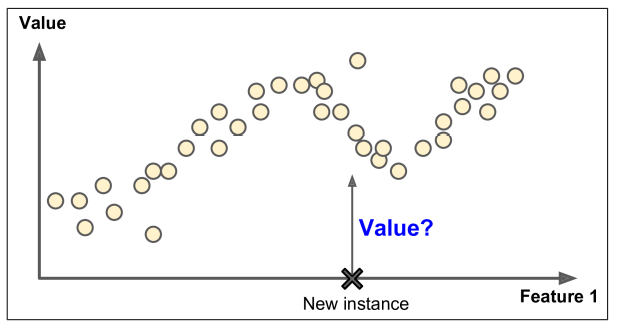
Supervised learning là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước.

Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính:

**Classification (Phân loại)**

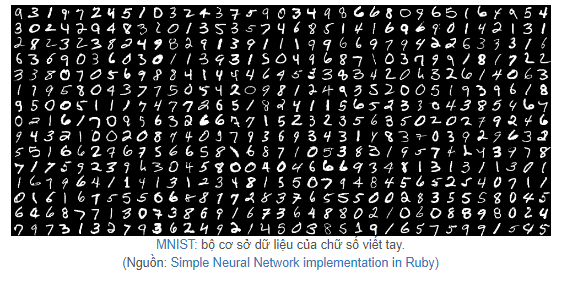
Một bài toán được gọi là classification nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không

**Regression (Hồi quy)**

Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng x m2x m2, có yy phòng ngủ và cách trung tâm thành phố z kmz km sẽ có giá là bao nhiêu?

Gần đây [**Microsoft có một ứng dụng dự đoán giới tính và tuổi dựa trên khuôn mặt**](https://how-old.net/). Phần dự đoán giới tính có thể coi là thuật toán **Classification**, phần dự đoán tuổi có thể coi là thuật toán **Regression**.

**Chú ý rằng phần dự đoán tuổi cũng có thể coi là Classification nếu ta coi tuổi là một số nguyên dương không lớn hơn 150, chúng ta sẽ có 150 class (lớp) khác nhau.**

VD: Trong nhận dạng chữ viết tay, ta có ảnh của hàng nghìn ví dụ của mỗi chữ số được viết bởi nhiều người khác nhau. Chúng ta đưa các bức ảnh này vào trong một thuật toán và chỉ cho nó biết mỗi bức ảnh tương ứng với chữ số nào. Sau khi thuật toán tạo ra một mô hình, tức một hàm số mà đầu vào là một bức ảnh và đầu ra là một chữ số, khi nhận được một bức ảnh mới mà mô hình **chưa nhìn thấy bao giờ**, nó sẽ dự đoán bức ảnh đó chứa chữ số nào.

Một số thuật toán phổ biến trong Supervised Learning

k-Nearest Neighbors

Linear Regression

Logistic Regression

Support Vector Machines (SVMs)

Decision Trees and Random Forests

#### Unsupervised Learning (Học không giám sát)

Trong thuật toán này, chúng ta không biết được outcome hay nhãn mà chỉ có dữ liệu đầu vào. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Một cách toán học, Unsupervised learning là khi chúng ta chỉ có dữ liệu vào XX mà không biết nhãn YY tương ứng.

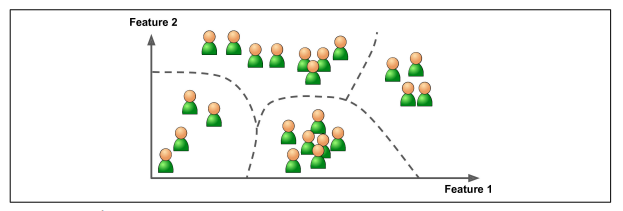
Những thuật toán loại này được gọi là Unsupervised learning vì không giống như Supervised learning, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào.

Unsupervised learning được tiếp tục chia nhỏ thành hai loại:

Clustering (phân nhóm)

Một bài toán phân nhóm toàn bộ dữ liệu X thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm

VD: tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.



Association

Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước.

Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm

#### Semisupervised learning (Học bán giám sát )

Các bài toán khi chúng ta có một lượng lớn dữ liệu XX nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên.

#### Online learning

Online learning là: Hệ thống có khả năng học hỏi tưng bước bằng cách cho nó dữ liệu tuần tự. Mỗi bước học nhanh chóng và hệ thống có thể tìm hiểu về dữ liệu mới. Máy sẽ học rất nhanh những dữ liệu đó và giải phóng dữ liệu để tránh tiêu tốn tài nguyên. Thích hợp với những hệ thống có nguồn tài nguyên hạn chế và dữ liệu liên tục cập nhật và đổi mới.

VD: Chương trình đánh giá “Giá cổ phiếu”.

#### Batch learning

Batch learning là: Hệ thống không có khả năng học hỏi từng bước. Đưa dữ liệu đầu vào (data training ) máy sẽ học tất cả dữ liệu và được vận dụng giải quyết các bài toán theo dữ liệu có sẵn. Muốn xử lý dữ liệu mới thì máy tính học lại tập dữ liệu cũ và thêm dữ liệu mới sau đó vận hành lại, quá trình này sẽ mất rất nhiều thời gian và tài nguyên máy tính.

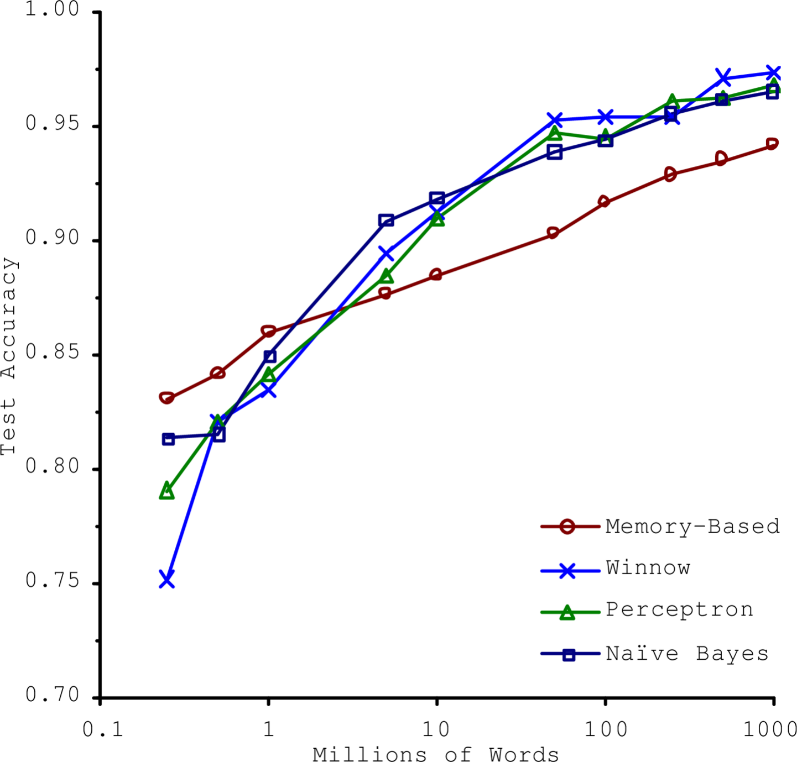
VD: Trò chơi game cờ vua

#### Instance-based learning

#### Model-based learning

### Những thách thức trong Machine Learning

#### Bad Data

**Insufficient Quantity of Training Data**

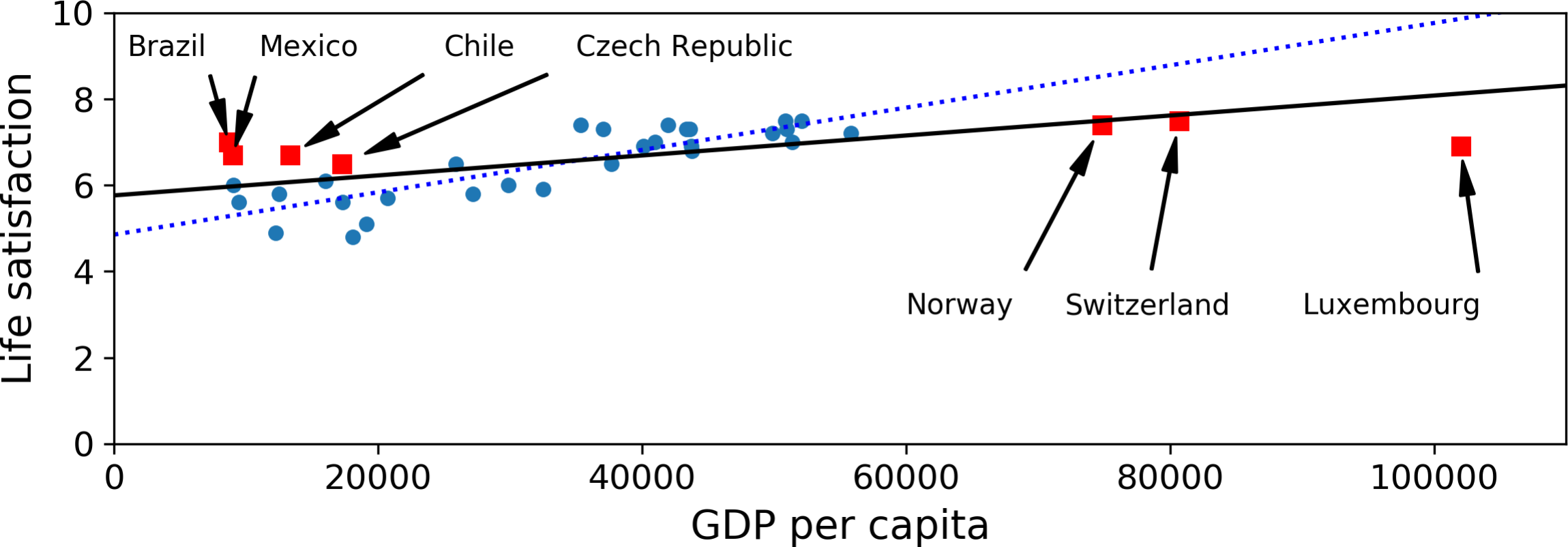
Thu thập training data quá ít không đủ để cho máy học và sẽ có một số trường hợp hay một dữ liệu nào đó mà máy sẽ đánh giá và đưa ra kết quả không được tốt hoặc có thể không xử lý được

**Nonrepresentative Training Data**

Dữ liệu không bị nhiễu



Dữ liệu bị nhiễu dư thừa



Muốn thu thập dữ liệu về một số người mắc covid mà ta chỉ thu thập dữ liệu số người mắc bệnh ở các quốc gia như Trung Quốc, Mỹ thì máy sẽ không thể đánh giá mức độ mắc bệnh của một quốc gia khác như Việt Nam.

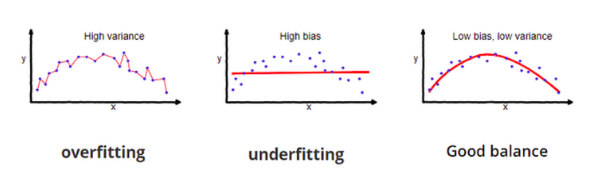
**Missing Data**

Muốn thu thập dữ liệu thống kê giá nhà ở Việt Nam. Nhưng chúng ta thu thập lại thiếu các cột dữ liệu như (số phòng, độ an ninh khu vực, …) thì máy sẽ không thể đưa ra đánh giá chính xác về giá nhà ở khu An Ninh hay nhà có nhiều phòng.

**Irrelevant Features**

Thu thập nhiều thông tin không cần thiết. Muốn thu thập dữ liệu chiều cao mà thu thập them các dữ liệu về nơi ở, tôn giáo, dân tộc, …

#### Bad algorithms



**Overfitting**

Overfitting là mô hình quá khít với dữ liệu, nó sẽ đúng trên tập training nhưng trên tập test (kiểm thử ) thì kết quả rất tệ. Mô hình này thường có bias nhỏ và variance lớn là mô hình quá khít với dữ liệu, nó sẽ đúng trên tập training nhưng trên tập test (kiểm thử ) thì kết quả rất tệ. Mô hình này thường có bias nhỏ và variance lớn.

Giải pháp: Thu thập thêm dữ liệu để tăng độ đa dạng, phong phú của dữ liệu. Dừng train model phù hợp để tránh overfitting để đạt được model tốt nhất (sẽ được đề cập các cách tránh overfitting trong các phần sau Gather more data, Simple model, Use Regularization,Early stoping ).

**Underfitting**

Underfitting là mô hình mà high bias và variance low. Hiện tượng này xảy ra khi lượng dữ liệu quá ít hoặc ta cố gắng mô tả các dữ liệu phức tạp bằng các mô hình tuyến tính đơn giản như hồi quy.

Giải pháp: Thu thập thêm dữ liệu và tăng độ phức tạp của model.

### Testing and Validating

#### Testing

Test set là một tập dữ liệu dùng để kiểm tra độ chính xác của model được train từ training data.

Test set là bộ dữ liệu mới không có nằm trong training set máy phải sử dụng bộ data training để học mà xử lý những dữ liệu mới chưa được train để giải quyết vấn để nhằm đánh giá khách quan tăng độ chính xác cao. Còn training set là tập dữ liệu được đưa vào để huấn luyện cho máy khi đó nếu ta đưa dữ liệu này vào thì máy sẽ đánh giá đưa ra kết quả tốt và mình không biết dữ liệu khác thì model đó có đánh giá có tốt hay không nên training set không dùng để training test để kiểm tra thử tính hiểu quả hay tính chính xác của model đem lại.

#### Validating

Hyperprameter tuning là lựa chọn một tập hợp training set tối ưu cho máy học. Một Hyperprameter là một tham số có giá trị được thiết lập trước khi quá trình học tập và bắt đầu.

Validation set là một tập dữ liệu được trích ra từ tập training set để đánh giá mức độ chính xác của model. Giải pháp này thường hoạt động khá tốt. Tuy nhiên nếu bộ dữ liệu test quá nhỏ thì sẽ đánh giá mô hình không được chính xác và ngược lại nếu bộ dữ liệu test quá lớn thì bộ training còn lại ít sẽ dẫn đễn hiệu quả của máy học.

Ta cần phải dùng validation set vì nếu chỉ cho máy học trên training set và đưa ra hoạt động thì chắc chắn model đó sẽ hoạt động không như ý mình mong muốn nhằm tăng độ chính xác của model chúng ta cần phải test model trước khi sử dụng. Nếu bạn chỉ học lý thuyết mà không thực hành thì cũng không thể đánh giá được năng lực của bạn và máy cũng thế. Nhằm mục đích cho model hoạt động tốt nhất thì phải cần dùng validation set để kiểm chứng lại quá trình model học có chính xác hay không. Từ đó có thể điều chỉnh lại tập hợp training set hoặc xử lý lại model bằng thuật toán khác nhằm mục đích giúp cho model hoạt động hiệu quả nhất có thể.

### End to end machine learning project

#### Look at the big picture ( Quan sát một cách tổng quát về dự án)

* Cần xem mục tiêu dự án là gì?
* Độ chính xác như thế nào?
* Xác nhận lại yêu cầu vì khi xác nhận đúng yêu cầu ta mới có hướng đi rõ ràng.
* Chọn một model thích hợp và thước đo đánh giá dự án.
* Chọn dữ liệu phù hợp với dự án để đạt được kết quả cao.
* Đánh giá model bằng các công cụ tính toán và chạy thử nghiệm.

#### Get the data (Thu thập và xử lý dữ liệu).

* Phân tích những dữ liệu cần cho dự án.
* Thu thập những dữ liệu phù hợp với nhu cầu của đồ án.
* Tiền xử lý giữ liệu nắm rõ những dữ liệu đang có vd: số lượng feature, số lượng sample để thuận tiện cho việc xử lý dữ liệu
* Vẽ biểu đồ đánh giá độ tương quan mật thiết của dữ liệu.
* Chia dữ liệu ra tập train và tập test để còn kiểm tra model.

#### Discover and visualize the data to gain insights (Khám phá và trực quan hóa dữ liệu).

* Sau khi vẽ biểu đồ trực quan về dữ liệu chúng ta có thể loại bỏ những dữ liệu không thích hợp trong dự án
* Hoặc có thể kết hợp các feature lại để tạo ra feature mới có thể tăng độ chính xác của dự đoán
* Xử lý dữ liệu thật sạch thì việc đánh giá của model sẽ được hiểu quả cao

#### Prepare the data for machine learning algorithms (Chuẩn bị dữ liệu cho thuật toán).

* Làm sạch dữ liệu:
  + Hầu hết trong Machine Learning không thể xử lý được feature có dữ liệu thiếu hoặc dữ liệu có giá trị chuỗi ( Vì vậy ta cần điền vào những dữ liệu bị thiếu ). Cách xử lý:
    - Loại bỏ những dữ liệu không cần thiết.
    - Điền giá trị vào những ô bị khuyết dữ liệu.
  + Loại bỏ những dữ liệu văn bản vì hầu hết machine learning không thể xử lý được.
* features scaling lại về khoảng cho tiện trong việc tính toán và tăng tốc độ xử lý của model

#### Train and evaluate models (Chọn model).

* Sau khi định hình được vấn đề, thu thập và sử lý dữ liệu thì ta chọn một model thích hợp để huấn luyện model.
* Đánh giá model bằng cách sử dụng Cross-Validation với cách này ta chia nhỏ tập training set của chúng ta thành nhiều tập nhỏ hơn và tập validation set bằng K-fold cross-validation của python.

#### Fine-tune your models.

* Ta có thể dùng một số cách như grid search hoặc randomized search để dò tìm ra các siêu tham số thích hợp nhất cho model của chúng ta (những cách này python đã cung cấp sẵn các hàm xử lý hết sức tiện lợi cho ta dùng)
* Các phương thức kết hợp: Một cách khác để tinh chỉnh model là cố gắng kết hợp các mô hình hoạt động tốt. Bởi vì một nhóm sẽ thường hoạt động tốt hơn một cá thể tốt thất.
* Phân tích model và lỗi của chúng: tính điểm từng thuộc tính ảnh hưởng đến model ta có thể thêm feature hoặc giảm bớt feature không cần thiết.
* Đánh giá lại model bằng tập test set: sau khi xử lý xong ta nên dùng tập test set để đánh giá lại thuật toán ta có hoạt động tốt trên những dữ liệu chưa gặp lần nào? Nếu model hoặc động không tốt ta nên xem lại dữ liệu hoặc model ta train không phù hợp với dữ liệu này. Nếu model tốt phù hợp tiêu chí của dự án thì ta chuyển sang bước 7

#### Analyze and test your solution.

* Khởi chạy, giám sát và bảo trì hệ thống: Sau khi đã hoàn thành các bước trên thì cơ bản model của chúng ta đã hoàn thành, cho model chạy trên nguồn dữ liệu thực tế. Kiểm tra xem model có gặp những lỗi sai gì không để xử lý kịp thời. Bảo trì thường niên hệ thống để có thể giúp hệ thống hoạt động tốt trên nguồn dữ liệu biến đổi liên tục từng ngày

## Classification

### MNIST dataset

Là một bộ dữ liệu gồm 70000 hình ảnh nhỏ gồm các chữ số (0-9) được viết tay bởi học sinh trung học và nhân viên của Cục điều tra dân số Hoa Kỳ.

Label của những tấm ảnh trên là những con số tương ứng trong hình.

### Binary Classifier

Binary Classifier giúp cho chúng ta phân biệt được kết quả giữa 2 class. Kết quả đầu ra (output) là class.

### Performance meansure

DumpClassifier bất cứ giá trị nào đưa vào thuật toán này đều trả ra giá trị dự đoán là False nhưng độ chính xác của nó trên tập dữ liệu mnist thì cực kỳ cao >90%. Tại sao bất kỳ dữ liệu nào đưa vào nó đều đưa kết quả là False mà độ chính xác lại cao như vậy? Bởi vì tập dữ liệu này bị mất cân bằng. Tập mnist của ta có 10 class nhưng ta lại biến nó thành binary class (2 class là 5 và not \_5) số lượng sample của class 5 chỉ chiếm 1/10 trong khi class không phải \_5 thì nó lại 9/10 nên DumpClassifier xác suất đưa ra dự đoán > 90%.

Confusion matrix: ma trận này thể hiện dự đoán đúng và sai của thuật toán chúng ta huấn luyện.

0 1

0 [[53892 687]

1 [ 1891 3530]]

* 0-0: là không phải 5 và máy đoán không phải 5
* 1-1: là 5 và máy đoán là 5
* 0-1: là không phải 5 nhưng máy dự đoán là 5
* 1-0: là 5 nhưng máy dự đoán không phải 5

[[54579 0 ]

[ 0 5421]]

Matrix phía trên là một matrix lý tưởng trong machine leanring dự đoán đúng với mọi sample.

Precision và recall: dựa vào confusion matrix phía trên:

* [0,0] là True Negative(TN)
* [0,1] là False Positive(FP),
* [1,0] là False Negative(FN),
* [1,1] True Positive (TP)

Ta tính precision và recall bằng công thức:

Precision = TP/(TP+FP)

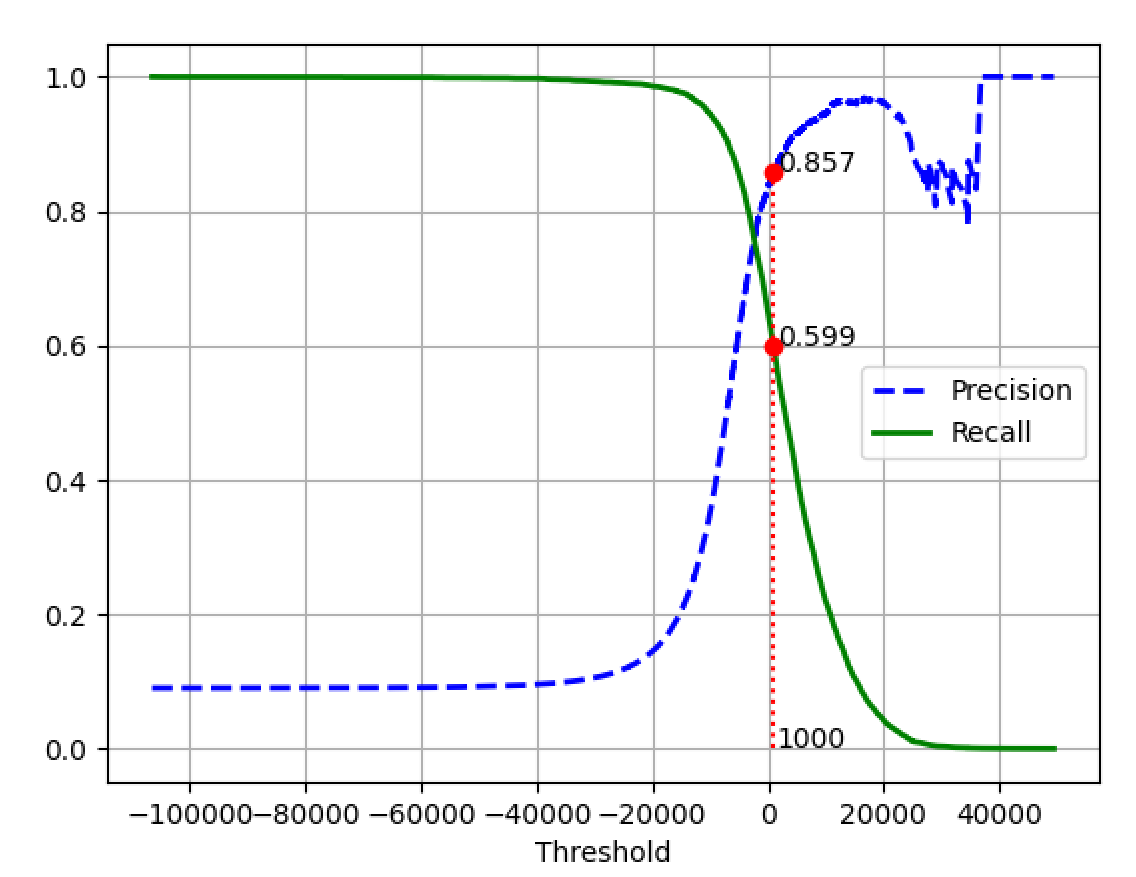
Recall = TP/(TP+FN)

* Precision cho chúng ta thấy độ chính xác của thuật toán (độ chính xác của kết quả do thuật toán dự đoán ra)
* Recall cho chúng ta thấy độ đầy đủ của dự đoán của thuật toán (Cho chúng ta trong tổng số các dự đoán đúng chúng ta chỉ ra được bao nhiêu trường hợp đúng)

Precision and recall tradeoff:

* Một model tốt nhất khi cả precision và recall đều đạt giá trị lớn.
* Nhưng trong thực tế ta khó có thể làm cho cả 2 đều lớn được vì vậy mà tùy thuộc vào yêu cầu thực tế của thuật toán mà ta buộc phải đánh đổi mất cái này để được cái kia nhưng vẫn cố gắng giữ 2 giá trị này cùng cao nhất có thể.

Ví dụ trong bài toán ta đang làm thì ta có biểu đồ như sau:



*Ảnh 7 precision & recall theo threshold*

Nhìn theo hình trên ta chọn threshold = 1000 vì precision và recall điều đạt được giá trị cao.

F1 score: cho ta thấy giá trị trung bình độ hài hòa của precision và recall.

Được tính theo công thức:

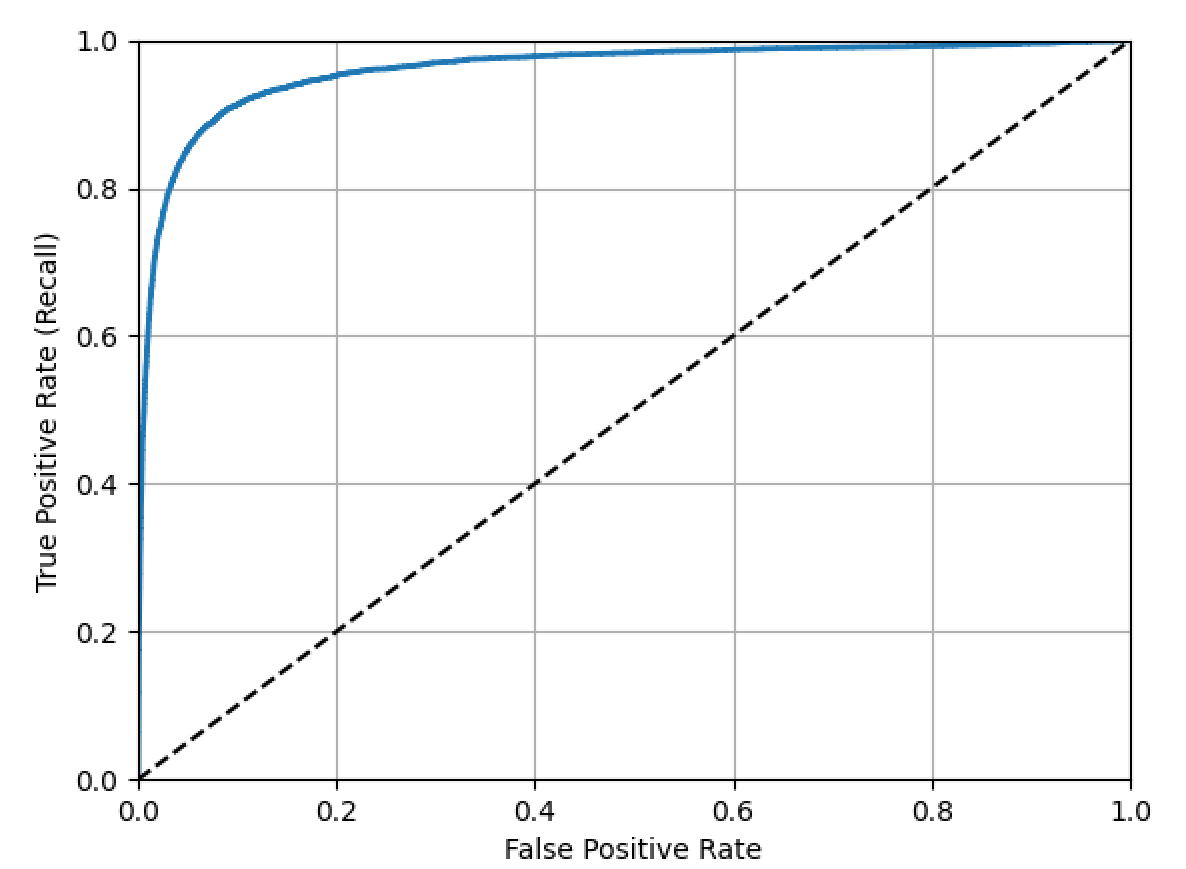
F1=2 x Precision x Recall / (Precision + Recall)

Giá trị của F1 nằm trong [0 - 1]

F1 đạt giá trị tốt nhất khi và chỉ khi F1 = 1 tức là Precision và Recall điều lần lượt =1

F1 có giá trị tệ nhất khi và chỉ khi F = 0 tức là Precision = 0 hoặc Recall = 0 với điều kiện biểu thức hợp lệ là (Precision + Recall) != 0

ROC-curve (Receiver operating characteristic curve):



*Ảnh 8 ROC curve*

ROC curve là tỉ lệ True Positive Rate (Recall) / False Positive Rate (số lượng FP/(TN+FP)).

FPR là tỉ lệ của các trường hợp Negative được phân loại nhầm thành Positive.

TPR càng cao thì bộ phân loại càng tạo ra nhiều FPR.

Bộ phân loại càng tốt nếu ROC-curve càng đi cao về góc trên bên trái của biểu đồ.

### Multiclass classfication

Ở trên ta đã tìm hiểu binary classification là chúng ta phân loại sample trên 2 lớp. Thì Multiclass classfication giúp cho ta phân loại > 2 lớp.

Có một số thuật toán giúp ta phân loại Multiclass ví dụ như: RandomForest, KNN, những lại có một số thuật toán thì không có thể giúp cho ta.

Những thuật toán không hỗ trợ chúng ta phân loại Multiclass classfication thì chúng ta phải làm sao? Chúng ta có thể áp dụng các kỹ thuật bên dưới để giúp tăng sức mạnh cho các thuật toán không thể phân loại Multiclass classfication có thể phân loại Multiclass classfication:

* One Vs All: nếu ta có N classes cần phân loại thì nó sẽ tạo ra N binary classifier, mỗi lớp mới đó sẽ phân loại một sample có thuộc lớp A hay not A. Khi dự đoán thì mỗi lớp binary sẽ cho điểm và ta xem cái nào điểm cao nhất thì sample này thuộc về lớp đó.
* One Vs One: nếu ta có N classes cần phân loại thì nó sẽ tạo N\*(N-1)/2 binary classifier, mỗi lớp mới đó sẽ phân loại một sample có thuộc lớp A hay một lớp còn lại cứ thể mà tiếp tục đến hết các lớp. Ở đây nó bắt cặp 2 lớp trong số N lớp ra để làm classification. Khi dự đoán thì lớp nào được chọn nhiều hơn thì cuối cùng sample sẽ thuộc về lớp đó.
* OvO trở nên xịn khi ta train một tập dữ liệu lớn mà ít class do đặc điểm của OvO là nếu có 3 lớp A, B, C thì binary classifier của nó là A vs B, A vs C, B vs C. Nghĩa là ở đây mỗi lần huấn luyện với classifier nào nó chỉ cần load 1 phần của dữ liệu vào thôi làm nó nhanh hơn so với OvA phải load hết cả bộ vào. Và mỗi binary classifier được tạo ra từ OvO là độc lập nên ta có thể cùng train một lúc nhiều cái.

### Đánh giá lỗi

Đánh giá độ chính xác khi dùng SGD và RandomForest cho multiclass:

* Quan sát độ chính xác của 2 phương pháp:



*accuracy khi chưa scale feature*

Ta có thể thấy độ chính xác của RandomForest lớn hơn rất nhiều so với SGD

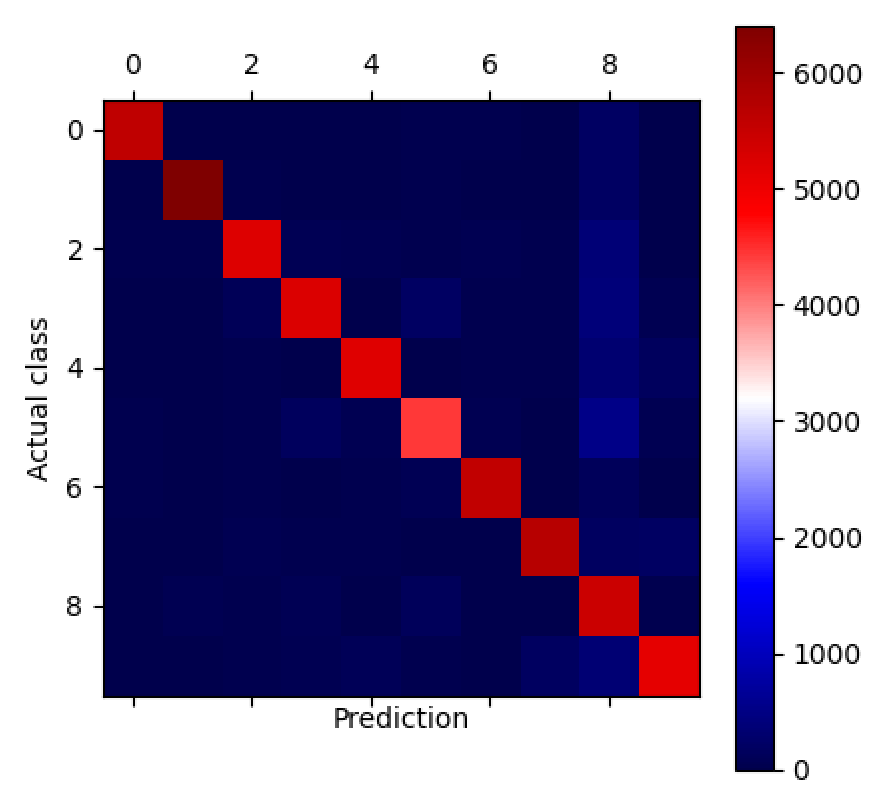
Scale các features và đánh giá lại các phương pháp phân loại. Sau khi scale ta có được độ chính xác của 2 phương pháp trên như sau:



*accuracy khi đã scale feature*

Ta có thể thấy được độ chính xác của phương pháp SGD được cải thiện lên rất nhiều, còn RandomForest thì không được cải thiện nhiều tại vì sao thì chúng ta sẽ tìm hiểu ở các chương sao.

Phân tích các lỗi của model chúng ta vừa xử lý ra được:



*Số sample được predict đúng và sai ở từng class*

Máy dễ đoán sai ở một số trường hợp: ví dụ như số 3 nhưng lại nhầm thành 5, 6 nhưng lại nhầm thành 9,…

Khác phục các dự đoán sai ta có thể:

* + Thu thập thêm dữ liệu huấn luyện
  + Đếm số lượng vòng lặp đã đóng
  + Xử lý hình ảnh trước khi cho máy học.

### Multilabel classification

Đối với một bài toán thông thường thì dự đoán của một sample là một label phù hợp với sample đó. Nhưng đối với mutilable thì kết quả dự đoán lại là một vector label mang giá trị tương ứng là vector binary. Hay còn gọi các khác cho dễ hiểu là kết quả dự đoán sẽ chứa các label dự đoán kết quả.

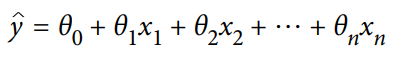
VD: một mutilabel có điều kiện > 5 và là số lẻ khi ta nhập 4 thì vector label là [0,0], 7 thì [1,1],……

### Multioutput classification

Cũng tương tự như multilabel nhưng đối với multilabel thì kết quả trả ra của thuật toán là một binary vector thì multioutput kết quá trả ra là một multiclass vector. Có nghĩa là kết quả nằm trong vector trả ra không phải là những số 0 1 (true, false) mà là một vector chứ các giá trị tùy thuộc vào giá trị cần trả ra.

## Tranning model

### Linear Regression (Hồi quy tuyến tính)

**Hypothesis function of LR:** là một đường dùng để mô tả dữ liệu

Trong đó :

y là label (outcome)

n là số lượng feature

x là feature

ϴj là model parameter

ϴ0 còn được gọi là bias độ lệch với góc tọa độ

**Cost function:** Mean Squared Error (MSE)

Tìm ra bộ parameter có giá trị cost function MSE nhỏ nhất.

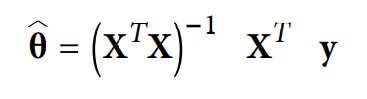
Trong đó :

m là số lượng sample

**Chú ý:**

* y là giá trị thực của outcome, trong khi đó ŷ là giá trị dự đoán của Linear Regression. Kết quả đầu ra mong muốn rằng giá trị của y và ŷ có độ chênh lệch là nhỏ nhất.
* Linear hay tuyến tính hiểu một cách đơn giản là thẳng hoặc phẳng
  + Trong không gian 2 chiều Linear có dạng là 1 đường thẳng
  + Trong không gian 3 chiều Linear có dạng là 1 mặt phẳng
  + Trong không gian nhiều hơn 3 chiều khái niệm mặt phẳng không còn phù hợp nữa thay vào đó khái niệm siêu mặt phẳng ra đời (hyperplane).

#### The Normal Equation

Tìm ra giá trị minimizes của cost function cho theta. Bằng phương pháp closed – form solution đưa kết quả dựa trên công thức bên dưới được gọi là The Normal Equation.

Trong đó:

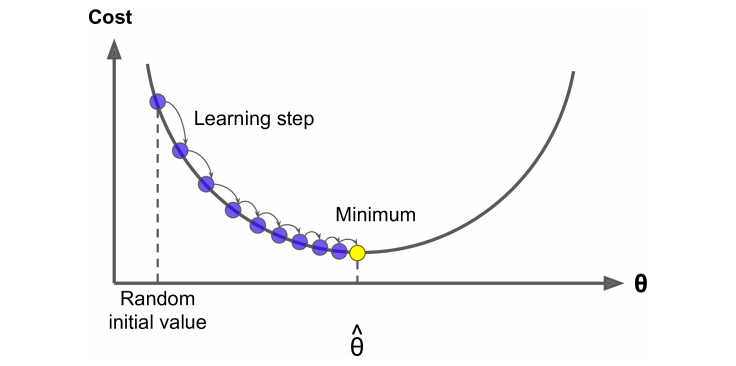
Theta mũ là giá trị tối thiểu của theta của hàm cost function.

y là một vector chứa giá trị từ y1 cho đến ym.

#### Computational Complexity

Do The Normal Equation tính toán phức tạp (XT X)-1 trong việc tìm ma trận nghịch đảo. Khi chạy với số lượng feature lớn thì chương trình sẽ chạy khá chậm vì tốn nhiều thời gian tìm ma trận nghịch đảo. Vì vậy chúng ta sẽ xem xét cách khác để đào tạo một Linear tốt hơn phù hợp cho các trường hợp và phù hợp với bộ nhớ. Tiếp theo ở phần sau.

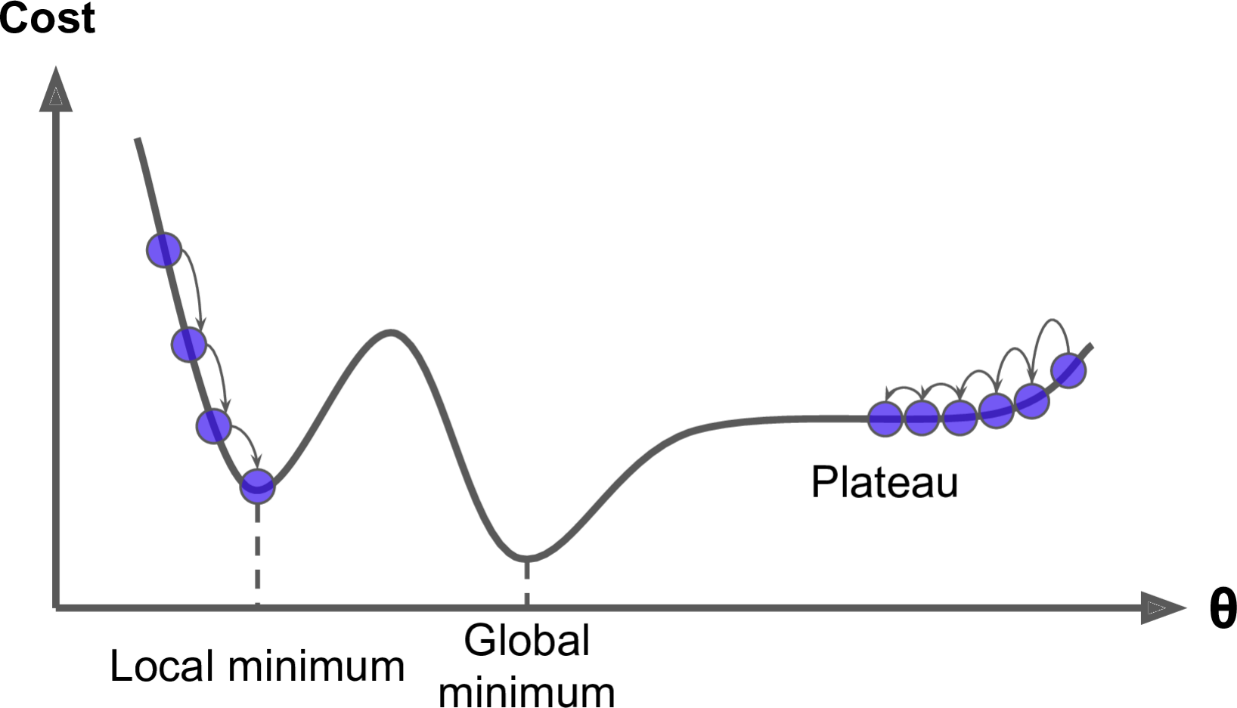
### Gradient Descent

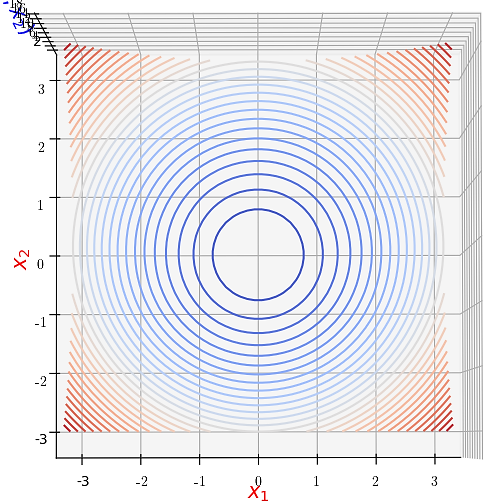
Gradient Descent là một thuật toán tối ưu hóa khả năng tìm ra giải pháp nhiều vấn đề. Ý tưởng chung của Gradient Descent là tinh chỉnh các tham số lặp để giảm tối thiểu hàm chi phí bằng learning step.

Phù hợp cho tập dữ liệu lớn.

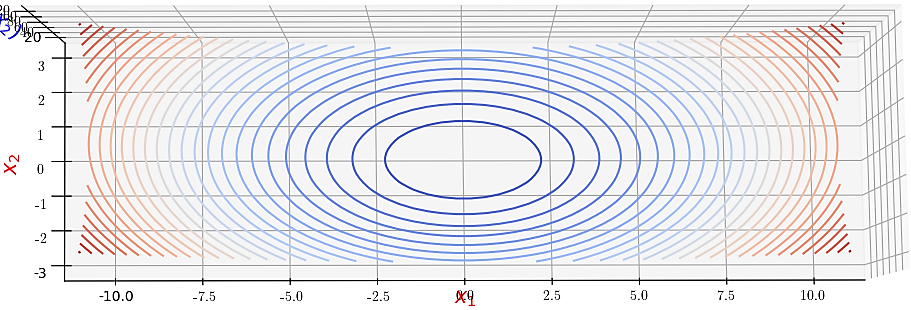
Tốc độ thực thi của thuật toán nhanh hơn The Normal Equation

**Problems of Gradient descent:**

* Local minima
  + Thuật toán có thể dừng lại Local minimun
  + Thuật toán có thể dừng lại ở Plateau
* Feature **s**cales
  + Nếu các feature có các khoảng giá trị giống nhau thuật toán sẽ chạy nhanh hơn





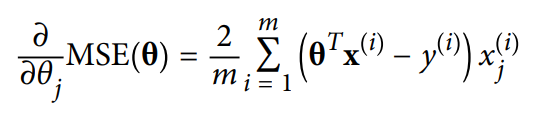
* + Nếu các feature có các khoảng giá trị lệch nhau nhiều thuật toán sẽ chạy lâu hơn

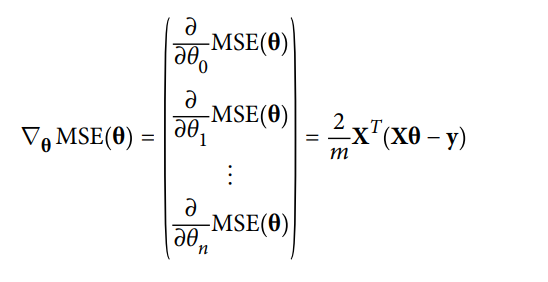
Các biến thể của Gradient Descent:

* **Batch** Gradient Descent
* **Stochastic** gradient descent
* **Mini-batch** gradient descent

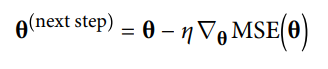
#### Batch Gradient Descent

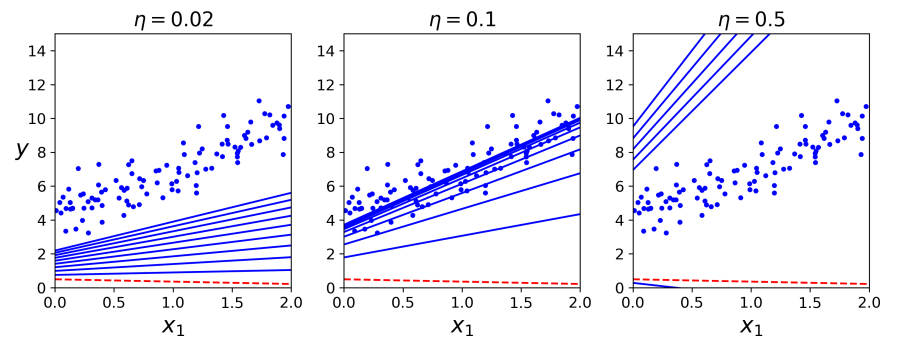
Lặp cho đến hội tụ.

**Partial derivatives of the Cost function**

**Gradient vector of the cost function**

Tại mỗi bước Batch gradient descent sử dụng và tính toán trên tập trainingset.

****Khi bạn đã có vector gradient thì bạn có thể trừ ∇θMSE(θ). Đây là lúc tỷ lệ học tập của η được phát huy tác dụng.

**Code**

#### Stochastic Gradient Descent

Trong thuật toán này, tại 1 thời điểm, ta chỉ tính đạo hàm của cost function dựa trên chỉ một điểm dữ liệu xi rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả.

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.

Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD

Trong đó:

J(θ;xi;yi) là cost function với chỉ 1 cặp điểm dữ liệu (input, label) là (xi,yi)

#### Mini – batch Gradient Descent

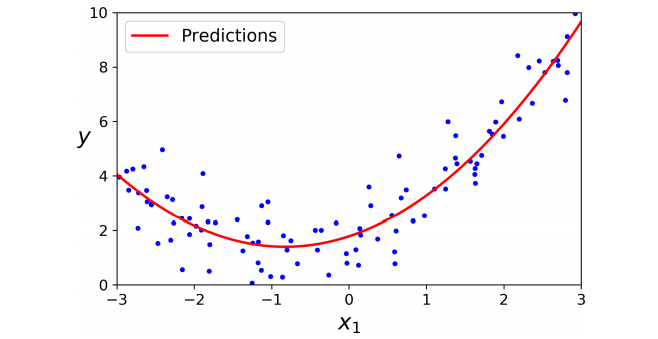
Mini-batch sử dụng một số lượng n lớn hơn 1 (nhưng vẫn nhỏ hơn tổng số dữ liệu Nrất nhiều). Giống với SGD, Mini-batch Gradient Descent bắt đầu mỗi epoch bằng việc xáo trộn ngẫu nhiên dữ liệu rồi chia toàn bộ dữ liệu thành các mini-batch, mỗi mini-batch có n điểm dữ liệu (trừ mini-batch cuối có thể có ít hơn nếu N không chia hết cho n). Mỗi lần cập nhật, thuật toán này lấy ra một mini-batch để tính toán đạo hàm rồi cập nhật.

Công thức:

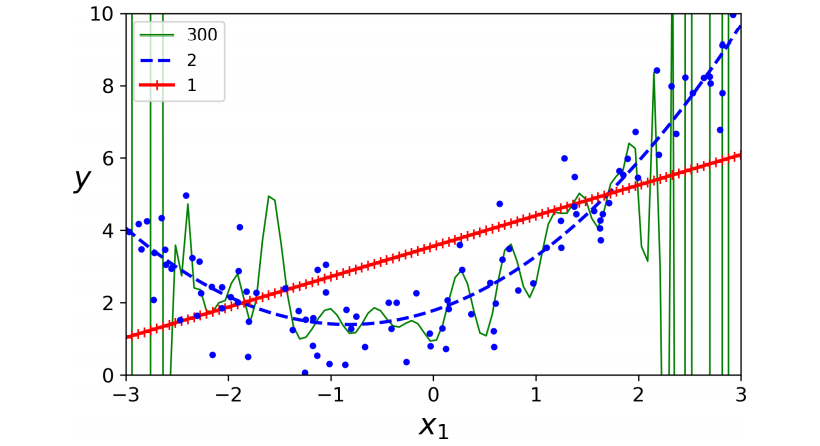
Trong đó:  
xi:i+n được hiểu là dữ liệu từ thứ i tới thứ i+n−1

**Code:**

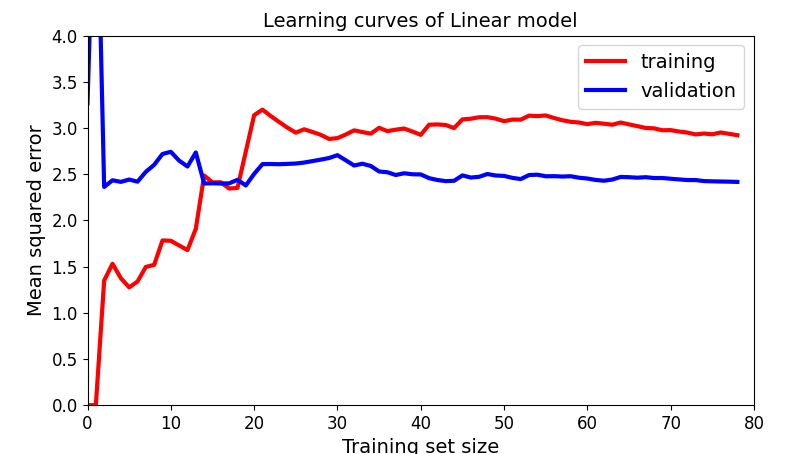
### Polynomial Regression

Nhìn vào hình ta thấy không thể dùng một đường thẳng để mô tả dữ liệu này. Mà nhìn hình ta lại tưởng tưởng ra một đường parabol để mô tả dữ liệu. Khi đó chúng ta cần **thêm bậc** rồi ấp dung linear để mô tả đúng cho dữ liệu và sklearn có hỗ trợ thêm bậc và kết quả như hình dưới.

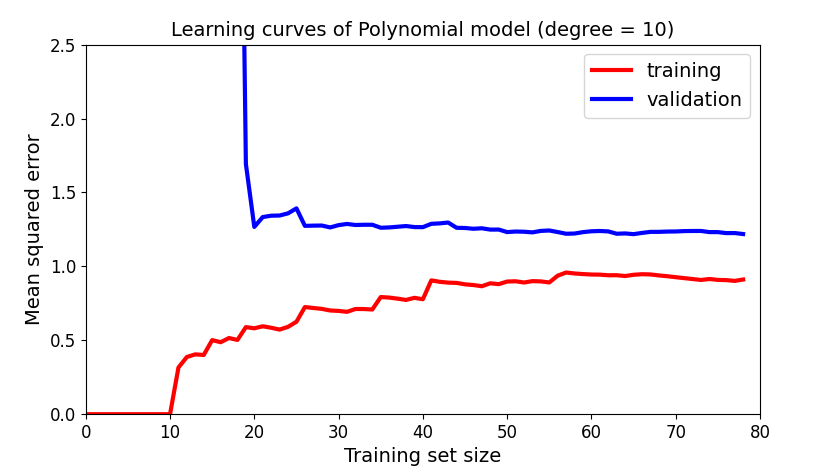
### Learning Curves



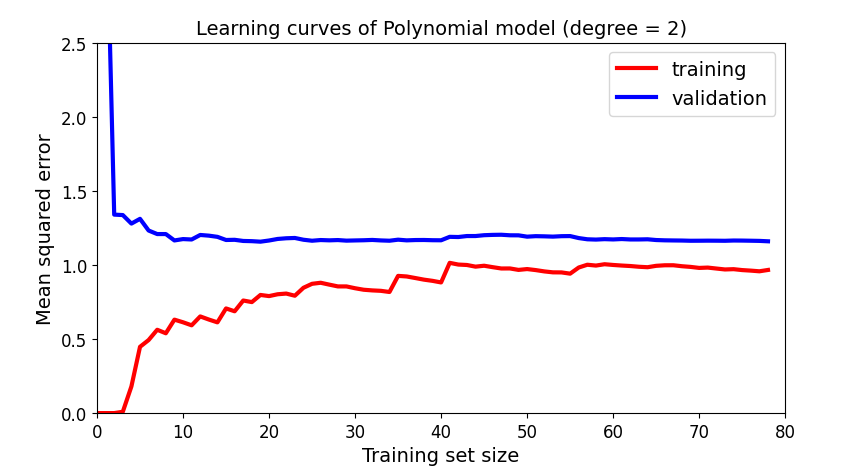
Khi ta sử dụng Polynomial Regression bậc 1 ta thấy được là một đường thẳng, bậc 2 ta thấy nó là một được parabol còn bậc 300 ta thấy một đường nó bám xác theo dữ liệu tranning. Từ đó cho ta thấy có hiện tượng là **overfitting hoặc underfitting** từng trường hợp. Đặc biệt bậc 1 đang bị underfitting còn bậc 300 sẽ bị overfitting.Ở chương 2 bạn đã sử dụng **cross-validation** để biết overfitting hoặc underfitting. Hôm nay mình sẽ giới thiệu bạn các phát hiện overfitting hoặc underfitting bằng **Learning Curves.**

Learning Curves là biểu đồ hiệu quả của model trên tập tranning set và validation set.

Ta thấy error của trainning vs validation > 2.5 cao nên ta suy ra hiện tượng này là underfitting



Ta thấy error trainning thấp mà error validation cao hoặc khoảng cách giữa training và validation cao thì chúng ta kết luận là hiện tượng này là overfitting



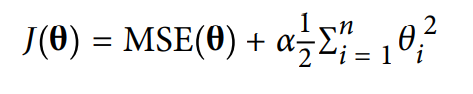
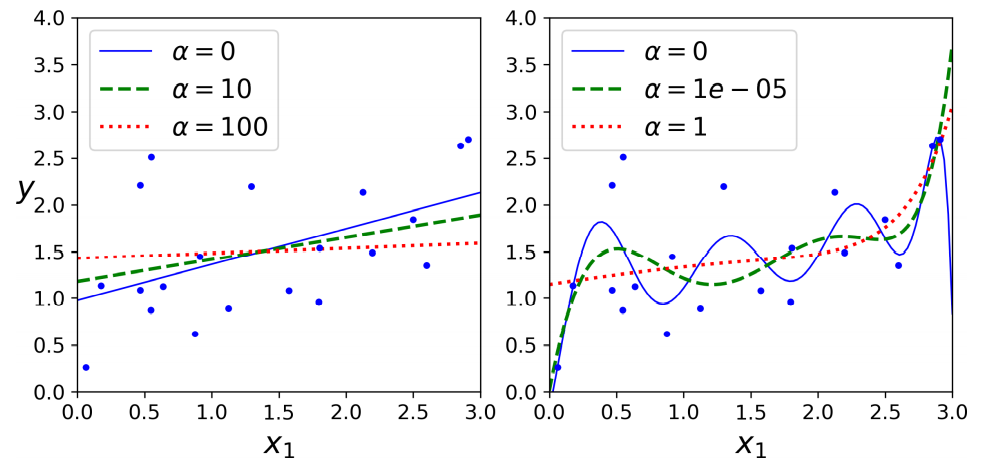
**Các lỗi thường gặp trong machine learning:**

* Bias
  + Phần lỗi tổng quát hóa này là do các giả định sai. Chẳng hạn như giả định rằng dữ liệu là tuyến tính nhưng thực sự là bậc 2.
  + High bias dẫn đến hiện tượng underfitting
  + Cách khác phục : Chọn model khác đặc biệt là model phức tạp hơn.
* Variance
  + Model này cố bán sát theo dữ liệu training do mô hình đa thức.
  + High Variance dẫn đến hiện tượng overfitting.
  + Model quá mức tạp so với dữ liệu.
  + Cách khác phục:
    - Thêm dữ liệu
    - Sử dụng những model đơn giản hơn
    - Regularization
    - Early stop
* Irreducible error
  + Do lỗi nhiễu của dữ liệu.
  + Cách duy nhất để giảm nhiễu sữa các nguồn dữ liệu chẳng hạn như cảm biến, ảnh kém chất lượng.

### Regularized Linear Models

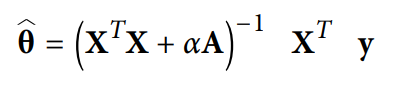
#### Ridge Regression

**Ridge Regression cost function**

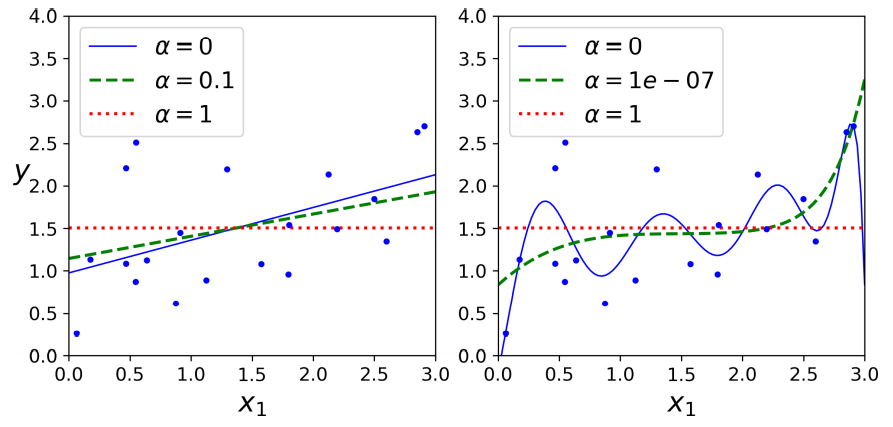
* Nếu α = 0 thì Ridge Regression chính là Linear Regression.
* Nếu α rất lớn thì tất cả các trọng số sẽ sấp xỉ bằng 0 là một đường thẳng đi quá MSE data.

Hình bên trái sử dụng Linear Regression có α.

Hình bên phải sủ dụng Polynomial Regression có α khi α càng tăng thì độ gợn sóng của **Hypothesis function** sẽ được giảm đi. Từ đó cho ta thấy Ridge Regression là một cách giúp cho ta khắc phục được hiện tượng overfitting.

**Ridge Regression closed-form solution**

#### Lasso Regression

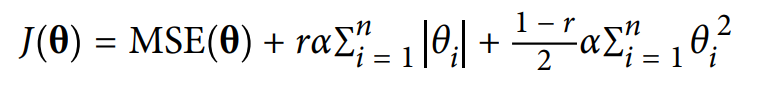
Thay vì Ridge Regression sủ dụng công thức theta bình phương thì Lasso Regression lại sử dụng công thức theta trị tuyệt đối phương pháp này giúp cho chúng ta ngăn chặn việc overfitting mạnh hơn Ridge Regression.

Khi α = 1 ở Ridge Regression gần có hiện tượng goodfitting trong khi α = 1 Lasso Regression thì đã bị underfitting từ đó cho ta thấy Lasso Regression ngăn chặn overfitting mạnh hơn Ridge Regression.

Do Lasso Regression ngăn overfitting quá mạnh nên có thể dẫn đến hiện tượng underfitting.

#### Elastic Net

Do những model đơn giản mà sử dụng Lasso Regression thì việc ngăn chặn overfitting quá mạnh dẫn đến underfitting hoặc những model phức tạp quá mà sử dụng Ridge Regression thì không đủ ngăn chặn overfitting từ đó Elastic Net ra đời từ sự kết hợp của Lasso Regression và Lasso Regression.



Trong đó:

* Lasso Regression

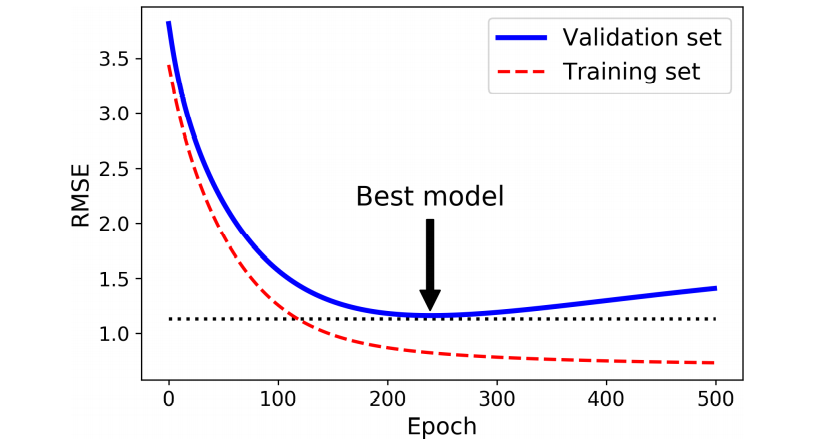


* Ridge Regression
* r là tham số để ta có thể ưu tiên chọn Lasso Regression hay Ridge Regression để ngăn chặn overfitting.

Khi đó ta có thể điều chỉnh r cho phù hợp để đưa ra model tốt nhất có thể.

**Code**

#### Early Stopping



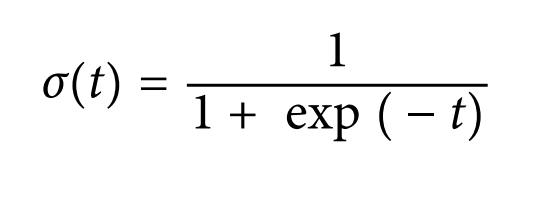
Ý tưởng là chúng ta dừng sau một khoảng thời gian RMSE không còn giảm được nữa thì ta dừng thuật toán.

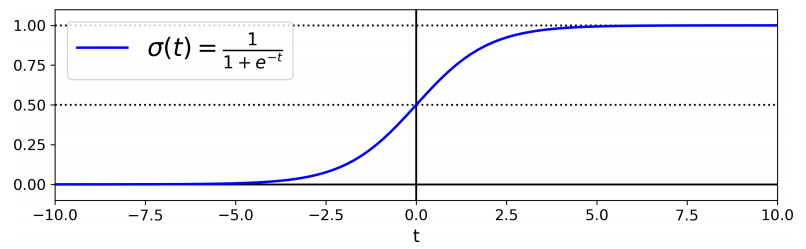
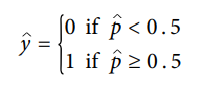
**Code**

### Logistic Regression

#### Estimating Probabilities

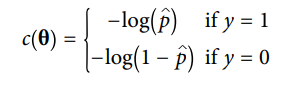
**Hypothesis function**

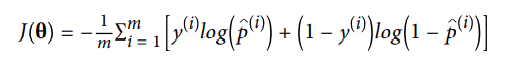
**Logistic function**

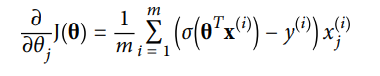
 **Logistic Regression model prediction**

Lưu ý: σ (t) < 0,5 khi t < 0 và σ (t) ≥ 0,5 khi t ≥ 0, do đó, **Logistic Regression model** dự đoán 1 nếu xTθ là dương và 0 nếu là âm.

#### Training and Cost Function

**Cost function**

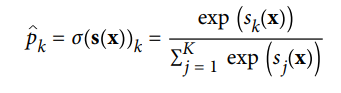
**Logistic Regression cost function**

**Logistic cost function partial derivatives**

**Code:**

#### Decision Boundaries

#### Sorfmax Regression

**Sorfmax function**

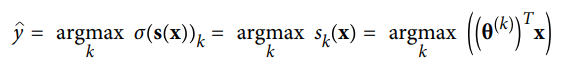


Trong đó:

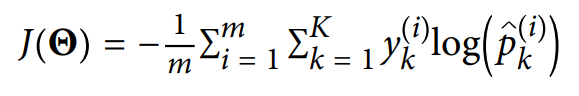
k là số lớp

s(x) là một vector chứa các scores của từng classs của một instance x

σ(s(x))klà xác suất của instance x của class k điểm của mỗi class của sample đó.

**Sorfmax Regression classifier prediction**

Toán tử argmax trả về giá trị maximizes a function

**Cross entropy cost function**

**Code:**

## Support Vector Machines

Support Vector Machine (SVM) model mạnh và linh hoạt trong học máy, có khả năng thực hiện phân loại linear, nonlinear, regression và even outlier detection.

SVM là mộ trong những model phổ biến nhất trong machine learning.

SVM là đặc biệt rất thích hợp cho việc phân loại các tập dữ liệu phức tạp nhưng có kích thước vừa và nhỏ.

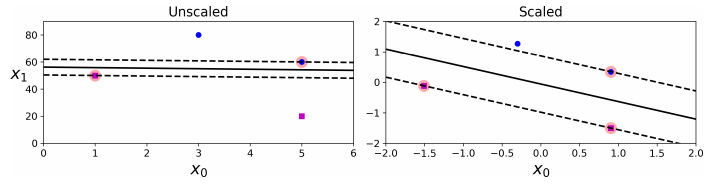
Chú ý: SVM là binary classifier

### Chart, scatter chart Description automatically generatedLinear SVM Classification

Model 1: Không tốt vì phân loại sai.

Model 2: Phân loại tốt với dữ liệu hiện tại những nếu ta thêm dữ liệu vào thì có thể không phân loại tốt nữa. Model này có **small margin.**

Model 3: Là model tốt nhất trong 3 model trên vì nó có **lagre marin**



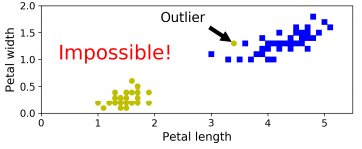
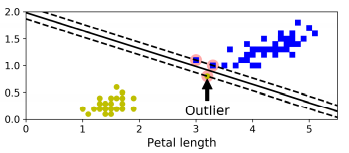
Những sample nằm gần vs Decision boundary giúp cho chúng ta định nghĩa ra suport vector để tạo ra margin.

#### Hard Margin Classification

Tất cả các sample của một class phải nằm về một phía của margin.

Hard margin là cố gắn tìm ra lagre margin lớn.

Problems:

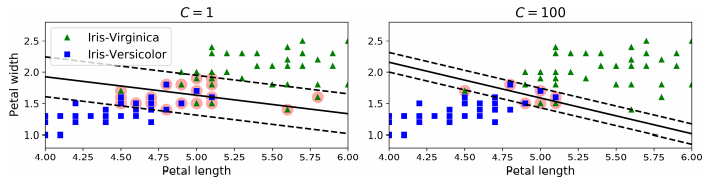
* Nếu dữ liệu là nonlinear.
* Outlier ( Nhạy cảm với dữ liệu ).

#### Sort Margin Classification

Cho phép một số sample của một class nằm về một phía còn lại của margin. Mục đích là để tìm lager margin.

Chú ý:

* Trong Scikit-Learn’s SVM bạn có thể điều chỉnh **C hyperparameter.** Nếu C small thì margin càng lớn và ngược lại nếu C large thì margin càng nhỏ.
* Larger margin thì sẽ cho ra better generalization vì nó mô tả đúng huynh hướng của dữ liệu.



C = 1 chấp nhận bị overfitting nhưng khi thêm dữ liệu vào nó vẫn đúng một số trường hợp cao hơn là C = 100.

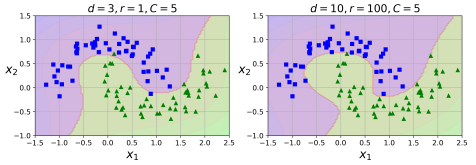
**Code:**

### Nonlinear SVM Classification

Như ta thấy trên hình không có một linear nào có thể mô tả được dữ liệu như hình trên.

Việc thêm các feature để thực hiện polinomial và có thể hoạt động tốt với tất cả các loại thuật toán Machine Learning (không chỉ SVM), nhưng ở mức độ degree thấp, nó không thể xử lý các bộ dữ liệu rất phức tạp và với mức độ degree cao, nó tạo ra một số lượng feature lớn, làm cho model quá chậm.

#### Polynomial Kernel

Thật may mắn khi SVM có hỗ trợ kernel trick nó có thể làm cho model có kết quả tương tự như thêm các feature và sử dụng polinomial, ngay cả degree cao mà không phải thêm feature. Vì vậy không có hiện tượng bùng nổ số lượng feature.

SVM classifiers with a polynomial kernel

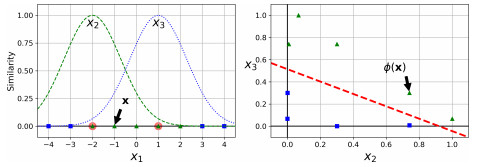
#### Adding Similarity Features

**Euclidean distance**

Như hình trên bạn thấy không có một đường thẳng nào có thể phân chia 2 class này được. Trong không gian ít chiều không nhận ra được thì ta thêm 1 feature gọi là landmarks độ giống nhau của dữ liệu ta được như hình bên dưới.

Hình này cho ta thấy có một đường thẳng cắt ngang và phân tách được dữ liệu.

#### Gaussian RBF Kernel

**Gaussian RBF:**





Trong đó:

* Abc
* (x,l) là khoảng cách hoặc độ giống nhau giữa sample x và landmark
* Y > 0 nếu y càng lớn thì khu vựcGaussian càng bị gom hay còn gọi là tập trung landmark lại và ngược lại.

##### Data transformation

Given a 1D data (data with 1 feature):

* 1 landmark 🡺 add 1 new feature: 1D 🡺 2D
* 2 landmarks 🡺 add 2 new features: 1D 🡺 3D
* K landmarks 🡺 add K new features: 1D 🡺 (K+1) D

Chuyển data sang một chiều mới lớn hơn số chiều trước khi chuyển.

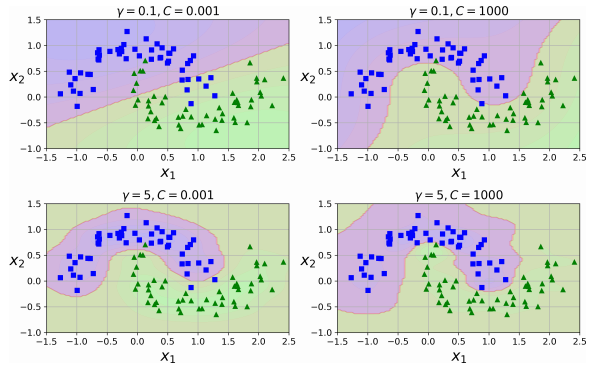
##### Select the landmarks

**Chọn mỗi sample làm landmarks .**

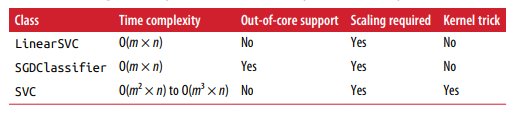
* Higher dimension: Có nhiều cơ hội để giải quyết vấn đề hơn:
* Nếu có m sample thì ta sẽ từ 1D 🡺 (m +1)D sẽ dẫn đến chậm trong việc tính toán.

**Sử dụng kernel trick**

* Kết quả sẽ giống tương tự như việc thêm lanmark
* 1D 🡺 1D giúp cho Chương trình chạy nhanh hơn.



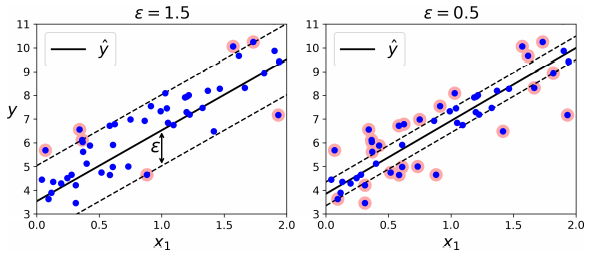
#### Computational Complexity



### SVM Regression

Thuật toán SVM khá linh hoạt: nó không chỉ hỗ trợ phân loại linear, nonlinear mà còn hỗ linear Regression và nonlinear Regression.

Thay vì cố gắng tìm cái large margin, SVM Regression cố gắng điều chỉnh phù hợp margin. Margin được kiểm soát bởi siêu thông số ϵ.



**Code:**

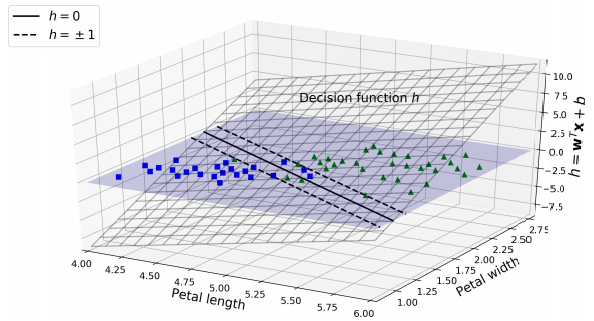
### Under the Hood

Các phần trước chúng ta đã tìm hiểu cách cài đặc thuật toán SVM bằng skitlearn. Phần này chúng ta sẽ tìm hiểu sâu về thuật toán SVM.

#### Decision Function and Predictions

The linear SVM classifier model predicts sample x theo công thức wTx + b = w1x1+ ⋯ + wnxn+ b nếu kết quả là dương thì dự đoán là class 1 ngược lại dự đoán là 0.

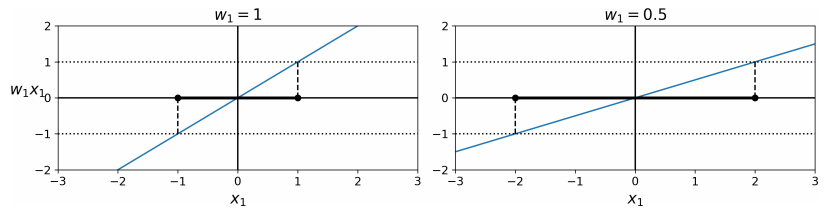




Hình ảnh minh họa

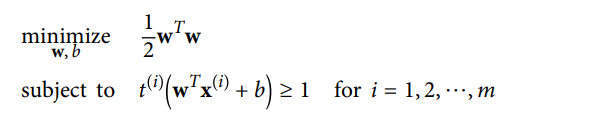
Hai đường margin trong đó decision function bằng 1 hoặc -1. Sẽ đưa ra kết quả dự đoán. Training a linear SVM classifier tìm ra w và b làm sao cho margin đạt được khoảng cách lơn nhất (hard margin) hoặc giới hạn nhỏ nhất (soft margin).

#### Training Objective



Với h = 1 và h = -1 khi w càng nhỏ thì margin càng lớn và ngược lại w càng lớn thì margin càng ngắn lại như hình trên được vẽ trên 2D

Trong Classification ta luôn muốn tìm cái large margin thì bắt buộc chúng ta phải minimize w

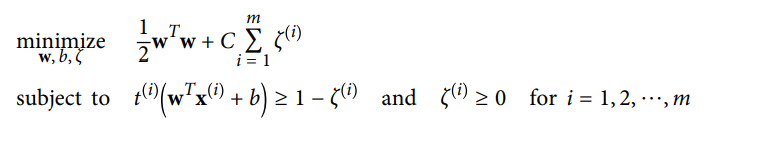
**Hard margin linear SVM classifier objective**

Để minimize w thì biểu giá trị của ½(wTw) phải đạt giá trị nhỏ thì bắt buộc (w12+ w22+…..+ wn-12+ wn2) thì các w1,w2,….. Phải đạt giá trị nhỏ.

Do nó là Hard margin nó không chấp nhận có những sample nằm trong margin nên nó kèm theo điều kiện subject to.

Do Hard margin nhạy cảm với outlier nên ta sẽ đi tìm hiểu sorf margin.

**Soft margin linear SVM classifier objective**



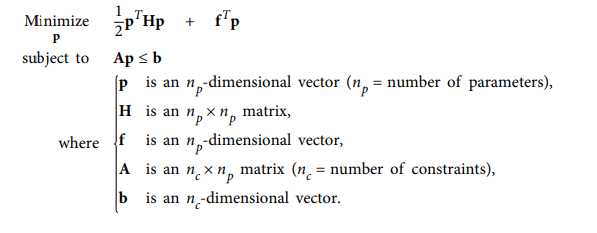
Ta cộng thêm tổng các slack valiable \* C cho phép hàm h(x) được tính sai. Giá trị slack càng lớn thì độ cho phép sai nhiều và ngược lại.

C large => slack small => giảm violation

#### Quadratic Programming

The hard margin and soft margin có các vấn đề tối ưu quadratic optimization về rằng buộc linear.

**Quadratic Programming problem**



Lưu ý biểu thức Ap <= b thực sự xác định nc : PTa(i) < b(i) với mọi i <= nc và thuộc N

A(i) Là một vector của ith tại một dòng của A và b(i) là một ith của b

Bạn có thể thật dễ dạng set QP parameters theo cách sau: Bạn cần get the hard margin linear SVM classfier objective

* np = n + 1, trong đó n là số lương feature (+1 là thêm bias)
* nc = m , trong đó m là số lượng sample trainning
* H là ma trận np X np
* f = 0 np – vector không gian chiều full 0
* b = -1 nc – vector không gian chiều full -1
* a(i) = -t(i)x(i) , trong đó x(i) bằng với x(i) extra bias feature x0 =1

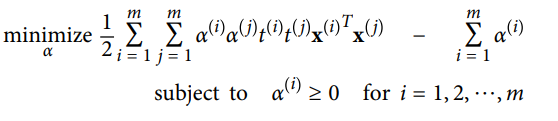
Vì vậy một cách train a hard margin linear SVM classifier chỉ sử dụng bộ giải pháp QP bằng cách passing cho nó các parameters trước đó.

Tuy nhiên bạn có thể sử dụng kernel trick để tối ưu hơn.

#### The Dual Problem

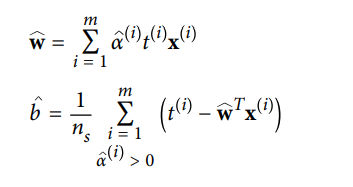
Với một bài toán tối ưu hóa có giới hạn được gọi là primal problem, có thể biểu diễn bài toán đó trên một bài toán khác nhưng có liên quan chặt chẽ với nhau gọi là dual problem. Lời giải của dual problem thường cung cấp giới hạn thấp hơn của bài toán primal problem nhưng trong số điều kiện nó thậm chí có thể có các lời giải tương tự bài toán primal problem. May mắn thay SVM đáp ứng các điều kiện trên. Vì vậy để chọn giải pháp primal problem hoặc dual problem sẽ có cùng một giải pháp.

**Dual form of the linear SVM objective**



Khi đó bạn tìm một vector a mũ minimized của vector. Bạn có thể tính w và b tối thiểu của bài toán primal problem

**From the dual solution to the primal solution**



The dual problem nhanh hơn the primal khi số lượng sample tranning nhỏ hơn số lượng feature. Quan trọng hơn nó cho thủ thuật kernel trick. Trong khi primal thì không được sử dụng thủ thuật.

#### Kernelized SVM

#### Online SVMs

## Decision Trees

Decision Trees là một thuật toán linh hoạt trong machine learning có thể kết hợp classification và regression.

Chúng là những thuật toán tạo ra model rất mạnh có khả năng phù hợp với các bộ dữ liệu phức tạp có thể dẫn đến overfitting.

Decision Trees là thành phần cơ bản của random forests.

### Trainning and Visualizing a Descision Tree

### Making Predictions

#### Estimating Class Probabilities

A Decision Tree cũng có thể ước tính xác suất một sample thuộc cụ thể một class nào:

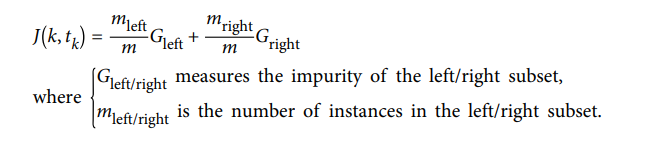
* B1: Nó đi ngang qua cây để tìm nút lá cho sample
* B2: Trả về tỷ lệ các trường hợp huấn luyện của từng class trong node.
* VD: bạn tìm thấy một bông hoa có độ dài 5cm và rộng 1.5 cm nhìn vào hình trên node tìm thấy là Node màu xanh lá. Lần lượt các xác xuất là [0/54,49/54,5/54] trong xác xuất ta thấy class 1 chiếm tỷ lệ cao nên được dự đoán bông mình tìm thấy là thuộc class 1.

### The Cart trainning Algorithm

IDE: Chọn feature k và threshold t\_k cắt train set thành 2 subsets làm cho impurity giảm xuống và lặp lại tiếp tục chọn feature k và threshold t\_k

Thuật toán dừng khi:

* Đạt max depth
* Không thể giảm impurity được nữa.

m = mleft + mright

The Cart algorithm là một thuật toán tham lam: nó tham lam trong việc đi tìm optimum split. Sau đó lặp quy trình ở mỗi cấp. Một thuật toán tham lam thường tạo ra một giải pháp hợp lý những nó không được đảm bảo là giải pháp tối ưu.

### Computational Complexity

Về việc yêu cầu đưa ra dự đoán thì Decision tree đi từ node root đến node leaf. Decision tree thường gần như cân bằng. Vì vậy việc tìm kết quả trong Decision tree là O(log2(m)) nodes. Vì mỗi node chỉ yêu cầu kiểm tra giá trị của một feature nên độ phức tạp của dự đoán là O(log2(m)) không phụ thuộc vào số lượng feature. Vì vậy tốc độ xử lý nhanh với các large tranning sets.

Tuy nhiên Decision tree tranning toàn bộ features (hoặc ít hơn max features). Điều này dẫn đến độ phức tạp huấn luyện là O(n x mlog(m)).

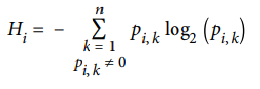
### Gini Impurity or Entropy?

Mặc định, dùng để đo độ tạp chất Gini, những bạn có thể sử dụng Entropy thay thế bằng cách đặt siêu thông số.

Khái niệm entropy được bắt nguồn từ nhiệt động học như là một thước đo của sự rối loạn phân tử: entropy = 0 khi các phân tử không chuyển động.

Trong machine learning nó là một thước đó tạp chất: set entropy = 0 khi nó chỉ chứa các sample chỉ thuộc một class.

**CT**



**EX:** − 49/54 log2(49/54) – 5/54 log2(5/54) ≈ 0.445

Vậy khi nào chúng ta sử dụng Gini hoặc Entropy?

* Most of the time: Dẫn đến Decision tree gần giống nhau.
* Gini impurity: Gini nhanh hơn một chút so với Entopy.
* Entropy: Tạo ra một cây cân đối hơn Gini.

### Regularization Hyperparameters

Parametric models: Số lượng tham số đã được định nghĩa trước khi được train

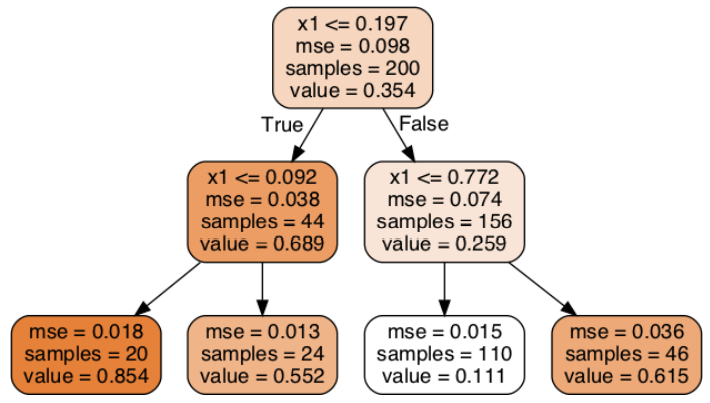
Non-parametric models: Không xác định được tham số khi được train có thể dẫn đến overfitting

Regularization for decision trees:

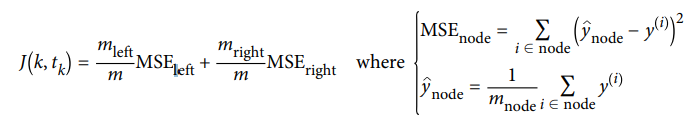
* max\_depth:
* min\_samples\_split: Số lượng sample của một node
* min\_samples\_leaf: Số lượng sample của node lá
* max\_leaf\_nodes: Số lượng node lá
* max\_features: Số lượng feature

Nếu tiền tố là max thì ta nên giảm lại và tiền tố là min thì nên tăng thì sẽ giảm overfitting

### Regresssion

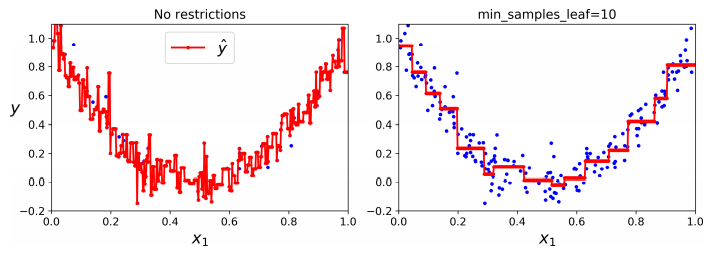


**CART cost function for regression**



Thuật toán Cart hầu như hoạt động như thuật toán Cart trên ngoại trừ chia tập huấn luyên theo MSE

**Regularizing a Decision Tree regressor**



Instability

.m

..

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.

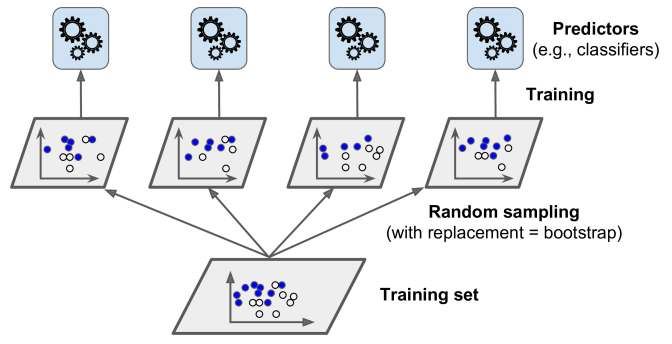
**Bootstrapping**

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).

**NonBootstrapping**

Đây là một kỹ thuật thống kê từ 1 dữ liệu mới và không để lại bộ dữ liệu cũ.

**Mô hình**



Do khi ta chia ra data nên ta có thể chạy song song tối ưu được thời gian trainning cho thời gian cho dữ liệu khổng lớn.

Do tạo ra nhiều tập con nên ta có thể sử dụng tập còn lại làm validation để test model và đánh giá.

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.

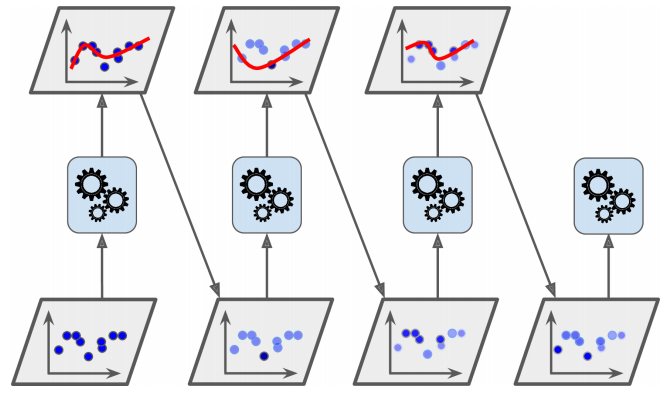
Random splip point của feature rất nhanh chống không cần thời gian tìm thresholds. Phù hợp cho dữ liệu lớn hoặc số lượng cây nhiều.

Khi chạy random forests nó sẽ cho mình biết cách feautre quan trọng. random forests nó sẽ trả số lượng các node mà nó học ra.

Boosting là một kỹ thuật thực hiện thông qua việc xây dựng một mô hình từ dữ liệu huần luyện, sau đó các mô hình tiếp theo sẽ được tạo ra và cố gắng khắc phục cố gắn sửa lỗi các model đầu tiên. Các mô hình được thêm vào cho đến khi huấn luyện được dự đoán hoàn hảo hoặc số lượng mô hình đạt cực đại.

AdaBoost được sử dụng với Decision tree có depth nhỏ. Sau khi Decision tree đầu tiên tạo ra hiệu xuất của cây trên mỗi mẫu training được sử dụng thông tin để quyết định cây tiếp theo sẽ tập trung vào mẫu dữ liệu nào. Dữ liệu huấn luyện khó dự đoán sẽ được đánh trọng số lớn hơn so với các trường hợp khác.

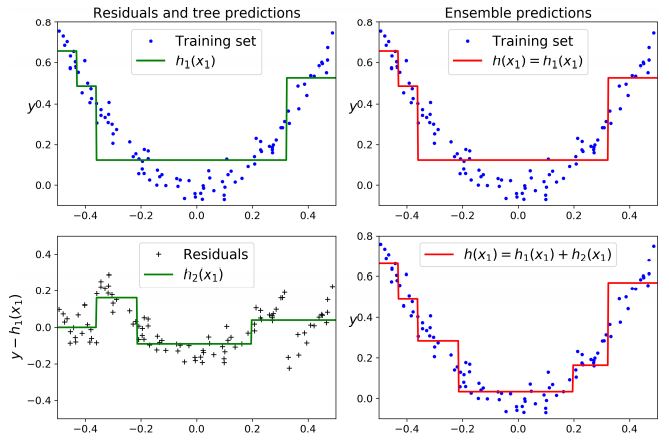
Các mô hình được tạo ra lần lược. Hiệu suất của mô hình trước ảnh hưởng đến hiệu suất của mô hình sau được tạo ra. Sau khi tất cả các mô hình được xây dựng dự đoán được thực hiện trên dữ liệu mới. Lần này mỗi cây được đánh trọng số tùy vào độ chính xác của nó trên tập dữ liệu training.

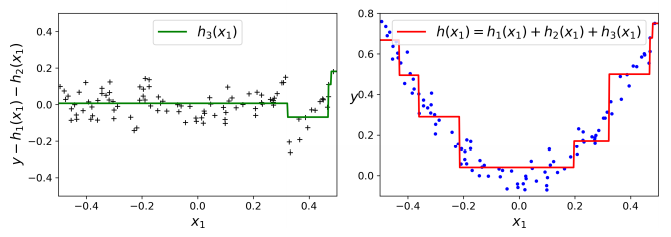


Màu xanh đậm là đánh weight các model sau khác phục lỗi model trước

#### Do mô hình nó train nó là nối tiếp nên sẽ rất mất thời gian để tìm ra model tốt nhất

Thay vì tăng dần trọng số của các ví dụ bị học sai, trong Gradient Boosting các learner sau sẽ học sai số của learner trước nó.





Prediction của ensemble model sẽ là tổng của các prediction của các learner thành phần (regression).

Gradient Boosting hoạt động được cả cho Classification và Regression

Ta thấy rằng, các leaner sau học sai số của learner trước do đó, ensemble model sẽ dần dần khiến cho loss=0 –> Boosting **giúp giảm bias cho mô hình.**