TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

Báo cáo đồ án môn học

Project II

Tìm hiểu về Machine learning và Deep learning

Sinh viên thực hiện : Lê Thị Trung Thu

MSSV : 20183637

Giảng viên hướng dẫn : Thầy Đỗ Phan Thuận

Hà Nội

7/2021

Mục lục

[Lời mở đầu 2](#_Toc77986517)

[Chương 1. Học máy (machine learning) và áp dụng 4](#_Toc77986518)

[1.1. Hồi quy tuyến tính (Linear Regression) 4](#_Toc77986519)

[1.2. Hồi quy Logistic (Logistic Regression) 4](#_Toc77986520)

[1.3. Mạng neural nhân tạo 7](#_Toc77986521)

[1.4. Máy vector hỗ trợ (Support Vector Machine) 8](#_Toc77986522)

[Chương 2. Thị giác máy tính (computer vision) và áp dụng 14](#_Toc77986523)

[2.1. Convolutional Neural network 14](#_Toc77986524)

[2.1.1. Cơ sở lý thuyết 14](#_Toc77986525)

[15](#_Toc77986526)

[2.1.2. Áp dụng vào bài toán nhận diện chứ số MNIST 16](#_Toc77986527)

[2.2. Phát hiện vật thể (Object Detection) 17](#_Toc77986528)

[2.2.1. Cơ sở lý thuyết 17](#_Toc77986529)

[2.2.2. Áp dụng vào bài toán phát hiện xe trên cao tốc 23](#_Toc77986530)

[Chương 3. Khó khăn và hướng phát triển 26](#_Toc77986531)

[Tài liệu tham khảo 26](#_Toc77986532)

# Lời mở đầu

Trong những năm trở lại đây, AI – Artificial Intelligent (Trí tuệ nhân tạo) , cụ thể hơn là ML – Machine learning (Học máy hoặc máy học) đang phát triển mạnh mẽ và được sự quan tâm đặc biệt từ cộng đồng học thuật cũng như từ xã hội. Là trái tim của cuộc Cách mạng công nghiệp lần thứ 4, trí tuệ nhân tạo đang len lõi vào mọi lĩnh vực trong cuộc sống hằng ngày của chúng ta mà có thể chúng ta không nhận ra. Ví dụ như: hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý quảng cáo từ Facebook, khả năng tự động tag khuôn mặt trong ảnh của Facebook, tính năng tự động phân loại ảnh theo Album của Google photo và nổi bật hơn cả không thể không kể đến xe tự lái của Google, Tesla; máy chơi cờ vây AlphaGo của Google DeepMind, . . .

Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning. Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc.

Trong môn học này, tôi được sự hướng dẫn của thầy Đỗ Phan Thuận, đã tìm hiểu được các kiến thức tổng quan về Machine learning và Deep learning thông qua khóa học Machine learning trên trang <https://www.coursera.org/> và khóa học Computer Vision trên <https://www.deeplearning.ai/> và đi sâu một số kiến trúc áp dụng trong các bài toán cụ thể.

Bài báo cáo của tôi bao gồm 2 chương :

Chương 1: Học máy (Machine learning) và áp dụng

Chương 2: Thị giác máy tính (Computer Vision) và áp dụng

Chương 3: Khó khăn và hướng phát triển

# Chương 1. Học máy (machine learning) và áp dụng

## Hồi quy tuyến tính (Linear Regression)

* Biểu diễn:

m : số lượng ví dụ học

n : số lượng đặc trưng của 1 ví dụ học

* Đầu ra dự đoán :
* Hàm mất mát (Loss function) :
* Gradient Descent:
* Phương trình chuẩn tắc:

## Hồi quy Logistic (Logistic Regression)

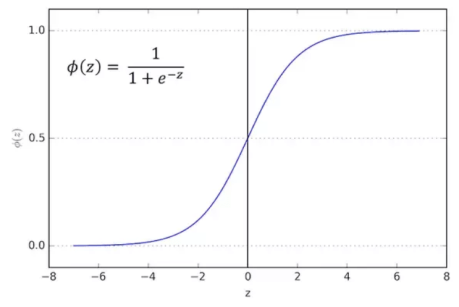
Đầu ra dự đoán của bài toán hồi quy logistic :

Trong đó : là hệ số cần học

x là giá trị đầu vào

g là hàm logistic, ở đây g là hàm sigmoid

a, Hàm Sigmoid



Nguồn : https://nttuan8.com/bai-2-logistic-regression/

* Hàm số liên tục, nhận giá trị thực trong khoảng (0,1).
* Hàm có đạo hàm tại mọi điểm (để áp dụng gradient descent)

g′(z) = = = g(z)(1 – g(z))

b, Mô hình

Xét bài toàn phân loại 2 nhãn lớp, giá trị dự đoán y nhận 2 giá trị 0 và 1

là xác suất của điểm dữ liệu x để output y = 1

c, Hàm mất mát

Hàm mất mát cho 1 điểm dữ liệu :

Ta có:

Do đó :

Biểu diễn dưới dạng vector:

Cập nhật hệ số theo Gradient Decent :

d, Ranh giới quyết định (Decision Boundary)

## Mạng neural nhân tạo

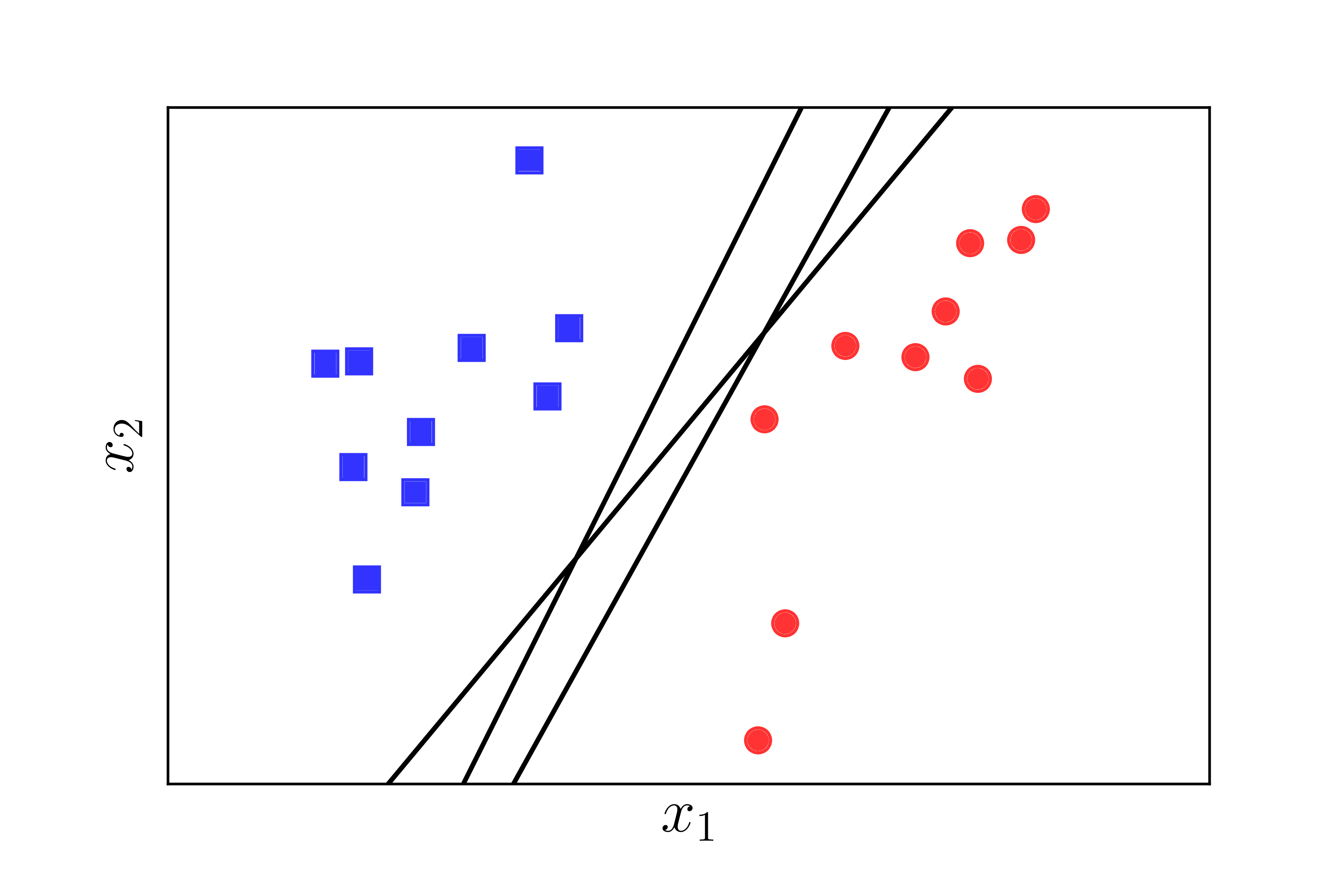
* Đầu vào : x1…xn, bias x0 = 0
* Đầu ra :
* : “activation” của nốt i trong layer j
* : ma trận trọng số từ lớp j đến lớp j+1, nếu lớp j có sj­ nốt và lớp j+1 có sj+1 nốt thì có chiều là sj+1 x (sj + 1)
* g là hàm kích hoạt (bẻ cong sự tuyến tính) : sigmoid, tanh, relu, leaky relu
* Biểu diễn vector:

Đặt

## Máy vector hỗ trợ (Support Vector Machine)

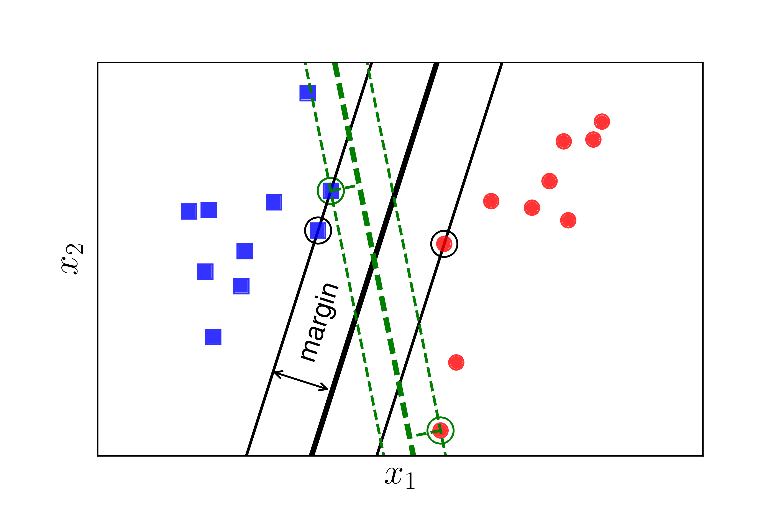
a, Linear SVM:

Trong bài toán phân loại 2 nhãn lớp, có thể tồn tại vô số siêu phẳng phân chia hai lớp, vậy câu hỏi đặt ra là siêu phẳng nào phân chia hai lớp tốt nhất theo một tiêu chuẩn nào đó



Nguồn : https://machinelearningcoban.com/2017/04/09/smv/

SVM được ra đời nhằm giải quyết câu hỏi trên, bằng cách xây dựng bài toán tối ưu sao cho tìm được đường phân chia có lề (margin) lớn nhất. Lề (margin) được định nghĩa là khoảng cách từ điểm gần nhất của mỗi lớp tới siêu phẳng phân chia và khoảng cách này là bằng nhau đối với 2 lớp



Nguồn : https://machinelearningcoban.com/2017/04/09/smv/

Bài toán tối ưu cho SVM được định nghĩa như sau:

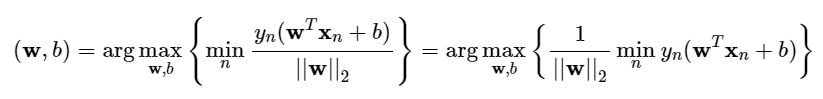
Giả sử rằng các cặp dữ liệu của tập huấn luyện là (x1, y1), (x2, y2), … , (xn, yn) với vector xi ∈ R­d là đầu vào của một điểm dữ liệu và yi là nhãn của điểm dữ liệu đó. d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi yi = 1 (class 1) hoặc yi = -1 (class 0)

Xét siêu phẳng phân chia 2 lớp có dạng: wTx + b = 0 trong đó w ∈ Rd và b ∈ R là các hệ số của siêu phẳng chúng ta cần tìm. Với cặp dữ liệu (xn, yn) bất kỳ, khoảng cách từ điểm đó đến siêu phẳng được xác định bởi công thức:

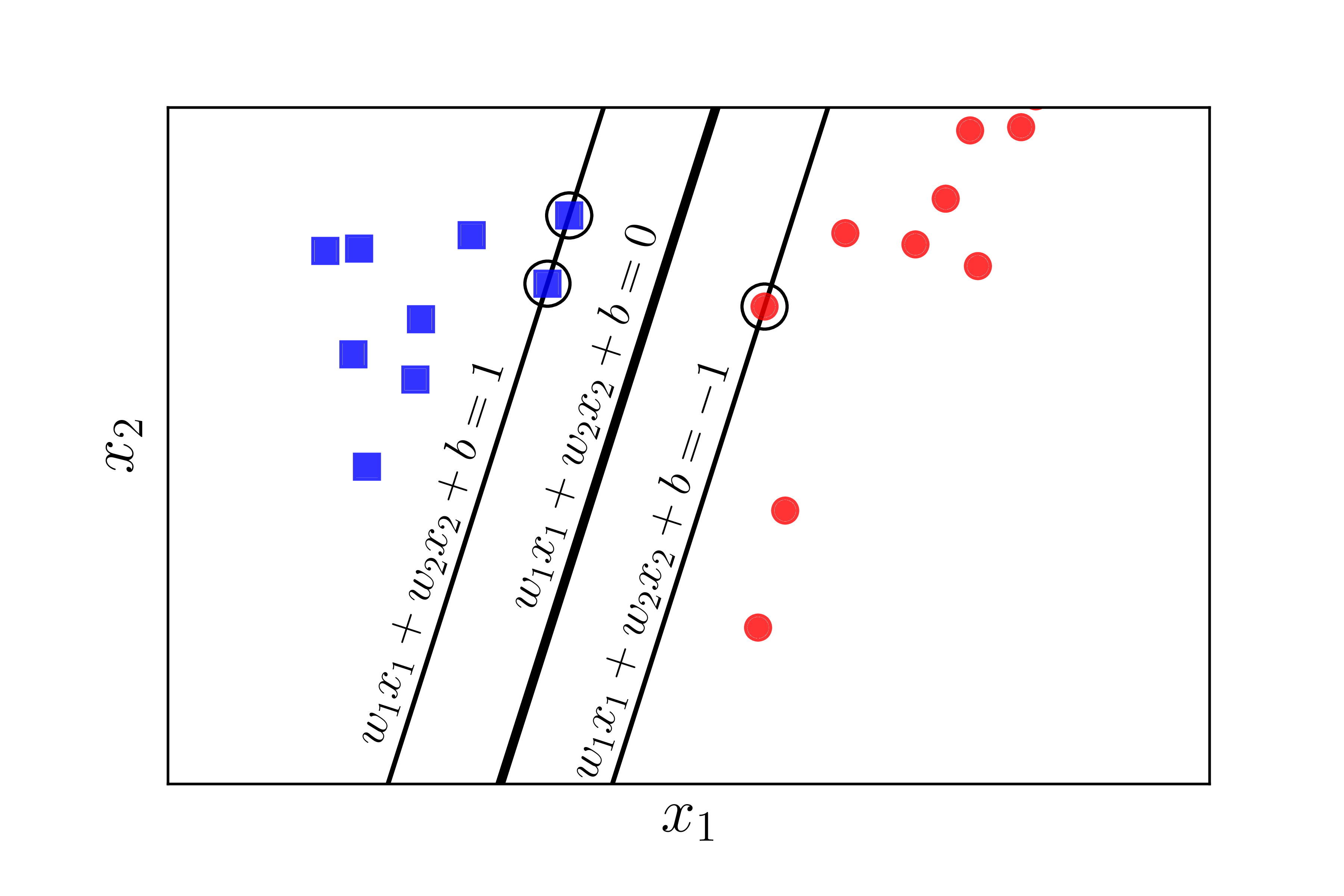
Với cách tính khoảng cách như trên, margin được tính là khoảng cách gần nhất từ 1 điểm tới siêu phẳng phân cách (bất kể điểm nào trong 2 lớp)

margin = min

Bài toán tối ưu trong SVM chính là bài toán tìm **w** và **b** sao cho *margin* này đạt giá trị lớn nhất



Giả sử yn(wTxn + b) = 1 với những điểm nằm gần siêu phẳng phân chia nhất như hình vẽ trong không gian 2 chiều dưới đây

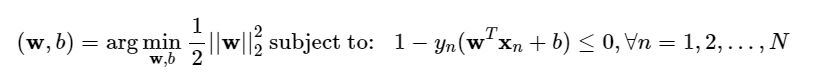


Nguồn : https://machinelearningcoban.com/2017/04/09/smv/

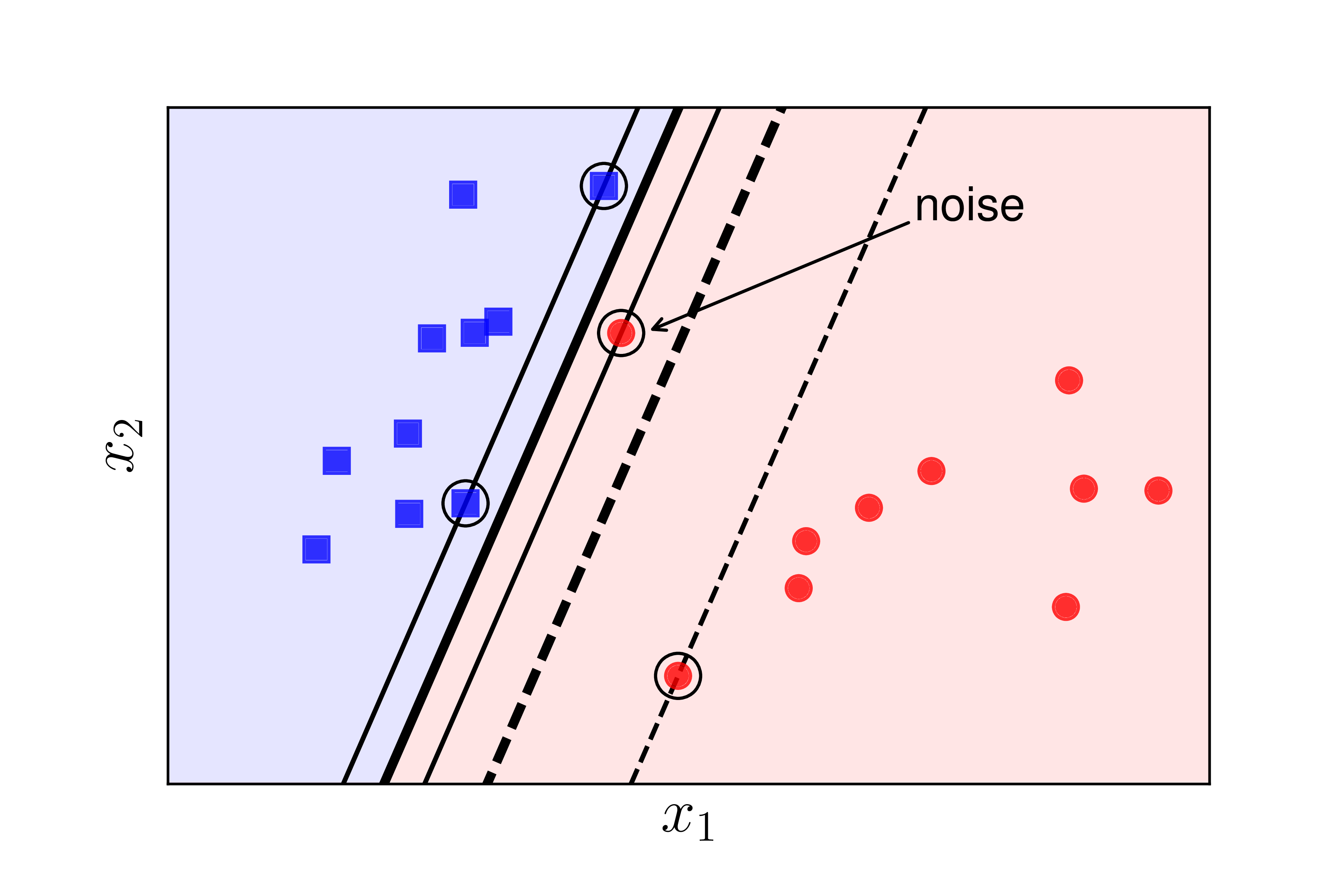
Như vậy với mọi n, ta có:

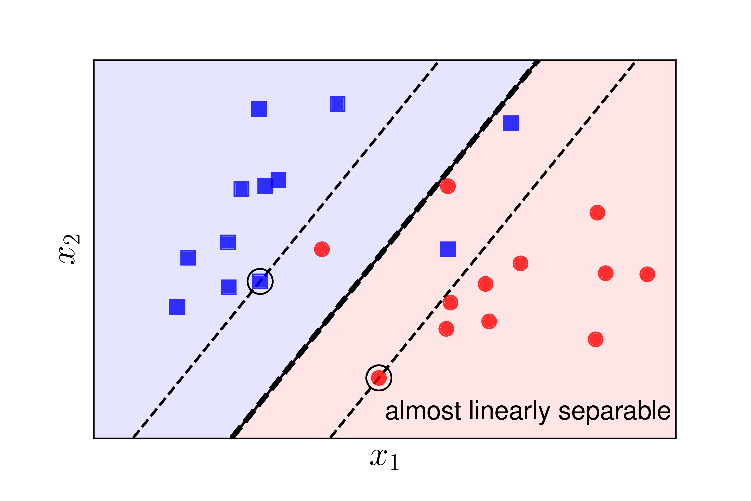
yn(wTxn + b) ≥ 1

Ta đưa bài toán tối ưu ban đầu về bài toán tối ưu có ràng buộc sau đây:



Đây là bài toán lề cứng (hard-margin) SVM. Tuy nhiên, trong thực tế, tập dữ liệu chúng ta sử dụng thường không phân biệt tuyến tính giữa 2 lớp hoặc có chứa các điểm dữ liệu nhiễu, khi đó bài toán tối ưu trên trở nên vô nghiệm. Nếu ta chịu hy sinh một số điểm dữ liệu nằm ở trong khoảng lề (margin), ta vẫn có thể tạo được một siêu phẳng phân chia khá tốt như ví dụ trong hình vẽ dưới đây





Nguồn : https://machinelearningcoban.com/2017/04/09/smv/

Dựa trên quan sát trên, ta thu được một biến thể của Hard Margin SVM có tên gọi là Soft Margin SVM.

Bài toán tối ưu cho Soft Margin SVM được xây dựng tương tự như bài toán tối ưu của Hard Margin SVM, với sự bổ sung của một đại lượng biểu diễn sự hy sinh ξn tương ứng với mỗi điểm dữ liệu xn trong toàn bộ dữ liệu huấn luyện. Với những điểm xn nằm trong vùng an toàn (nghĩa là nằm bên ngoài lề của hai lớp, ξn = 0. Ngược lại, với những điểm không nằm trong vùng an toàn , ta có ξn > 0.

Ta có bài toán tối ưu cho Soft Margin SVM như sau:

min

Với điều kiện:

Trong đó hằng số C được dùng để điều chỉnh tầm quan trọng giữa việc tối ưu margin và tối ưu sự hy sinh. Hằng số này được xác định từ trước hoặc có thể được xác định bằng việc sử dụng một tập tối ưu (validation set)

Việc giải bài toán tối ưu trên là không hề dễ dàng, do đó bài toán tối ưu trên thường được chuyển về bài toán đối ngẫu tương ứng để giải

Bài toán đối ngẫu (dual problem) có dạng:

max

Với điều kiện:

Bài toán đối ngẫu trên có thể được giải quyết bằng các phương pháp số học (để giải quyết bài toán tối ưu hàm lồi bậc hai có ràng buộc tuyến tính)

Sau khi giải được bài toán đối ngẫu tìm ra các , ta có thể tính ngược lại w và b của siêu phẳng cần tìm.

Quá trình phân loại một điểm dữ liệu mới được thực hiện bằng cách tính sign(wTx + b) và gán nhãn y= 1 nếu kết quả trả về lớn hớn 0 và y = -1 trong trường hợp ngược lại. Một đặc điểm quan trọng trong việc dự đoán nhãn cho một điểm dữ liệu mới là việc tính (wTx + b) chỉ phụ thuộc vào các điểm dữ liệu có hệ số tương ứng khác 0. Các ví dụ này được gọi là các **vector hỗ trợ (support vector)**. Do đó, ta có thể thấy rằng siêu phẳng phân tách được xác định dựa trên tập các **vector hỗ trợ**.

Ngoài góc nhìn bài toán tối ưu có ràng buộc, Soft Margin SVM có thể được đưa về bài toán tối ưu không ràng buộc dựa trên hinge loss:

min

b, Kernel SVM:

Đối với bài toán phân loại 2 lớp không thể phân tách bằng một siêu phẳng, Kernel SVM có thể được sử dụng để tìm một phép biến đổi sao cho dữ liệu ban đầu là không phân biệt tuyến tính được biến đổi sang không gian mới, với hy vọng dữ liệu sau khi được biến đổi sẽ trở nên phân biệt tuyến tính và từ đó có thể áp dụng các phương pháp Linear SVM như đã đề cập ở phần trước.

Như vậy, Kernel SVM tiến hành tìm một hàm biến đổi dữ liệu x từ không gian feature ban đầu thành dữ liệu trong một không gian mới bằng hàm số

Sau khi biến đổi x sang không gian mới, bài toán đối ngẫu trong Soft Margin SVM trở thành:

max

Với điều kiện:

Tuy nhiên, việc tính toán trực tiếp cho mỗi điểm dữ liệu có thể sẽ tốn rất nhiều bộ nhớ và thời gian vì số chiều của thường rất lớn, có thể là vô hạn. Thêm nữa, để tìm nhãn của một điểm dữ liệu mới x ta lại phải biến đổi x thông qua hàm rồi lấy tích vô hướng của nó với tất cả các với m trong tập hợp các support. Để tránh việc này, người ta áp dụng kĩ thuật **kernel trick** để tính dựa trên x và z mà không cần tính trực tiếp và

Trong bài toán này, chúng tôi thử nghiệm 3 hàm kernel để thử nghiệm cho bài toán Kernel SVM:

* Sigmoid kernel:

Trong đó và r là các hằng số

* Polynomial kernel:

Trong đó d là một hằng số dương để chỉ bậc của đa thức. d có thể không là số tự nhiên vì mục đích của chúng ta không phải bậc của đa thức mà là cách tính kernel. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên

* Radial Basis Function (RBF):

# Chương 2. Thị giác máy tính (computer vision) và áp dụng

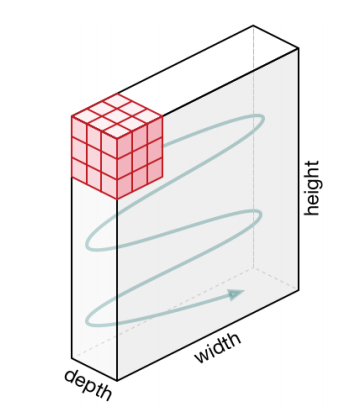
## 2.1. Convolutional Neural network

### 2.1.1. Cơ sở lý thuyết

a. Convolutional Layer

Ảnh được đưa vào các Concolution layers để trích xuất ra các đặc trưng của ảnh. Mỗi kernel khác nhau có ý nghĩa khác nhau (làm nét ảnh, làm mờ ảnh, xác định các đường, ...)

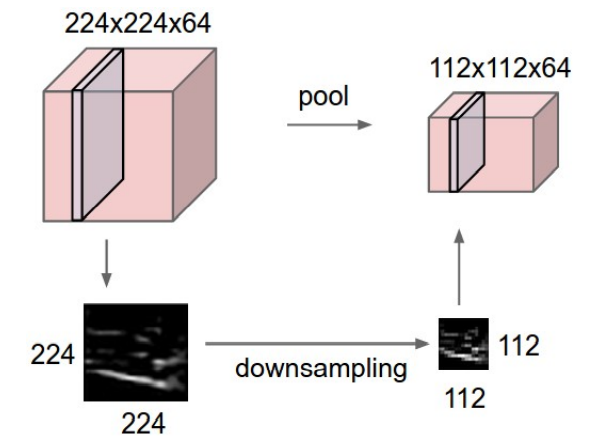
Output của convolutional layer này sẽ qua hàm non-linear activation function trước khi trở thành input của convolutional layer tiếp theo



b. Pooling Layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng.

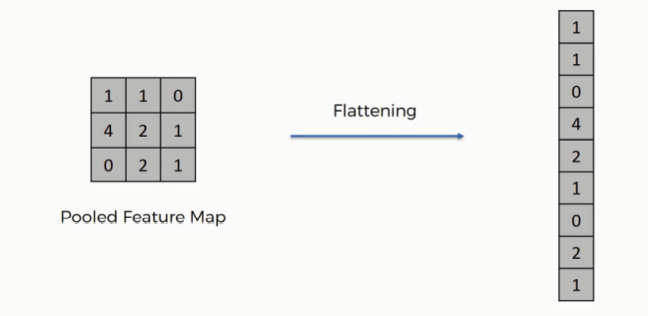
Có 2 loại pooling layer phổ biến là max pooling và average pooling.



### 

c. Fully Connected Layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối sẽ được Flattened và được đưa và fully connected layer. Tại đây, các đặc điểm của ảnh được kết hợp lại và đưa ra outout cho model



### 2.1.2. Áp dụng vào bài toán nhận diện chứ số MNIST

Model gồm:

* 2 convolution layers, mỗi layers đều sử dụng 32 kernel kích thước 3x3, activation fucnction là sigmoid
* 1 max pooling layer với pool-size = 2x2
* Fully connected layer với 128 nodes, activation fucnction là sigmoid
* Output layer với 10 node và dùng softmax function để chuyển sang xác suất
* Độ chính xác trên tập test set 98.86%

## 2.2. Phát hiện vật thể (Object Detection)

### 2.2.1. Cơ sở lý thuyết

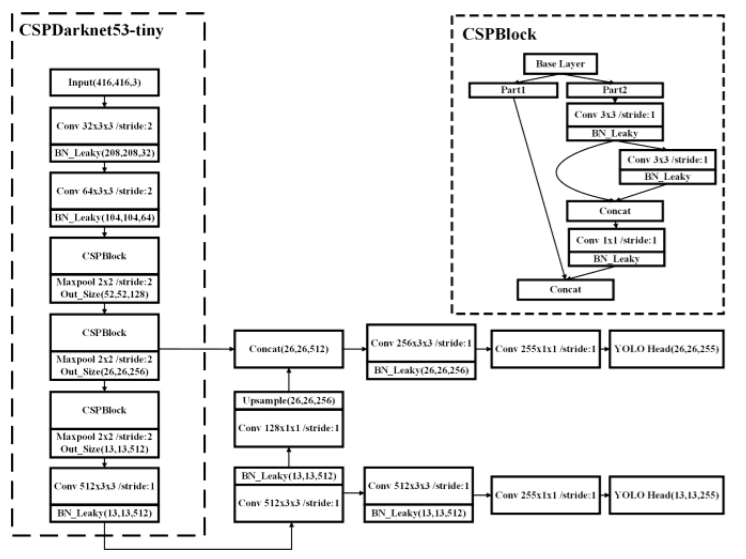
2.2.1.1. Giới thiệu bài toán phát hiện vật thể

* Object detection là bài toán phát hiện vật thể thuộc lớp nhất định trong ảnh.
* Đầu vào của bài toán phát hiện vật là một bức ảnh. Chúng ta không chỉ phải phân loại được object trên bức ảnh mà còn phải định vị được vị trí của đối tượng đó. Các phương pháp trước YOLOv1 thường sử dụng 2 bước: bước (1) thường sử dụng sliding window để lấy các vùng khác nhau của bức ảnh, hoặc sử dụng một thuật toán lựa chọn các vùng ứng viên (có thể chứa vật), tiếp theo đó, bước (2) sẽ phân loại các vị trí này xem vật đó thuộc lớp nào. Các cách tiếp cận này có nhược điểm là yêu cầu một lượng tính toán lớn, và bị phân nhỏ thành nhiều bước, khó có thể tối ưu về mặt tốc độ
* You only look once (YOLO) là một mô hình CNN để detect object mà một ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình cũ. Một trong nhưng ưu điểm mà YOLO đem lại đó là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ object box chứa các đối tượng, mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descent
* Dù đều được gọi là YOLO, các phiên bản của mô hình này đều có những cải tiến rất đáng kể sau mỗi phiên bản. Sau 3 phiên bản của tác giả chính [Joseph Redmon](https://pjreddie.com/) là YOLOv1 đến v3, tính đến thời điểm hiện tại có thêm một paper [YOLOv4](https://arxiv.org/pdf/2004.10934.pdf) của Alexey Bochkovskiy được dẫn link trực tiếp từ [repository gốc của Joseph Redmon](https://github.com/pjreddie/darknet) và [YOLOv5](https://github.com/ultralytics/yolov5) đang được phát triển

#### 2.2.1.2. YOLO

a, Yolotinyv4

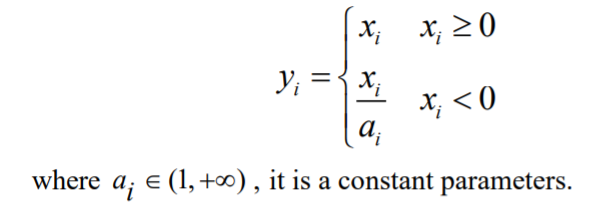
* YOLOv4-tiny được đề xuất dựa trên YOLOv4 để đơn giản cấu trúc mạng và giảm tham số, phù hợp phát triển trên các thiết bị mobile và nhúng.
* YOLO cho kết quả real-time khi chạy trên các nền tảng GPU cao cấp, cho nên không phù hợp để chạy real-time với các thiết bị mobiles và thiết bị nhúng do các thiết bị này bị hạn chế về khả năng tính toàn và bộ nhớ giới hạn. Để giải quyết vấn đề này, phương pháp lightweight object detection được đề xuất với kiến trúc mạng đơn giản hơn và ít tham số hơn. Do đó yêu cầu nguồn tính toán và bộ nhớ thấp hơn, tốc độ nhanh hơn tuy nhiên thì khả năng phát hiện đối tượng có độ chính xác thấp hơn.
* Một số phương pháp lightweight object detection được đề xuất như là MobileNet series, Squeezenet series, ShuffleNet series, lightweight YOLO series
* Các phương pháp lightweight YOLO được thiết kế dựa trên YOLO đầy đủ.
* YOLOv2 sử dụng mạng backbone là Darknet19, chứa 19 lớp tích chập và 6 lớp pooling. Phương pháp YOLOv2-tiny xóa các lớp tích chập còn 9 lớp để giảm độ phức tạp của mạng
* YOLOv3-tiny sử dụng 7 mạng tích chập và 6 lớp max pooling thay cho kiến trúc ResBlock trong mạng Darknet53 của YOLOv3. Nó cũng giảm đầu ra tự 3 dự đoán tỉ lệ 52x52, 26x26, 13x13 thành 2 dự đoán tỉ lệ 26x26, 13x13.
* YOLOv4-tiny cũng là 1 trong những phương pháp thuộc lightweight YOLO series. Nó sử dụng mạng backbone CSPDarknet53-tiny thay vì CSPDarknet53 của YOLOv4. The spatial pyramid pooling (SPP) và path aggregation network (PANet) cũng thay cho feature pyramid networks (FPN) để giảm thời gian detection. Bên cạnh đó, YOLOv4-tiny cũng sử dụng 2 dự đoán tỉ lệ 26x26 và 13x13.
* Kiến trúc mạng



Kiến trúc mạng Yolov4-tiny với ảnh đầu vào kích thước 416x416 và 80 classes.

Nguồn : https://arxiv.org/ftp/arxiv/papers/2011/2011.04244.pdf

* Backbone : CSPDarknet53-tiny
* CSPBlock module in cross stage partial network thay vì ResBlock module trong mạng phần dư. Nó có thể nâng cao khả năng học của mạng tích chập so với ResBlock module.
* Để quá trình tính toán đơn giản hơn, Yolov4-tiny sử dụng hàm kích hoạt LeakyReLU trong mạng CSPDarknet53-tiny mà không sử dụng Mish activation như ở Yolov4 :

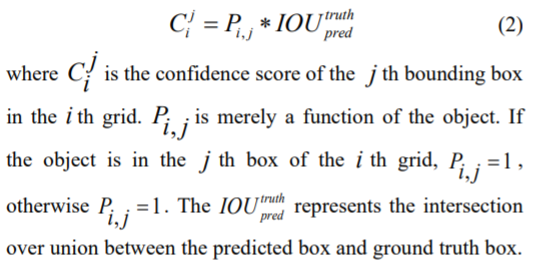


* Trong phần kết hợp đặc trưng, Yolov4-tiny sử dụng Feature pyramid network (FPN) để trích xuất đặc trưng với các tỉ lệ khác nhau để tăng tốc độ phát hiện đối tượng, mà không sử dụng the Spatial Pyramid Pooling (SPP) và Path Aggregation Network (PAN) trong phương pháp Yolov4. Yolov4-tiny sử dụng 2 bản đồ đặc trưng khác nhau là 13x13 và 26x26 để dự đoán kết quả phát hiện đối tượng.

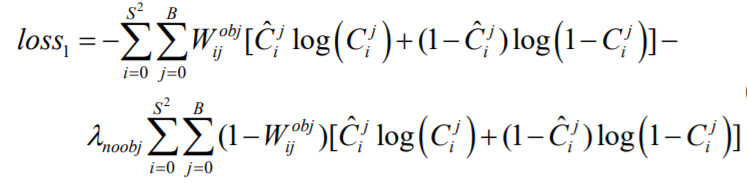
Quá trình dự đoán

* Quá trình dự đoán của phương pháp Yolov4-tiny giống phương pháp Yolov4. Đầu tiên nó cũng điều chỉnh kích thước tất cả ảnh đầu vào thành 1 kích thước cố định giống nhau. Thứ hai, ảnh đầu vào được chia thành các lưới có kích thước SxS, mỗi ô lưới sử dụng B bounding boxes để phát hiện đối tượng. Do đó, nó sinh ra ra SxSxB bounding boxes cho 1 ảnh đầu vào. Nếu trung tâm của vật thể nằm vào ô lưới nào thì những bounding boxes trong ô lưới đó sẽ dự đoán vật thể.
* Để giảm sự dư thừa của các bounding boxes trong quá trình dự đoán, ngưỡng tin cậy được đề xuất.

Độ tin cậy của bounding box được tính bằng :



* Loss function : giống Yolov4, gồm 3 phần:
  + - * + loss = loss1 + loss2 + loss3
    - The confidence loss function :

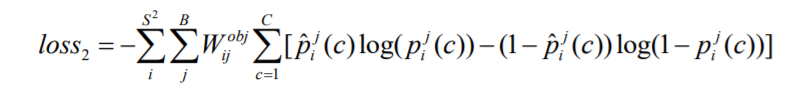


Trong đó : S2 là số lượng ô lưới trong ảnh đầu vào

* + - * + B là số lượng bounding box của 1 ô lưới
        + = 1 nếu bounding box thứ j của ô lưới thứ I chứa obj

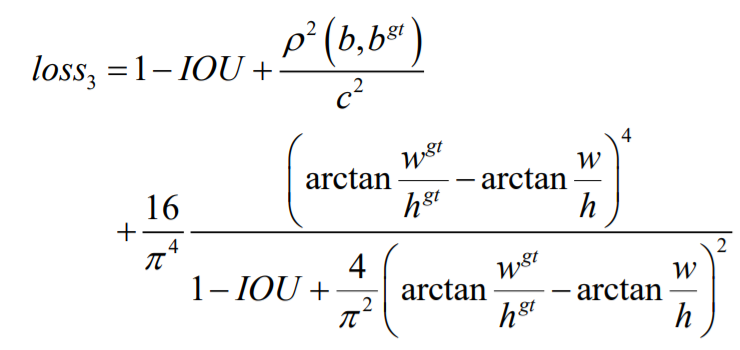
là độ tin cậy của box thật sự và box được dự đoán

* + - The classification loss function :



Trong đó là xác suất thực tế và xác suất dự đoán mà đối tượng thuộc về lớp phân loại c trong bounding box j của ô lưới i

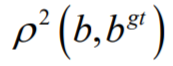
* + - The bounding box regression loss function :



Trong đó IOU là vùng chồng nhau giữa box dự đoán và box thực tế.

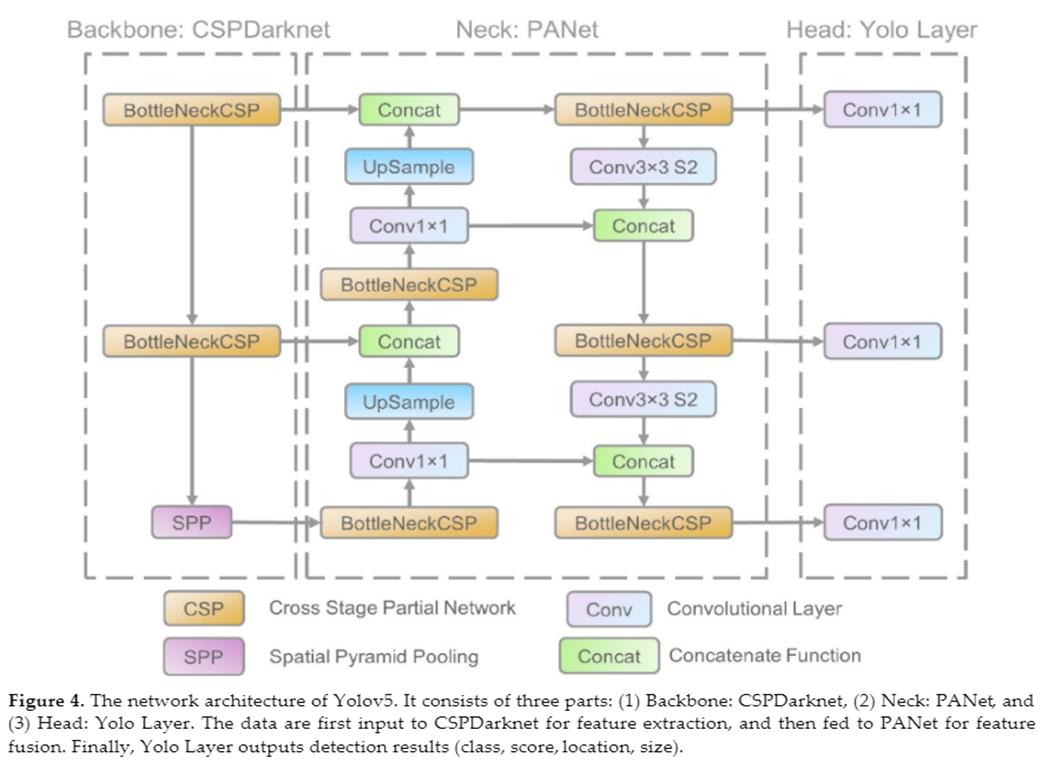
wgt  và hgt là chiều rộng và chiều cao thực tế của bounding box.

w và h là chiều rộng và chiều cao dự đoán của bounding box

 là khoảng cách Euclid giữa điểm trung tâm của bouding box dự đoán và bounding box thực tế.

c là khoảng cách đường chéo nhỏ nhất của box mà có thể chứa bounding box dự đoán và bounding box thực tế.

b, Yolov5



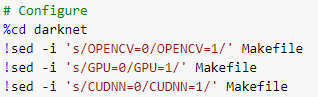
Nguồn : <https://machinelearningknowledge.ai/>

### 2.2.2. Áp dụng vào bài toán phát hiện xe trên cao tốc

a. Huấn luyện mô hình lại trên mạng Darknet sẵn có cho mode nhận diện xe đạp, ô tô, xe máy, xe tải

* Dataset: Bộ dữ liệu gồm 400 ảnh được gán nhãn tay cho 4 lớp xe đạp, ô tô, xe máy, xe tải
* B1: Clone, cấu hình & biên dịch Darknet

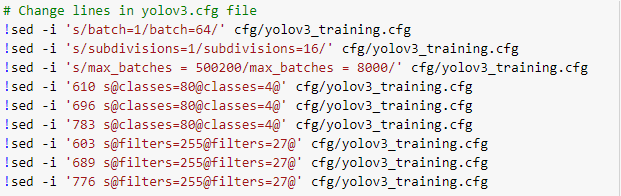






* B2: Cấu hình yolov3.cfg file





* B3: Tạo .names and .data files



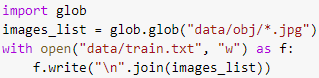
* B4: Lưu yolov3\_training.cfg và obj.names vào Google drive



* B5: Tạo thư mục và giải nén image dataset



* B6: Tạo train.txt file



* B7: Download pre-trained weights cho convolutional layers file



* B8: Bắt đầu quá trình training



b. Kết quả đạt được

* Khả năng nhận diện chưa cao với ảnh có độ phân giải thấp
* Thường nhầm lẫn giữa xe đạp và xe máy
* mAP = 60,311%

# Chương 3. Khó khăn và hướng phát triển

Trong quá trình thực hiện đồ án môn học, tôi gặp phải những khó khăn sau :

* Khó khăn trong việc tự học các kiến thức về Machine learning và Deep learning do đa số tài liệu là tiếng Anh, có những chỗ chưa được sáng tỏ và hiểu rõ
* Khó khăn trong việc chuẩn bị dữ liệu

Trong tương lai, tôi sẽ tìm hiểu các phương pháp, mô hình mới để bắt kịp với xu hướng để áp dụng vào các bài toán cụ thể

# Tài liệu tham khảo

1. <https://www.coursera.org/learn/machine-learning>
2. <https://nttuan8.com/>
3. <https://machinelearningcoban.com/>
4. <https://github.com/AlexeyAB/darknet>