Nếu hai lớp không thể phân tách tuyến tính thì sao?

¨ Khả năng phân tách tuyến tính là lý tưởng trong thực tế.

¨ Dữ liệu thường bị nhiễu hoặc sai lệch, tạo thành hai lớp chồng chéo (nhiễu/lỗi có thể làm 2 lớp giao nhau)

Trong trường hợp khả năng phân tách tuyến tính:

Text

Description automatically generated

Trong trường hợp nhiễu hoặc chồng chéo, những hạn chế đó có thể không bao giờ gặp nhau đồng thời.

¨ Điều đó có nghĩa là chúng ta không thể giải được w\* và b\*.

Giống như [Perceptron Learning Algorithm (PLA)](https://machinelearningcoban.com/2017/01/21/perceptron/), [Support Vector Machine (SVM) thuần](https://machinelearningcoban.com/2017/04/09/smv/) chỉ làm việc khi dữ liệu của 2 classes là [linearly separable](https://machinelearningcoban.com/2017/01/21/perceptron/#bai-toan-perceptron). Một cách tự nhiên, chúng ta cũng mong muốn rằng SVM có thể làm việc với dữ liệu gần linearly separable giống như [Logistic Regression](https://machinelearningcoban.com/2017/01/27/logisticregression/) đã làm được.

Điểm nhiễu xa và xb được đặt sai vị trí

Diagram

Description automatically generated

(Hình có nhiễu)

Chart, scatter chart

Description automatically generated

Có hai trường hợp dễ nhận thấy SVM làm việc không hiệu quả hoặc thậm chí không làm việc:

* Trường hợp 1: Dữ liệu vẫn *linearly separable* như Hình 1a) nhưng có một điểm *nhiễu* của lớp tròn đỏ ở quá gần so với lớp vuông xanh. Trong trường hợp này, nếu ta sử dụng SVM *thuần* thì sẽ tạo ra một *margin* rất nhỏ. Ngoài ra, đường phân lớp nằm quá gần lớp vuông xanh và xa lớp tròn đỏ. Trong khi đó, nếu ta *hy sinh* điểm nhiễu này thì ta được một *margin* tốt hơn rất nhiều được mô tả bởi các đường nét đứt. SVM *thuần* vì vậy còn được coi là *nhạy cảm với nhiễu* (*sensitive to noise*).
* Trường hợp 2: Dữ liệu không linearly separable nhưng gần linearly separable như Hình 1b). Trong trường hợp này, nếu ta sử dụng SVM thuần thì rõ ràng bài toán tối ưu là infeasible, tức feasible set là một tập rỗng, vì vậy bài toán tối ưu SVM trở nên vô nghiệm. Tuy nhiên, nếu ta lại chịu hy sinh một chút những điểm ở gần biên giữa hai classes, ta vẫn có thể tạo được một đường phân chia khá tốt như đường nét đứt đậm. Các đường support đường nét đứt mảnh vẫn giúp tạo được một margin lớn cho bộ phân lớp này. Với mỗi điểm nằm lần sang phía bên kia của các đường suport (hay đường margin, hoặc đường biên) tương ứng, ta gọi điểm đó rơi vào vùng không an toàn. Chú ý rằng vùng an toàn của hai classes là khác nhau, giao nhau ở phần nằm giữa hai đường support.

Trong cả hai trường hợp trên, margin tạo bởi đường phân chia và đường nét đứt mảnh còn được gọi là soft margin (biên mềm). Cũng theo cách gọi này, SVM thuần còn được gọi là Hard Margin SVM (SVM biên cứng).

Trong bài này, chúng ta sẽ tiếp tục tìm hiểu một biến thể của Hard Margin SVM có tên gọi là Soft Margin SVM.

Cách giải quyết thứ nhất là giải một bài toán tối ưu có ràng buộc bằng cách giải bài toán đối ngẫu giống như Hard Margin SVM; cách giải dựa vào bài toán đối ngẫu này là cơ sở cho phương pháp Kernel SVM cho dữ liệu thực sự không linearly separable mà tôi sẽ đề cập trong bài tiếp theo. Hướng giải quyết này sẽ được tôi trình bày trong Mục 3 bên dưới.

Cách giải quyết thứ hai là đưa về một bài toán tối ưu không ràng buộc. Bài toán này có thể giải bằng các phương pháp Gradient Descent. Nhờ đó, cách giải quyết này có thể được áp dụng cho các bài toán [large cale](https://machinelearningcoban.com/2017/01/12/gradientdescent/#large-scale). Ngoài ra, trong cách giải này, chúng ta sẽ làm quen với một hàm mất mát mới có tên là hinge loss. Hàm mất mát này có thể mở rộng ra cho bài toán multi-class classification mà tôi sẽ đề cập sau 2 bài nữa (Multi-class SVM). Cách phát triển từ Soft Margin SVM thành Multi-class SVM có thể so sánh với cách phát triển từ Logistic Regression thành [Softmax Regression](https://machinelearningcoban.com/2017/02/17/softmax/). Hướng giải quyết này sẽ được tôi trình bày trong Mục 4 bên dưới.

***2. Phân tích toán học***

Như đã đề cập phía trên, để có một margin lớn hơn trong Soft Margin SVM, chúng ta cần hy sinh một vài điểm dữ liệu bằng cách chấp nhận cho chúng rơi vào vùng không an toàn. Tất nhiên, chúng ta phải hạn chế sự hy sinh này, nếu không, chúng ta có thể tạo ra một biên cực lớn bằng cách hy sinh hầu hết các điểm. Vậy hàm mục tiêu nên là một sự kết hợp để tối đa margin và tối thiểu sự hy sinh.

Giống như với Hard Margin SVM, việc tối đa margin có thể đưa về việc tối thiểu ||w||2^2 để xác định sự hy sinh, chúng ta cùng theo dõi Hình 2 dưới đây:

Scatter chart

Description automatically generatedText, letter

Description automatically generated

Text

Description automatically generated

Với Soft Margin SVM, hàm mục tiêu sẽ có thêm một số hạng nữa giúp tối thiểu sự hy sinh. Từ đó ta có hàm mục tiêu:  
A picture containing Word

Description automatically generated

Text, letter

Description automatically generated