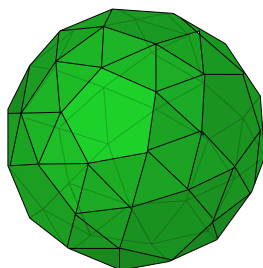


Projects in Mathematics and Applications

SUPPORT VECTOR MACHINE

Vũ Đình Bách ^{*} [†]Hoàng Khánh Linh
Phạm Tiến Sơn [‡] [§]Phan Trường Trí

Ngày 15 tháng 8 năm 2021



^{*} Trường Đại học Bách khoa Hà Nội
[†] Trường Phổ Thông Năng Khiếu - ĐHQG TP HCM
[‡] Trường Đại học Bách khoa Hà Nội
[§] Trường THPT Chuyên Bạc Liêu

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành nhất đến Ban Tổ chức và các anh chị mentor PiMA 2021. Mặc dù tình hình dịch bệnh còn đang diễn biến phức tạp nhưng mọi người đã tạo mọi điều kiện tốt nhất để chúng em có thể được tiếp cận với những kiến thức Toán học nói chung và Data Science nói riêng. Bên cạnh đó chúng em còn được rèn luyện và trau dồi các kỹ năng thiết yếu như làm việc nhóm, nghiên cứu, viết báo cáo...

Tiếp đến, chúng em muốn đặc biệt cảm ơn anh Vũ Lê Thế Anh và anh Nguyễn Đình Hoàng Phúc đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm việc. Nếu không có sự giúp đỡ của các anh, nhóm em đã không thể hoàn thành tốt phần thuyết trình và bài báo cáo này.

Chúng em cũng xin cảm ơn các diễn giả: chị Trịnh Kim Chi, anh Trần Quốc Cơ, anh Lê Thiện, thầy Hồ Phạm Minh Nhật, anh Nguyễn Xuân Khánh, anh Hồ Quốc Đăng Hưng và anh Phạm Tuấn Huy. Qua những chia sẻ và bài giảng của các diễn giả, chúng em đã có những cái nhìn sâu sắc hơn về mảng Data Science và tích lũy cho mình những kinh nghiệm trong cuộc sống.

Lời cuối, chúng em hy vọng PiMA sẽ luôn phát triển không ngừng trong thời gian tới để tiếp tục mang những kiến thức bổ ích đến các bạn học sinh vì đây là cơ hội đáng trân quý để chúng em có thể chuẩn bị tốt cho tương lai của mình.

Xin cảm ơn PiMA 2021 vì tất cả.

TÓM TẮT NỘI DUNG

Trong Machine Learning, Support Vector Machine (SVM) là mô hình học có giám sát, được sử dụng phổ biến trong bài toán phân loại dữ liệu. Bài báo cáo sau đây là cái nhìn tổng quan về chủ đề này. Bắt đầu với ý tưởng cơ bản trong phân loại nhị phân, ta đi xây dựng bài toán gốc của SVM. Trên thực tế để dễ dàng hơn trong việc tính toán, ta thường giải bài toán đối ngẫu thay vì bài toán gốc. Tiếp đến, ta xem xét các trường hợp mà dữ liệu là không tách biệt tuyến tính hay trường hợp xuất hiện các điểm nhiễu bằng phân loại biên mềm SVM và phương pháp Kernel. Cuối cùng, bản báo cáo sẽ giới thiệu bài toán phân loại nhiều lớp SVM và mô phỏng thuật toán SVM với dữ liệu cụ thể.

MỤC LỤC

Bảng ký hiệu	4
1 Nền tảng về tối ưu	5
1.1 Tối ưu lồi	5
1.2 Bài toán đối ngẫu	7
2 Phân loại nhị phân có giám sát	9
3 Bài toán gốc của SVM	10
3.1 Siêu phẳng phân tách	10
3.2 Bài toán tối ưu lồi của SVM	11
3.3 Phân loại biên cứng SVM	14
4 Bài toán đối ngẫu SVM	14
4.1 Kiểm tra điều kiện Slater	14
4.2 Hàm Lagrangian, đối ngẫu Lagrange của bài toán gốc	15
4.3 Dùng điều kiện KKT giải bài toán tối ưu mới	15
4.4 Những ưu điểm khi giải bài toán đối ngẫu	17
5 Phân loại biên mềm SVM	18
5.1 Động lực ra đời	18
5.2 Bài toán gốc biên mềm SVM	18
5.3 Bài toán đối ngẫu của biên mềm SVM	19
6 Phương pháp Kernel trong SVM	21
6.1 Động lực	21
6.2 Phương pháp kernel	22
6.3 Kernel SVM	23
7 Bài toán tối ưu mềm không ràng buộc	24
7.1 Xây dựng bài toán bài toán tối ưu mềm không ràng buộc	24
7.2 Áp dụng Stochastic Gradient Descent giải quyết bài toán	25
8 Multi-class SVM	26
9 Thực hành	27
9.1 Chạy trên dữ liệu tự sinh	27
9.2 Chạy trên dữ liệu MNIST	29
9.3 Cài đặt SVM biên cứng và SVM biên mềm bằng qpsolvers và cvxopt	31
Tài liệu	33

BẢNG KÝ HIỆU

Ký hiệu	Ý nghĩa
a, b, c	Đại lượng vô hướng được ký hiệu bằng chữ in thường
x, y, z	Vector được ký hiệu bằng các chữ in thường đậm
A, B, C	Ma trận được ký hiệu bằng các chữ in hoa đậm
x^T, A^T	Vector chuyển vị, ma trận chuyển vị
$\langle x, y \rangle$	Tích vô hướng của x và y
$B = [b_1, b_2, b_3]$	Ma trận được tạo bởi các vector cột
\mathbb{R}	Tập số thực
\mathbb{R}^n	Không gian Euclid n chiều
$\forall i$	Với mọi i
$x \succeq y$	Các phần tử của x không nhỏ hơn phần tử tương ứng của y
$x \preceq y$	Các phần tử của x không lớn hơn phần tử tương ứng của y
$a := b$	a được định nghĩa là b
\mathcal{X}	Tập hợp \mathcal{X}
$a \in A$	a là một phần tử của tập hợp A
D	Số chiều, trong $d = 1, \dots, D$
N	Số điểm dữ liệu, trong $n = 1, \dots, N$
I_N	Ma trận đơn vị kích thước $N \times N$
$0_{M \times N}$	Ma trận không kích thước $M \times N$
$1_{M \times N}$	Ma trận một kích thước $M \times N$
$\ \cdot\ $	Norm
$\sum_{n=1}^N x_n$	Tổng của $x_n : x_1 + \dots + x_N$
$\prod_{n=1}^N x_n$	Tích của $x_n : x_1 \cdot \dots \cdot x_N$
$\text{dom} f$	Tập xác định của hàm f
$\frac{\partial f}{\partial x}$	Đạo hàm riêng theo x của f
∇	Gradient
\mathcal{L}	Lagrangian
$\sup \mathcal{S}$	Cận trên đúng của tập số thực \mathcal{S}
$\inf \mathcal{S}$	Cận dưới đúng của tập số thực \mathcal{S}
$x^* = \arg \min_x f(x)$	Giá trị x^* mà hàm $f(x)$ đạt giá trị nhỏ nhất
SVM	Support Vector Machine
KKT	Karush-Kuhn-Tucker
SGD	Stochastic Gradient Descent
QP	Quadratic Programming

1 NỀN TẢNG VỀ TỐI ƯU

1.1 Tối ưu lồi

1.1.1 Hàm lồi, tập hợp lồi

Định nghĩa 1.1 (Tập hợp lồi). Một tập hợp \mathcal{D} được gọi là *tập lồi* nếu với bất kì $x_1, x_2 \in \mathcal{D}$ và bất kì số thực θ với $0 \leq \theta \leq 1$ ta có :

$$\theta x_1 + (1 - \theta)x_2 \in \mathcal{D}. \quad (1)$$

Tính chất 1.2. Giao của nhiều tập lồi là một tập lồi.

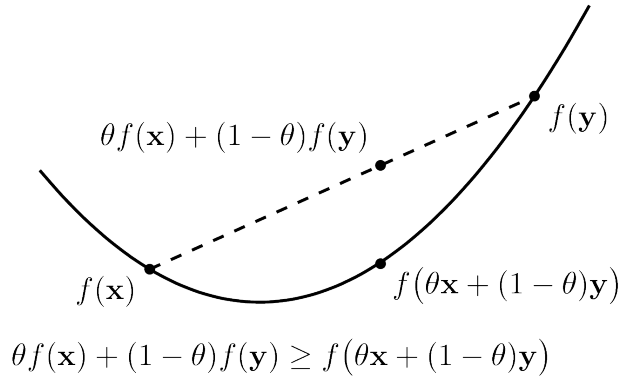
Chứng minh: Giả sử $\{D_\alpha\}_{\alpha \in A}$ là một trong các tập lồi và $D := \bigcap_{\alpha \in A} D_\alpha$. Với bất kì $x_1, x_2 \in D$ ta luôn có $x_1, x_2 \in D_\alpha$ với mọi $\alpha \in A$. Từ định nghĩa ta có: $\theta x_1 + (1 - \theta)x_2 \in D_\alpha$ với mọi $\alpha \in A$ và $0 \leq \theta \leq 1$. Suy ra $\theta x_1 + (1 - \theta)x_2 \in D$.

Vậy giao của nhiều tập lồi là một tập lồi. ■

Định nghĩa 1.3 (Hàm lồi). Hàm $f : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ được gọi là hàm lồi nếu \mathcal{X} là một tập lồi và thỏa mãn *bất đẳng thức Jensen*:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y), \quad (2)$$

với $x, y \in \mathcal{X}, 0 \leq \theta \leq 1$.



Hình 1: Bất kỳ đoạn thẳng nào nối hai điểm của hàm lõm đều nằm ở phía trên đồ thị hàm

Ví dụ 1.4. Mọi norm và hàm max trên \mathbb{R}^n đều là hàm lồi.

Chứng minh: Giả sử một norm $f : \mathbb{R}^n \rightarrow \mathbb{R}$ và $0 \leq \theta \leq 1$ thì:

$$\begin{aligned} f(\theta x + (1 - \theta)y) &\leq f(\theta x) + f((1 - \theta)y) && \text{(Bất đẳng thức tam giác)} \\ &= \theta f(x) + (1 - \theta)f(y). && \text{(Do } \theta \text{ là đại lượng vô hướng)} \end{aligned}$$

Giả sử hàm $h(x) = \max_i x_i$ thỏa mãn $0 \leq \theta \leq 1$ thì:

$$h(\theta x + (1 - \theta)y) = \max_i (\theta x_i + (1 - \theta)y_i) \quad (3)$$

$$\leq \theta \max_i x_i + (1 - \theta) \max_i y_i \quad (4)$$

$$= \theta h(x) + (1 - \theta)h(y). \quad (5)$$

Vậy ta có điều phải chứng minh. ■

Định nghĩa 1.5 (Hàm lõm). Một hàm là lõm nếu bất kỳ đoạn thẳng nào ở giữa, nối hai điểm của hàm lõm đều nằm ở phía dưới đồ thị hàm.

Định nghĩa 1.6 (Hàm lồi chặt). Hàm $f : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ được gọi là *hàm lồi chặt* nếu \mathcal{X} là một tập lồi và thỏa mãn:

$$f(\theta x + (1 - \theta)y) < \theta f(x) + (1 - \theta)f(y), \quad (6)$$

với $x, y \in \mathcal{X}, x \neq y, 0 \leq \theta \leq 1$.

Tính chất 1.7. Hàm lồi đạt giá trị nhỏ nhất tại các cực tiểu của nó. Hàm lồi chặt đạt giá trị nhỏ nhất tại duy nhất một điểm.

Chứng minh: Giả sử tập hợp các điểm cực tiểu của hàm số là không rỗng và tồn tại $\hat{x} \in \mathcal{X}$ là một cực tiểu địa phương của hàm f , một điểm $x^* \in \mathcal{X}$ sao cho $f(x^*) < f(\hat{x})$. Với $0 < \theta < 1$:

$$f(\theta x^* + (1 - \theta)\hat{x}) \leq \theta f(x^*) + (1 - \theta)f(\hat{x}) < f(\hat{x})$$

mâu thuẫn với việc \hat{x} là điểm cực tiểu địa phương.

Còn đối với hàm lồi chặt, giả sử có hai điểm cực trị $x, y \in \mathcal{X}$ sao cho:

$$f(x) = f(y) \leq f(z), \forall z \in \mathbb{X}.$$

Xét $z = \frac{x+y}{2}$, ta có:

$$f(z) = f\left(\frac{x+y}{2}\right) < \frac{1}{2}f(x) + \frac{1}{2}f(y) = \frac{1}{2}f(x) + \frac{1}{2}f(x) = f(x)$$

mâu thuẫn với giả thiết trên.

Vậy ta có điều phải chứng minh. ■

1.1.2 Bài toán tối ưu tổng quát

Định nghĩa 1.8 (Bài toán tối ưu tổng quát). *Bài toán tối ưu tổng quát* có dạng:

$$\begin{aligned} \min \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{aligned} \quad (7)$$

với vector $x = (x_1, x_2, \dots, x_n)$ là *biến tối ưu*, hàm số $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ là *hàm mục tiêu*, các hàm số $f_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m; j = 1, \dots, p$ là các *hàm ràng buộc*.

Tập xác định của bài toán tối ưu: $\mathcal{D} = \bigcap_{i=1}^m \text{dom } f_i \cap \bigcap_{j=1}^p \text{dom } h_j$

Một điểm $x \in \mathcal{D}$ là *khả thi* nếu nó thỏa mãn các ràng buộc. Tập hợp tất cả các điểm khả thi được gọi là *tập khả thi*. Khi tập này rỗng thì ta gọi bài toán tối ưu là *bất khả thi*, ngược lại thì gọi là *bài toán tối ưu khả thi*. *Giá trị tối ưu* của bài toán tối ưu tổng quát được định nghĩa là:

$$p^* = \inf\{f_0(x) \mid f_i(x) \leq 0, i = 1, \dots, m; h_j(x) = 0, j = 1, \dots, p\}. \quad (8)$$

1.1.3 Bài toán tối ưu lồi và dạng đặc biệt quy hoạch toàn phương

Định nghĩa 1.9 (Bài toán tối ưu lồi). *Bài toán tối ưu lồi* có dạng:

$$\begin{aligned} \min \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{aligned} \quad (9)$$

với :

- Hàm mục tiêu phải là hàm lồi.
- Hàm f_i là những hàm lồi.
- h_j là các tập hợp lồi.

Tính chất 1.10. Tập khả thi của một bài toán tối ưu lồi là một tập hợp lồi.

Chứng minh: Dễ dàng nhận thấy vì tập khả thi của bài toán là giao của tập xác định \mathcal{D} mà \mathcal{D} là một tập lồi. ■

Định nghĩa 1.11 (Bài toán quy hoạch toàn phương). *Bài toán quy hoạch toàn phương* có dạng tiêu chuẩn:

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Px + q^T x + r \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b, \end{aligned}$$

với $P \in \mathbb{S}_+^n$ (Tập các ma trận nửa xác định dương).

1.2 Bài toán đối ngẫu

1.2.1 Lagrangian và hàm đối ngẫu Lagrange

Xét bài toán tối ưu tổng quát:

$$\begin{aligned} \min \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{aligned} \quad (10)$$

với biến $x \in \mathbb{R}^n$ và tập xác định $\mathcal{D} = \bigcap_{i=1}^m \text{dom } f_i \cap \bigcap_{j=1}^p \text{dom } h_j$ không rỗng.

Định nghĩa 1.12 (Lagrangian). Ta định nghĩa *Lagrangian* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ ứng với bài toán trên là:

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^p \nu_j h_j(x), \quad (11)$$

với $\text{dom } \mathcal{L} = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. Trong đó λ_i, ν_j được gọi là các *nhân tử Lagrange*, hay *biến số đối ngẫu*.

Lấy cận trên đúng của Lagrangian ta sẽ có bài toán tối ưu ban đầu:

$$\begin{aligned}\sup_{\lambda \geq 0} \mathcal{L}(x, \lambda, \nu) &= \sup_{\lambda \geq 0} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^p \nu_j h_j(x) \right) \\ &= \begin{cases} f_0(x) & \text{khi } f_i(x) \leq 0 \forall i, h_j(x) = 0 \forall j \\ \infty & \text{các trường hợp khác} \end{cases}\end{aligned}$$

Tương đương bài toán tối ưu gốc:

$$p^* = \inf_x \sup_{\lambda \geq 0} \mathcal{L}(x, \lambda, \nu).$$

Định nghĩa 1.13 (Hàm đối ngẫu Lagrange). Ta định nghĩa *hàm đối ngẫu Lagrange* $g : \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ là cận dưới đúng của Lagrangian với $\lambda \in \mathbb{R}^m, \nu \in \mathbb{R}^p$:

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^p \nu_j h_j(x) \right). \quad (12)$$

Nếu Lagrangian không bị chặn dưới, hàm đối ngẫu tại λ, ν sẽ lấy giá trị $-\infty$.

Định nghĩa 1.14 (Bài toán đối ngẫu Lagrange). *Bài toán đối ngẫu Lagrange* có dạng:

$$\max g(\lambda, \nu) \quad (13)$$

$$\text{subject to } \lambda \succeq 0. \quad (14)$$

Ta gọi (λ^*, ν^*) là nhân tử Lagrange tối ưu nếu chúng tối ưu với bài toán. Bài toán đối ngẫu Lagrange là một toán tối ưu lồi vì:

1. Hàm Lagrangian là một hàm bậc nhất với λ, ν nên là một hàm lồi. Hàm đối ngẫu Lagrange là một cận dưới đúng của một hàm lồi nên $g(\lambda, \nu)$ là một hàm lồi.
2. Các ràng buộc là lồi.

Giá trị tối ưu của bài toán đối ngẫu Lagrange là:

$$d^* = \sup_{\lambda \geq 0} \inf_x \mathcal{L}(x, \lambda, \nu).$$

Ta thấy với mọi $f : W \times Z \rightarrow \mathbb{R}$ thì

$$\sup_{z \in Z} \inf_{w \in W} f(w, z) \leq \inf_{w \in W} \sup_{z \in Z} f(w, z).$$

Từ đó suy ra

$$d^* = \sup_{\lambda \geq 0} \inf_x \mathcal{L}(x, \lambda, \nu) \leq \inf_x \sup_{\lambda \geq 0} \mathcal{L}(x, \lambda, \nu) = p^*.$$

Tính chất 1.15 (Tính chất đối ngẫu yếu). Với mọi bài toán tối ưu, ta có tính chất *đối ngẫu yếu*:

$$d^* \leq p^*. \quad (15)$$

1.2.2 Đối ngẫu chặt và định lý Slater

Khi $d^* = p^*$ thì có *đối ngẫu chặt*. Khi đối ngẫu chặt xảy ra thì việc giải bài toán đối ngẫu sẽ giúp ta tìm được giá trị tối ưu của bài toán gốc. Đối ngẫu chặt sẽ xảy ra trong bài toán tối ưu nếu điều kiện dưới đây được thỏa mãn:

Định lý 1.16 (Định lý Slater). *Nếu tồn tại một điểm x trong tập khả thi, thỏa mãn $f_i(x) < 0$ với $i = 1, 2, \dots, m$ và bài toán gốc là bài toán tối ưu lồi thì đối ngẫu chặt thỏa mãn.*

1.2.3 Điều kiện Karush-Kuhn-Tucker (KKT) cho bài toán tối ưu lồi

Trong trường hợp bài toán là bài toán tối ưu lồi và đối ngẫu chặt thỏa mãn thì KKT là điều kiện cần và đủ của nghiệm:

1. $f_i(x^*) \leq 0$ với $i = 1, 2, \dots, m$.
2. $h_j(x^*) = 0$ với $j = 1, 2, \dots, p$.
3. $\lambda^* \geq 0$ với $i = 1, 2, \dots, m$.
4. $\lambda^* f_i(x^*) = 0$.
5. $\nabla f(x^*) + \sum_{i=1}^m \lambda_i \nabla f_i(x^*) + \sum_{j=1}^p v_j \nabla h_j(x^*) = 0$ với $i = 1, 2, \dots, m$ và $j = 1, 2, \dots, p$.

Chứng minh: Vì 4 điều kiện còn lại của điều kiện KKT có thể dễ dàng thấy được nên việc chứng minh điều kiện này chỉ tập trung vào chứng minh điều kiện : $\lambda^* f_i(x^*) = 0$ với $i = 1, 2, \dots, m$.

Giả sử x^* là biến tối ưu ban đầu và λ^*, v^* là nhân tử Lagrange tối ưu. Ta có:

$$\begin{aligned}
 f_0(x^*) &= g(\lambda^*, v^*) = \inf_x \mathcal{L}(x^*, \lambda^*, \theta^*) \quad (\text{Đối ngẫu chặt và theo định nghĩa Lagrangian}) \\
 &\leq \mathcal{L}(x^*, \lambda^*) \\
 &= f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{j=1}^p v_j^* h_j(x^*) \\
 &\leq f_0(x^*) \quad (\text{Vì ràng buộc } \lambda^* f_i(x^*) \leq 0 \text{ và } h_j(x^*) = 0)
 \end{aligned}$$

Từ trên ta suy ra mỗi phần tử trong $\sum_{i=1}^m \lambda_i^* f_i(x^*)$ phải bằng 0 và chính là điều phải chứng minh. ■

2 PHÂN LOẠI NHỊ PHÂN CÓ GIÁM SÁT

Trong nhiều trường hợp, chúng ta muốn mô hình máy học phân loại dữ liệu vào các lớp đã cho trước. Những bài toán phân loại này có thể dễ dàng quan sát thấy trong các ví dụ:

1. Gmail phải phân loại giúp người dùng email nào là email rác, chứa những thông tin quảng cáo hoặc quan trọng đến người dùng để tạo trải nghiệm sử dụng tốt nhất cho người sử dụng.
2. Hệ thống nhận diện gian lận tín dụng phải xác định xem trong hàng triệu các giao dịch diễn ra hàng ngày, giao dịch nào có dấu hiệu gian lận dựa trên các thông tin như địa chỉ IP người dùng, lịch sử giao dịch, ...

3. Dựa trên một loạt các kết quả xét nghiệm và những thông tin về sức khỏe, một mô hình sẽ đưa ra dự đoán rằng một người có mắc một bệnh nào đó không.

Phân loại nhị phân thuộc lớp các bài toán *Học có giám sát*, dự đoán đầu ra một dữ liệu mới dựa trên các cặp (*dữ liệu, nhãn*) đã biết trước. Giá trị của nhãn của bài phân loại nhị phân có tính nhị phân.

Một cách toán học, từ *tập huấn luyện* (đã biết trước) $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, trong đó $x_i \in \mathbb{R}^D$ và $y_i \in \{-1, +1\}$, ta đi tìm một hàm số $f: \mathbb{R}^D \rightarrow \{-1, +1\}$ sao cho:

$$f(x_i) = y_i \quad \forall i = 1, \dots, N \quad (16)$$

Với một điểm dữ liệu mới z , ta có thể đưa ra dự đoán về nhãn tương ứng của nó $y = f(z)$.

Trong những phần tới của báo cáo, chúng tôi sẽ trình bày một hướng tiếp cận của bài toán phân loại nhị phân là mô hình *máy vector hỗ trợ* (SVM). SVM là một trong những thuật toán phân loại phổ biến nhất. SVM có một nền tảng lý thuyết mạnh và trong nhiều lĩnh vực được áp dụng SVM mang lại kết quả ấn tượng hơn so với các thuật toán khác.

3 BÀI TOÁN GỐC CỦA SVM

3.1 Siêu phẳng phân tách

Định nghĩa 3.1 (Siêu phẳng). Một *siêu phẳng* là một tập hợp có dạng:

$$\{x \mid \langle a, x \rangle = b\}. \quad (17)$$

Trong đó $a \in \mathbb{R}^D$, $a \neq 0$ và $b \in \mathbb{R}$.

Ví dụ 3.2.

Trong \mathbb{R}^2 , một siêu phẳng có dạng: $\{(x_1, x_2) \mid a_1x_1 + a_2x_2 = b\}$.

Trong \mathbb{R}^3 , một siêu phẳng có dạng: $\{(x_1, x_2, x_3) \mid a_1x_1 + a_2x_2 + a_3x_3 = b\}$.

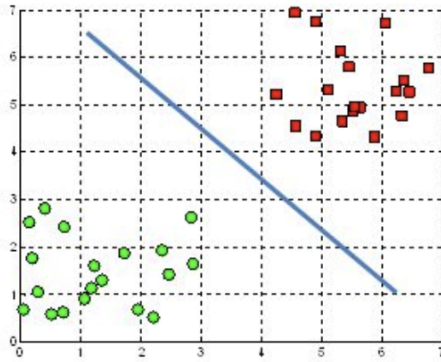
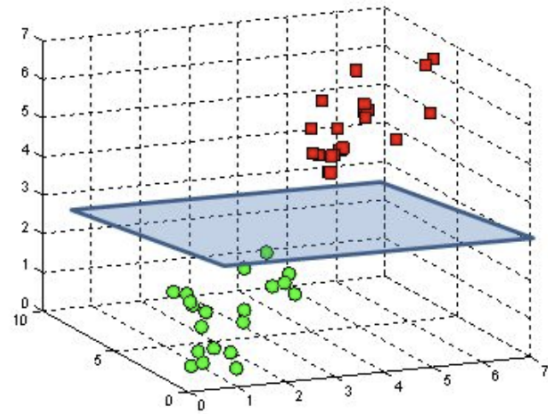
Định nghĩa 3.3 (Nửa không gian). Một siêu phẳng chia không gian \mathbb{R}^n thành hai nửa. Một *nửa không gian đóng* là một tập hợp có dạng:

$$\{x \mid \langle a, x \rangle \leq b\}. \quad (18)$$

Biên của nửa không gian (18) là một siêu phẳng $\{x \mid \langle a, x \rangle = b\}$. Tập hợp có dạng $\{x \mid \langle a, x \rangle < b\}$ gọi là *nửa không gian mở*.

Định nghĩa 3.4 (Tính tách biệt tuyến tính). Với $\mathcal{S} = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ là tập điểm dữ liệu và nhãn tương ứng, trong đó $x_i \in \mathbb{R}^D$ và $y_i \in \{-1, +1\}$. Tập \mathcal{S} có tính *tách biệt tuyến tính* nếu tồn tại siêu phẳng $\langle w, x \rangle + b = 0$ sao cho $y_i = \text{sign}(\langle w, x \rangle + b)$. Điều kiện này có thể viết lại thành:

$$y_i(\langle w, x \rangle + b) > 0 \quad \forall i = 1, \dots, N. \quad (19)$$

A hyperplane in \mathbb{R}^2 is a line

 A hyperplane in \mathbb{R}^3 is a plane


Hình 2: Trong không gian \mathbb{R}^2 siêu phẳng có dạng là một đường thẳng và trong \mathbb{R}^3 , siêu phẳng là một mặt phẳng.

Có thể hiểu định nghĩa 3.4 có nghĩa là tồn tại siêu phẳng $\langle w, x \rangle + b = 0$ phân tách không gian dữ liệu \mathbb{R}^d thành hai nửa không gian $\langle w, x \rangle + b > 0$ của những điểm dữ liệu thuộc lớp dương và $\langle w, x \rangle + b < 0$ của những điểm thuộc lớp âm.

Trong phân loại nhị phân, nếu tập dữ liệu huấn luyện có tính tách biệt tuyến tính thì ta có thể xem xét lớp hàm số

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \quad (20)$$

$$x \mapsto f(x) := \langle w, x \rangle + b, \quad (21)$$

được tham số hóa bởi $w \in \mathbb{R}^D$ và $b \in \mathbb{R}$. Do tính chất tách biệt tuyến tính của tập huấn luyện, tồn tại (w, b) sao cho

$$\text{sign}(f(x_i)) = y_i \quad \forall i = 1, \dots, N. \quad (22)$$

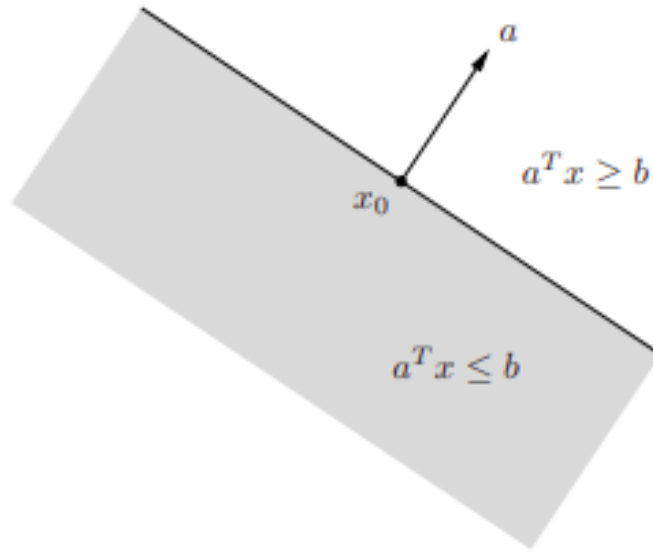
Với một điểm dữ liệu mới z , có thể tính nhãn tương ứng của z bằng $y = \text{sign}(f(z))$.

Có thể thấy rằng trong trường hợp dữ liệu tách biệt tuyến tính, có vô số siêu phẳng phân tách. Tuy nhiên *khả năng tổng quát hóa* của mô hình, làm việc tốt với những điểm dữ liệu mới, phụ thuộc vào vị trí của siêu phẳng phân tách mà mô hình lựa chọn.

Ví dụ 3.5. Xem xét ví dụ về một tập dữ liệu tách biệt tuyến tính ở hình 4. Cả hai đường gạch đen và đường xanh đều phân chia hai lớp điểm về hai nửa không gian riêng biệt nhưng trực quan của ta thấy rằng đường gạch đen sẽ là đường thẳng phân tách tốt hơn. Để giải thích cho nhận xét trực quan này ta giới thiệu khái niệm *lề* ở mục tiếp theo.

3.2 Bài toán tối ưu lề của SVM

Lề là khoảng cách từ siêu phẳng phân tách tới điểm dữ liệu gần nhất, với giả thiết bộ dữ liệu có tính tách biệt tuyến tính. Siêu phẳng có lề rộng hơn sẽ mang lại hiệu ứng phân lớp



Hình 3: Siêu phẳng $\langle a, x \rangle = b$ trong \mathbb{R}^2 chia không gian thành hai nửa. Nửa không gian $\langle a, x \rangle \geq b$ (không tô đậm) là nửa không gian theo chiều vector a . Nửa không gian $\langle a, x \rangle \leq b$ (không tô đậm) là nửa không gian ngược chiều vector a .

tốt hơn vì sự phân chia giữa hai lớp là rạch ròi hơn. Bài toán tối ưu trong SVM chính là bài toán đi tìm siêu phẳng phân cách có lề rộng nhất.

Vì mục tiêu của chúng ta là tìm một siêu phẳng phân tách $\langle w, x \rangle + b = 0$ sao cho lề của nó là lớn nhất, ta có thể cố gắng tham số khoảng cách này theo w, b . Nhận xét rằng ta có thể thay vector w bằng kw và b bằng kb mà không làm thay đổi mặt phân tách, do đó ta chọn w và b sao cho $\langle w, x_a \rangle + b = 1$ với x_a là điểm dữ liệu gần nhất.

Không mất tính tổng quát giả sử x_a ở phía dương. Nếu ta kí hiệu r là lề của siêu phẳng phân tách, r cũng chính khoảng cách giữa siêu phẳng với x_a , ta có:

$$r = \frac{|\langle w, x_a \rangle + b|}{\|w\|} = \frac{1}{\|w\|}. \quad (23)$$

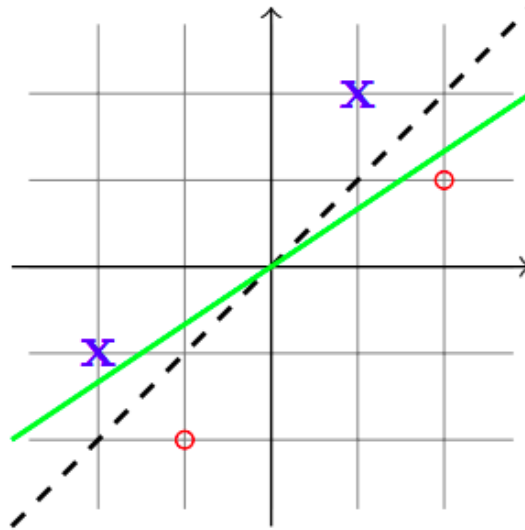
Do x_a là điểm dữ liệu gần siêu phẳng nhất nên các điểm dữ liệu còn lại phải cách siêu phẳng một khoảng lớn hơn hoặc bằng $r = \frac{1}{\|w\|}$:

$$\frac{y_n(\langle w, x_n \rangle + b)}{\|w\|} \geq \frac{1}{\|w\|} \Leftrightarrow y_n(\langle w, x_n \rangle + b) \geq 1. \quad (24)$$

Kết hợp mục tiêu tối đa lề với điều kiện các điểm dữ liệu các điểm dữ liệu phân lớp đúng, bài toán tối ưu của chúng ta trở thành:

$$\max_{w, b} \frac{1}{\|w\|} \quad (25)$$

$$\text{subject to } y_n(\langle w, x_n \rangle + b) \geq 1 \quad \forall n = 1, \dots, N. \quad (26)$$



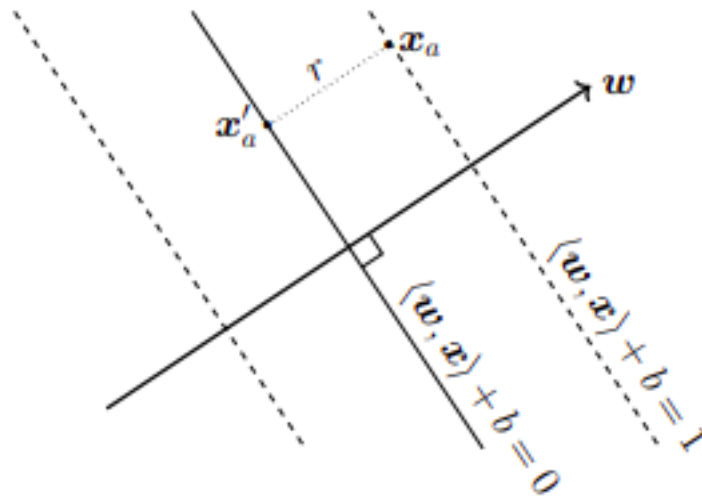
Hình 4: Hai đường thẳng gạch đen và xanh lá đều có thể phân tách những điểm dữ liệu đỏ và xanh lam về hai nửa không gian riêng biệt.

Ta có thể giải bài toán có nghiệm tương đương nhưng thuận tiện trong hơn tính toán:

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad (27)$$

$$\text{subject to } -y_n(\langle w, x_n \rangle + b) \leq -1 \quad \forall n = 1, \dots, N. \quad (28)$$

Nhận xét :



Hình 5: Bài toán tối ưu lề SVM trong \mathbb{R}^2 . Khoảng cách giữa hai siêu phẳng $\langle w, x \rangle + b = 0$ và $\langle w, x \rangle + b = 1$ là lề cần tối ưu.

1. Hàm chúng ta cần tối ưu là norm nên hàm mục tiêu là hàm lồi hơn nữa hàm mục tiêu có dạng $w^T I w$ với I là ma trận đơn vị nên xác định dương.
2. Các ràng buộc là tuyến tính với w, b .

Vậy bài toán tối ưu 27 có dạng quy hoạch toàn phương.

3.3 Phân loại biên cứng SVM

Phương trình 27 được biết đến là bài toán tối ưu của mô hình *phân loại biên cứng SVM* bởi vì điều kiện 28 yêu cầu các điểm dữ liệu phải được phân tách đúng ở hai nửa không gian hai bên lề là $\langle \mathbf{w}, \mathbf{x} \rangle + b \geq 1$ và $\langle \mathbf{w}, \mathbf{x} \rangle + b \leq -1$.

Từ đây mô hình bài toán biên cứng SVM:

Algorithm 1 Giải biên cứng SVM

Input: Tập dữ liệu $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$ và tập nhãn tương ứng $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^T \in \mathbb{R}^{N \times 1}$.

Output: \mathbf{w} và b

- 1: $\mathbf{Z} \leftarrow [\mathbf{X} \ \mathbf{1}]_{N \times (D+1)}$
 - 2: $\mathbf{G} \leftarrow -[y_j[\mathbf{Z}]_{jk}]_{jk}$
 - 3: $\mathbf{h} \leftarrow -\mathbf{1}_{N \times 1}$
 - 4: $\mathbf{q} \leftarrow \mathbf{0}_{(D+1) \times 1}$
 - 5: $\mathbf{P} \leftarrow \begin{pmatrix} \mathbf{I}_D & \mathbf{0}_{D \times 1} \\ \mathbf{0}_{1 \times D} & 0 \end{pmatrix}_{(D+1) \times (D+1)}$
 - 6: **solve QP with** $(\mathbf{P}, \mathbf{q}, \mathbf{G}, \mathbf{h}) \rightarrow \mathbf{S}$
 - 7: $\mathbf{w} \leftarrow \mathbf{S}[:, D]$
 - 8: $b \leftarrow \mathbf{S}[-1]$
 - 9: **return** \mathbf{w}, b
-

Algorithm 2 Truy vấn mô hình biên cứng SVM

Input: Điểm dữ liệu mới $\mathbf{z} \in \mathbb{R}^D, \mathbf{w}, b$

Output: Nhãn dự đoán y của \mathbf{z}

- 1: **if** $\langle \mathbf{w}, \mathbf{z} \rangle + b < 0$ **then**
 - 2: $y = -1$
 - 3: **else**
 - 4: $y = 1$
 - 5: **end if**
-

4 BÀI TOÁN ĐỐI NGẪU SVM

4.1 Kiểm tra điều kiện Slater

Với giả sử dữ liệu là tách biệt tuyến tính nên tập khả thi của bài toán là khác rỗng, ta luôn tìm được (\mathbf{w}_0, b_0) sao cho :

$$\begin{aligned} y_n(\langle \mathbf{w}_0, \mathbf{x}_n \rangle + b_0) &\geq 1 \quad \forall n = 1, \dots, N \\ \Leftrightarrow y_n(\langle 2\mathbf{w}_0, \mathbf{x}_n \rangle + 2b_0) &\geq 2 \quad \forall n = 1, \dots, N \end{aligned}$$

Chọn $(\mathbf{w}_1, b_1) = (2\mathbf{w}_0, 2b_0)$ ta có :

$$y_n(\langle \mathbf{w}_1, \mathbf{x}_n \rangle + b_1) > 1 \quad \forall n = 1, \dots, N.$$

Vậy định lý Slater thỏa mãn cho bài toán này.

4.2 Hàm Lagrangian, đối ngẫu Lagrange của bài toán gốc

Hàm Lagrangian ứng với bài toán lẻ cứng là :

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \alpha_n (y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) - 1), \quad (29)$$

với $\alpha_i \geq 0 \forall i = 1, \dots, N$ là nhân tử Lagrange tương ứng với mỗi ràng buộc của bài toán gốc.

Hàm đối ngẫu Lagrange chính là cận dưới đúng của Lagrangian:

$$\mathbf{g}(\alpha) = \inf_{\mathbf{w}, b} \left(\frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \alpha_n (y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) - 1) \right). \quad (30)$$

Từ đây ta có bài toán đối ngẫu tương ứng với bài toán gốc 27:

$$\alpha^* = \arg \max_{\alpha} \left(\inf_{\mathbf{w}, b} \left(\frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \alpha_n (y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) - 1) \right) \right). \quad (31)$$

4.3 Dùng điều kiện KKT giải bài toán tối ưu mới

Bài toán chúng ta là một bài toán lồi và đối ngẫu chặt xảy ra nên ta có điều kiện KKT trở thành điều kiện cần và đủ của nghiệm. Ta có hệ điều kiện cho α^* , \mathbf{w}^* và b^* :

$$1 - y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \leq 0, \quad (32)$$

$$\alpha_n \geq 0, \quad (33)$$

$$\alpha_n (1 - y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b)) = 0, \quad (34)$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n = 0, \quad (35)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^N \alpha_n y_n = 0. \quad (36)$$

Nhận xét 4.1. Dựa vào điều kiện thứ vị 34 và 83, ta có nhận xét:

1. Ta thấy giá trị tối ưu \mathbf{w}^* có dạng tổ hợp tuyến tính của các điểm dữ liệu:

$$\mathbf{w}^* = \sum \alpha_i^* y_i \mathbf{x}_i. \quad (37)$$

2. Từ 34, nếu $\alpha^* = 0$ là những điểm không đóng góp vào việc tính \mathbf{w}^* .
3. Nếu $1 - y_n (\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b^*) = 0 \Rightarrow \langle \mathbf{w}^*, \mathbf{x}_n \rangle + b^* = y_n$ tức là những điểm nằm trên lề và đóng góp vào việc tính toán \mathbf{w}^*, b^* do $\alpha_i^* \neq 0$. Các điểm này giúp kiến tạo siêu mặt phẳng phân cách nên được gọi là các vector hỗ trợ. Và từ đó, cái tên máy vector hỗ trợ ra đời.

Thay điều kiện 83 và 36 vào hàm Lagrangian, hàm đối ngẫu Lagrange trở thành:

$$\begin{aligned} \mathbf{g}(\alpha) &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ &\quad - b \sum_{i=1}^N y_i \alpha_i + \sum_{i=1}^N \alpha_i \end{aligned} \quad (38)$$

$$= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^N \alpha_i. \quad (39)$$

Ở phần 1.2.1, ta đã trình bày bài toán đối ngẫu dưới dạng tối đa hàm đối ngẫu Lagrange $\mathbf{g}(\alpha)$. Điều này tương đương với việc tối thiểu hóa $-\mathbf{g}(\alpha)$, ta phát biểu bài toán đối ngẫu SVM:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^N \alpha_i \\ \text{subject to} \quad & -\alpha_i \leq 0 \quad \forall i = 1, 2, \dots, N \\ & \sum_{i=1}^N y_i \alpha_i = 0. \end{aligned} \quad (40)$$

Đặt $\mathbf{V} = [y_1 \mathbf{x}_1, y_2 \mathbf{x}_2, \dots, y_N \mathbf{x}_N]^T$, $\mathbf{S} = [\alpha_1, \alpha_2, \dots, \alpha_N]$. Hàm mục tiêu của bài toán đối ngẫu có thể được viết lại thành:

$$-\mathbf{g}(\mathbf{S}) = -\frac{1}{2} \mathbf{S}^T \mathbf{V}^T \mathbf{V} \mathbf{S} + \mathbf{1}^T \mathbf{S}. \quad (41)$$

Bài toán đối ngẫu 40 giống như bài toán gốc 27 cũng là một bài toán quy hoạch toàn phương. Khi đã tìm được các biến số đối ngẫu α_i^* , có thể tính được nghiệm tối ưu của bài toán gốc \mathbf{w}^* dựa vào điều kiện 83.

$$\mathbf{w}^* = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n. \quad (42)$$

Để tính b^* , ta xem xét điểm dữ liệu \mathbf{x}_n nằm ở đúng đường biên lề, $\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b = y_n$. Do đó:

$$b^* = y_n - \langle \mathbf{w}^*, \mathbf{x}_n \rangle. \quad (43)$$

Tuy nhiên một phiên bản khác để tính b^* thường được sử dụng với ưu điểm là ổn định hơn trong tính toán là:

$$b^* = \frac{1}{|SV|} \sum_{n \in SV} (y_n - \langle \mathbf{w}^*, \mathbf{x}_n \rangle) = \frac{1}{|SV|} \sum_{n \in SV} \left(y_n - \sum_{m \in SV} \alpha_m^* y_m \langle \mathbf{x}_m, \mathbf{x}_n \rangle \right), \quad (44)$$

với $SV = \{i \mid \alpha_i^* \neq 0\}$ là tập các vector hỗ trợ.

Để đưa ra dự đoán nhãn cho một điểm dữ liệu mới, xét:

$$f(\mathbf{z}) = \langle \mathbf{w}^*, \mathbf{z} \rangle + b^* = \sum_{n=1}^N \alpha_n^* y_n \langle \mathbf{z}, \mathbf{x}_n \rangle + \frac{1}{|SV|} \sum_{n \in SV} \left(y_n - \sum_{m \in SV} \alpha_m^* y_m \langle \mathbf{x}_m, \mathbf{x}_n \rangle \right). \quad (45)$$

Algorithm 3 Giải bài toán đối ngẫu biên cứng SVM

Input: Tập dữ liệu $X = [x_1, x_2, \dots, x_N]^T \in \mathbb{R}^{N \times D}$ và tập nhãn tương ứng $Y = [y_1, y_2, \dots, y_N]^T \in \mathbb{R}^{N \times 1}$.

Output: w và b

```

1:  $V \leftarrow [y_j[X]_{jk}]_{jk}$ 
2:  $P \leftarrow V^T V$ 
3:  $q \leftarrow -\mathbf{1}_{N \times 1}$ 
4:  $G \leftarrow -I_N$ 
5:  $h \leftarrow \mathbf{0}_{N \times 1}$ 
6:  $A \leftarrow Y^T$ 
7:  $b \leftarrow 0$ 
8: solve QP with  $(P, q, G, h, A, b) \rightarrow S$ 
9:  $w \leftarrow \mathbf{0}_{D \times 1}$ 
10:  $SV \leftarrow 0$ 
11:  $\varepsilon \leftarrow 10^{-6}$ 
12: for all  $i$  such that  $S[i] > \varepsilon$  do
13:    $SV \leftarrow SV + 1$ 
14:    $w \leftarrow w + S[i]y_i x_i$ 
15: end for
16: for all  $i$  such that  $S[i] > \varepsilon$  do
17:    $b \leftarrow b + (y_i - \langle w, x_i \rangle)$ 
18: end for
19:  $b \leftarrow \frac{b}{SV}$ 
20: return  $w, b$ 

```

4.4 Những ưu điểm khi giải bài toán đối ngẫu

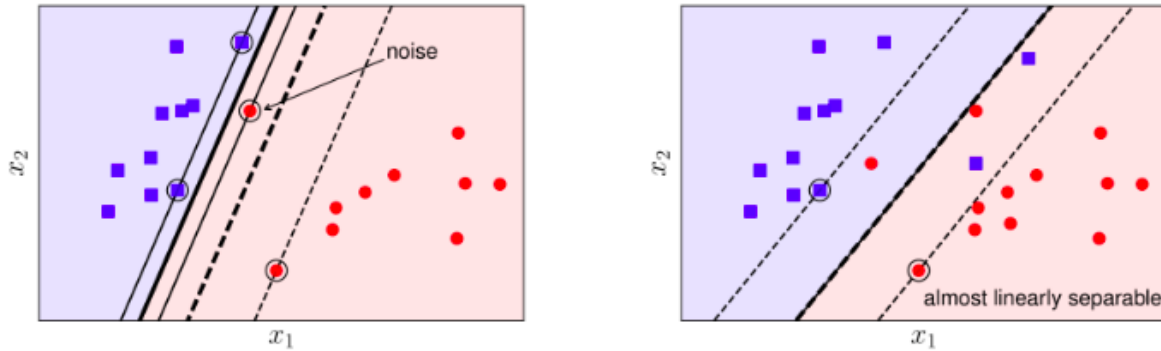
1. Vai trò của các vectơ hỗ trợ được thấy rõ qua nhận xét [4.1](#).
2. Bởi vì ta gán số nhân tử Lagrange α với mỗi ràng buộc nên nó có số chiều bằng với số điểm dữ liệu ta dùng để huấn luyện trong khi bài toán gốc w gán với số chiều dữ liệu, bài toán đối ngẫu có thể giải quyết các bài toán dữ liệu có số chiều lớn như dữ liệu là hình ảnh.
3. Việc giải bài toán đối ngẫu sẽ cho kết quả là một vector chứa rất ít phần tử khác không (tính thưa) nên cho phép một số phương pháp xấp xỉ nghiệm áp dụng để tìm kết quả nhanh.
4. Bài toán tối ưu [40](#) và bước truy vấn [45](#) đều chỉ phụ thuộc vào tích vô hướng giữa điểm dữ liệu cho phép phương pháp Kernel được áp dụng. Chi tiết về phương pháp này sẽ được giới thiệu ở phần [6](#).

5 PHÂN LOẠI BIÊN MỀM SVM

5.1 Động lực ra đời

Trong trường hợp dữ liệu là không tách biệt tuyến tính thì sử dụng phân loại lề cứng SVM để áp dụng cho bộ dữ liệu sẽ không thể tìm được w^*, b^* là nghiệm của bài toán tối ưu.

Ngoài ra, kể cả khi tập dữ liệu là tách biệt tuyến tính bài toán lề cứng cũng rất nhạy cảm với điểm nhiễu trong tập dữ liệu và dễ xảy ra quá khớp với dữ liệu dùng để huấn luyện.



Hình 6: Đường biên quyết định của mô hình phân loại biên cứng SVM (nét liền) và biên mềm SVM (nét gạch).

Phân loại biên mềm SVM là biến thể của lề cứng SVM cho phép một vài điểm dữ liệu "vượt lề". Việc cho phép một số lỗi trong việc gắn nhãn hoặc rơi vào trong khu vực lề sẽ giúp chúng ta tìm được siêu phẳng phân tách có lề lớn hơn. Nhưng ta cần hạn chế số điểm bị gắn sai nhãn lại vì việc chấp nhận quá nhiều lỗi sẽ làm lề quá lớn, thuật toán đã chấp nhận hầu hết các điểm bị sai. Vậy giữa việc tối ưu lề và việc hạn chế số điểm sai nhãn, việc nào nên được ưu tiên hơn?

5.2 Bài toán gốc biên mềm SVM

Ta đưa ra khái niệm biến ξ_n là *biến bù* (slack variables) ứng với mỗi cặp (x_n, y_n) . Sự hy sinh một số điểm bị nằm sai phía so với siêu mặt phẳng phân chia được thể hiện qua biến bù ξ_n .

1. $\xi_n = 0$ là những điểm đã được gắn nhãn đúng.
2. $0 < \xi_n < 1$ là những điểm nằm đúng nửa mặt phẳng nhưng nằm ở trong lề.
3. $\xi_n > 1$ là những điểm nằm sai phía so với đường biên.
4. $\langle w, x_n \rangle + b \geq 1 - \xi_n$ nếu $y_n = 1$ và $\langle w, x_n \rangle + b \geq -1 + \xi_n$ nếu $y_n = -1 \Leftrightarrow y_n(\langle w, x_n + b \rangle \geq 1 - \xi_n$.

Việc thêm biến bù vào hàm mục tiêu sẽ giúp chúng ta giảm thiểu những điểm dữ liệu bị gán sai nhãn nhưng vẫn đảm bảo lề của bài toán không nhỏ.

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n \\ \text{subject to} \quad & y_n(\langle w, x_n \rangle + b) \geq 1 - \xi_n \\ & \xi_n \geq 0 \quad \forall n = 1, \dots, N, \end{aligned} \quad (46)$$

trong đó, $C > 0$ là *hằng số phạt*:

- C thể hiện mức độ ưu tiên giữa tối đa lề của siêu phẳng phân tách về tối thiểu các biến bù.
- C càng lớn thì lỗi phân loại trong huấn luyện càng bị phạt nặng.

Vậy mô hình bài toán biên mềm SVM:

Algorithm 4 Giải biên mềm SVM

Input: Tập dữ liệu $X = [x_1, x_2, \dots, x_N]^T \in \mathbb{R}^{N \times D}$, tập nhãn tương ứng $Y = [y_1, y_2, \dots, y_N]^T \in \mathbb{R}^{N \times 1}$ và C .

Output: w và b .

- 1: $Z \leftarrow [X \ \mathbf{1}]_{N \times (D+1)}$
 - 2: $G \leftarrow -[y_j[Z]_{jk}]_{jk}$
 - 3: $G \leftarrow \begin{pmatrix} G & -I_N \\ \mathbf{0}_{N \times (D+1)} & -I_N \end{pmatrix}_{(2N) \times (D+N+1)}$
 - 4: $h \leftarrow \begin{pmatrix} -\mathbf{1}_{N \times 1} \\ \mathbf{0}_{N \times 1} \end{pmatrix}_{(2N) \times (1)}$
 - 5: $q \leftarrow [\mathbf{0}_{(D+1) \times 1}, C \cdot \mathbf{1}_{N \times 1}]^T$
 - 6: $P \leftarrow \begin{pmatrix} I_D & \mathbf{0}_{D \times (N+1)} \\ \mathbf{0}_{(N+1) \times D} & 0 \end{pmatrix}_{(D+N+1) \times (D+N+1)}$
 - 7: **solve QP with** $(P, q, G, h) \rightarrow S$
 - 8: $w \leftarrow S[:, D]$
 - 9: $b \leftarrow S[D]$
 - 10: **return** w, b
-

5.3 Bài toán đối ngẫu của biên mềm SVM

Kiểm tra điều kiện Slater: Với mọi $n = 1, \dots, N$, và mọi (w, b) luôn tìm được các số dương ξ_n đủ lớn sao cho:

$$\xi_n > 1 - y_n(\langle w, x_n \rangle + b).$$

Bài toán này thỏa mãn điều kiện Slater, vậy xảy ra đối ngẫu chặt.

Hàm Lagrangian ứng với bài toán lề mềm:

$$\begin{aligned} \mathcal{L}(w, b, \xi, \alpha, \gamma) = & \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n \\ & - \sum_{n=1}^N \alpha_n (y_n(\langle w, x_n \rangle + b) - 1 + \xi_n) - \sum_{n=1}^N \gamma_n \xi_n, \end{aligned} \quad (47)$$

trong đó $\alpha_n \geq 0$ và $\gamma_n \geq 0$ là các nhân tử Lagrange tương ứng với hai ràng buộc của bài toán tối ưu.

Bài toán đối ngẫu của biên mềm SVM:

$$\alpha^*, \gamma^* = \arg \max_{\alpha, \gamma} \left(\inf_{w, b, \xi} \mathcal{L}(w, b, \xi, \alpha, \gamma) \right). \quad (48)$$

Vì bài toán của chúng ta là bài toán lồi và đối ngẫu chặt xảy ra tương tự như với bài toán đối ngẫu của biên cứng, hệ điều kiện KKT trở thành điều kiện cần và đủ. Hệ điều kiện KKT của biên mềm SVM:

$$1 - \xi_n - y_n(\langle w, x_n \rangle + b) \leq 0, \quad (49)$$

$$-\xi_n \leq 0, \quad (50)$$

$$\alpha_n \geq 0 \quad \gamma_n \geq 0, \quad (51)$$

$$\alpha_n(1 - \xi_n - y_n(\langle w, x_n \rangle + b)) = 0, \quad (52)$$

$$\gamma_n \xi_n = 0, \quad (53)$$

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{n=1}^N \alpha_n y_n x_n = 0, \quad (54)$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^N \alpha_n y_n = 0, \quad (55)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \alpha_n - \gamma_n = 0. \quad (56)$$

Từ phương trình (54), (55) và (56) ta có bài toán đối ngẫu trở thành :

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^N \alpha_i \quad (57)$$

$$\text{subject to} \quad \sum_{i=1}^N y_i \alpha_i = 0 \quad (58)$$

$$0 \leq \alpha_i \leq C \quad \forall i = 1, 2, \dots, N. \quad (59)$$

Viết lại dưới dạng bài toán quy hoạch toàn phương ta có :

$$\min_{\alpha} \frac{1}{2} S^T V^T V S - \mathbf{1}^T S \quad (60)$$

$$\text{subject to} \quad -\alpha_i \leq 0 \quad \forall i = 1, 2, \dots, N \quad (61)$$

$$\alpha_i \leq C \quad \forall i = 1, 2, \dots, N \quad (62)$$

$$\sum_{i=1}^N y_i \alpha_i = 0. \quad (63)$$

Từ đây ta sẽ tính được α^* ta quay lại tìm b^* và w^* tương tự như bài toán đối ngẫu của biên cứng SVM. Bài toán trên giống với bài toán đối ngẫu của bài toán SVM biên cứng chỉ khác là ta có chặn trên cho mỗi λ_i bởi C và (59) được gọi là *điều kiện hộp*. Điều kiện hộp đảm bảo hàm Lagrangian được chặn dưới đúng bởi một hàm phụ thuộc vào α và γ .

Nhận xét 5.1. Phương trình (52) của điều kiện KKT cho ta kết quả thú vị

1. Nếu $\alpha_n = 0$ thì $y_i(\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b) > 1$. Đây là những điểm nằm đúng phía so với đường thẳng phân cách.
2. Nếu $0 < \alpha_n < C$ thì $y_i(\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b) = 1$. Đây là những điểm được gán đúng nhãn và nằm trên lề.
3. Nếu $\alpha_n = C$ thì $y_i(\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b) \leq 1$. Những điểm này nằm trong lề kể cả chúng có nằm sai và đúng phía so với đường thẳng phân cách.
4. Khác với bài toán biên cứng, tập các vectơ hỗ trợ bây giờ bao gồm các điểm nằm trên lề và cả các điểm nằm sai phía ở bên trong lề.

Cách cài đặt của bài toán đối ngẫu cho biên mềm :

Algorithm 5 Giải bài toán đối ngẫu biên mềm SVM

Input: Tập dữ liệu $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$ tập nhãn tương ứng $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^T \in \mathbb{R}^{N \times 1}$ và C .

Output: \mathbf{w} và b .

- 1: $\mathbf{V} \leftarrow [\mathbf{y}_j[\mathbf{X}]_{jk}]_{jk}$
 - 2: $\mathbf{P} \leftarrow \mathbf{V}^T \mathbf{V}$
 - 3: $\mathbf{q} \leftarrow -\mathbf{1}_{N \times 1}$
 - 4: $\mathbf{G} \leftarrow \begin{pmatrix} -\mathbf{I}_N \\ \mathbf{I}_N \end{pmatrix}_{(2N) \times (N)}$
 - 5: $\mathbf{h} \leftarrow \begin{pmatrix} \mathbf{0}_{N \times 1} \\ C \cdot \mathbf{1}_{N \times 1} \end{pmatrix}_{(2N) \times (1)}$
 - 6: $\mathbf{A} \leftarrow \mathbf{Y}^T$
 - 7: $b \leftarrow 0$
 - 8: **solve QP with** $(\mathbf{P}, \mathbf{q}, \mathbf{G}, \mathbf{h}, \mathbf{A}, b) \rightarrow \mathbf{S}$
 - 9: $\mathbf{w} \leftarrow \mathbf{0}_{D \times 1}$
 - 10: $SV \leftarrow 0$
 - 11: $\varepsilon \leftarrow 10^{-6}$
 - 12: **for all** i such that $\mathbf{S}[i] > \varepsilon$ **do**
 - 13: $SV \leftarrow SV + 1$
 - 14: $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{S}[i] \mathbf{y}_i \mathbf{x}_i$
 - 15: **end for**
 - 16: **for all** i such that $\mathbf{S}[i] > \varepsilon$ **do**
 - 17: $b \leftarrow b + (\mathbf{y}_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)$
 - 18: **end for**
 - 19: $b \leftarrow \frac{b}{SV}$
 - 20: **return** \mathbf{w}, b
-

6 PHƯƠNG PHÁP KERNEL TRONG SVM

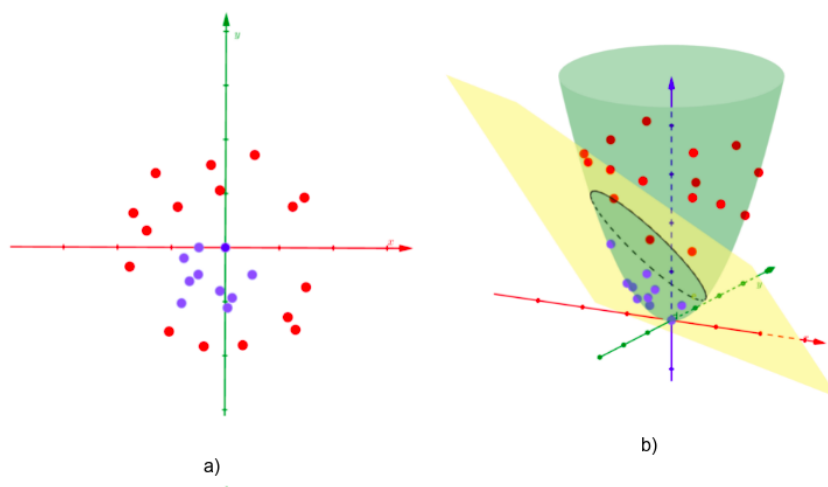
6.1 Động lực

Khả năng phân tách của siêu phẳng bị giới hạn trong trường hợp dữ liệu có tính tách biệt tuyến tính. Để khắc phục hạn chế với các bộ dữ liệu không tách biệt tuyến tính, ta có thể

đưa dữ liệu ban đầu lên một không gian khác (có thể cao chiều hơn) và tìm siêu phẳng phân tách dữ liệu trong không gian mới.

Ta có thể phát biểu ý tưởng sử dụng mô hình phân loại tuyến tính cho dữ liệu phi tuyến như sau:

1. Với không gian dữ liệu \mathcal{X} , ta chọn một ánh xạ $\psi : \mathcal{X} \rightarrow \mathcal{F}$, với không gian đặc trưng \mathcal{F} là một không gian Hilbert.
2. Từ tập dữ liệu huấn luyện ban đầu, $S = (x_1, y_1), \dots, (x_N, y_N)$, tính tập ảnh, $\hat{S} = (\psi(x_1), y_1), \dots, (\psi(x_N), y_N)$.
3. Huấn luyện mô hình h trên \hat{S} .
4. Dự đoán nhãn của điểm dữ liệu mới, z , bằng $h(\psi(z))$.



Hình 7: a, Dữ liệu ban đầu. b, Dữ liệu được biến đổi qua $\psi((x, y)) = (x, y, x^2 + y^2)$.

Tuy nhiên, nếu tập đích ψ là một không gian cao chiều (có thể là vô hạn chiều), thực hiện các tính toán trong đó trở nên cực kỳ đắt đỏ. Với trường hợp không gian vô hạn chiều thì tìm biểu diễn của w trong không gian đặc trưng là không thể. Phương pháp kernel được giới thiệu trong mục tới là giải pháp cho vấn đề về tính toán dữ liệu cao chiều.

6.2 Phương pháp kernel

Định nghĩa 6.1 (Hàm nhân - Kernel function). *Hàm nhân* $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ tương ứng với ánh xạ $\psi : \mathcal{X} \rightarrow \mathcal{F}$, \mathcal{F} là một không gian Hilbert, là:

$$K(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle. \quad (64)$$

Trên thực tế, có rất nhiều thuật toán có đặc điểm là các dữ liệu đầu vào chỉ xuất hiện dưới dạng các hàm nhân của các cặp điểm, mà đối ngẫu SVM là một ví dụ sẽ được đề cập trong mục tới. Sự thuận lợi mà những thuật toán đó đem lại nằm ở việc ta không cần phải định nghĩa rõ ràng ψ và đi tính toán $\psi(x)$. Việc thay thế các tính vô hướng bằng hàm nhân được gọi là phương pháp kernel (kernel trick).

Ví dụ 6.2 (Hàm nhân đa thức). *Hàm nhân đa thức bậc k* được định nghĩa là:

$$K(x, x') = (1 + \langle x, x' \rangle)^k. \quad (65)$$

Ta sẽ chứng minh rằng đây là một hàm nhân đúng, bằng cách chỉ ra một ánh xạ ψ sao cho $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$. Kí hiệu $x_0 = x'_0 = 1$. Ta có:

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \dots (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \quad (66)$$

$$= \left(\sum_{j=0}^n x_j x'_j \right) \dots \left(\sum_{j=0}^n x_j x'_j \right) \quad (67)$$

$$= \sum_{J \in \{0,1,\dots,n\}^k} \prod_{i=1}^k x_{J_i} x'_{J_i} \quad (68)$$

$$= \sum_{J \in \{0,1,\dots,n\}^k} \prod_{i=1}^k x_{J_i} \prod_{i=1}^k x'_{J_i} \quad (69)$$

Ta có thể định nghĩa $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^{(n+1)^k}$, với mỗi $J \in \{0,1,\dots,n\}^k$ có một phần tử của $\psi(\mathbf{x})$ bằng với $\prod_{i=1}^k x_{J_i}$. Do đó ta có:

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle. \quad (70)$$

Do ψ bao gồm những đơn thức bậc không vượt quá k , tìm một siêu phẳng phân tách với trong tập đích của ψ tương ứng với một mô hình phân loại có dạng đa thức bậc k trong không gian dữ liệu ban đầu. Một nhận xét nữa là độ phức tạp của việc tính toán hàm K chỉ là $O(n)$ trong khi số chiều của không gian đặc trưng là n^k .

Các hàm nhân có thể được thiết kế từ một ánh xạ đặc trưng ψ , nhưng thông thường thì chúng đều được thiết kế một cách trực tiếp. Tính đúng của một hàm nhân có thể được kiểm tra thông qua bổ đề sau:

Bổ đề 6.3. Hàm đối xứng $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ là hàm nhân đúng khi nó bán xác định dương, nghĩa là với một tập hữu hạn $\{x_1, \dots, x_n\} \in \mathcal{X}$, ma trận Gram, $G = (k(x_i, x_j))_{ij}$, là bán xác định dương.

Một số hàm nhân thông dụng:

- Hàm nhân tuyến tính: $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$.
- Hàm nhân đa thức: $K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^\gamma$, $\gamma > 0$
- Hàm nhân RBF: $K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2)$, $\gamma > 0$
- Hàm nhân Sigmoid: $K(\mathbf{x}, \mathbf{z}) = \tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$

6.3 Kernel SVM

Như đã được giới thiệu ở mục trước và phần 4.4, các điểm dữ liệu huấn luyện ở đối ngẫu SVM chỉ xuất hiện dưới dạng tích vô hướng với các điểm dữ liệu khác, điều này cho phép SVM (sử dụng một siêu phẳng phân tách tuyến tính) có thể phân loại được những dữ liệu phi tuyến tính mà chi phí tính toán không bị tăng lên quá nhiều thông qua phương pháp kernel.

Áp dụng phương pháp kernel, hàm mục tiêu và bước truy vấn của đối ngẫu SVM có thể viết lại:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N \alpha_i, \quad (71)$$

$$f(z) = \sum_{n=1}^N \alpha_n^* y_n K(z, x_n) + \frac{1}{|SV|} \sum_{n \in SV} \left(y_n - \sum_{m \in SV} \alpha_m^* y_m K(x_m, x_n) \right). \quad (72)$$

Nhận xét 6.4. Phương pháp kernel hoạt động rất hiệu quả khi dùng trong SVM:

- Từ 72 có thể thấy rằng để đưa ra dự đoán cho một điểm dữ liệu mới, ta phải tính hàm nhân của điểm mới với tất cả các điểm dữ liệu trong tập huấn luyện. Tuy nhiên, nhờ vào tính thưa của α^* trong SVM nên chỉ cần tính hàm nhân của điểm mới với số nhỏ các vector hỗ trợ (điểm có α^* tương ứng khác 0 - từ nhận xét 4.1) để đưa ra nhãn dự đoán.
- Hơn nữa, từ điều kiện bán xác định dương của hàm nhân, bài toán tối ưu vẫn là bài toán tối ưu toàn phương.

7 BÀI TOÁN TỐI ƯU MỀM KHÔNG RÀNG BUỘC

7.1 Xây dựng bài toán bài toán tối ưu mềm không ràng buộc

Với SVM, ta xem xét lớp hàm dự đoán $f(x_n, w, b) : \mathbb{R}^D \rightarrow \mathbb{R}$ có dạng :

$$f(x) = \langle w, x \rangle + b$$

Để đánh giá khả năng phân loại của hàm dự đoán ta sử dụng hàm mất mát $\ell(y_n, \hat{y}_n)$ với y_n là nhãn đúng và \hat{y}_n là nhãn dự đoán mà hàm f đưa ra ($\hat{y}_n = f(x_n)$). Hàm mất mát sẽ nhận giá trị 0 nếu $f(x_n) = y_n$ và lớn hơn 0 nếu dự đoán đưa ra là sai. Hàm mất mát được dùng trong SVM là hinge loss, có dạng :

$$\ell(x_n) = \max(0, 1 - y_n f(x_n)) \text{ với } f(x_n) = \langle w, x_n \rangle + b \quad (73)$$

Việc sử dụng hàm hinge loss cho ta kết quả như sau :

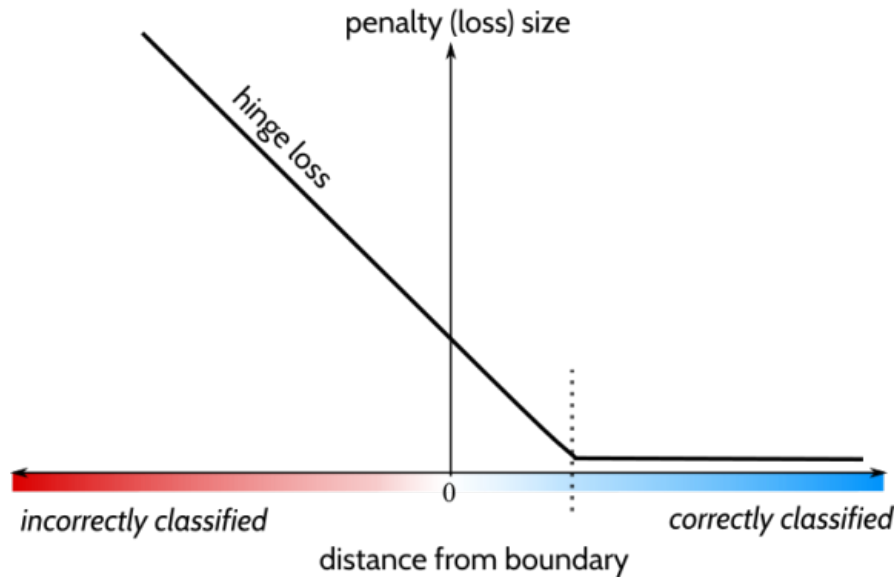
1. Nếu nhãn dự đoán trùng với nhãn thật thì $y_n f(x_n) \geq 1$ và hàm hinge loss có giá trị bằng 0.
2. Nếu $0 < y_n f(x_n) < 1$ thì điểm dữ liệu này cũng được gán nhãn đúng nhưng nằm bên trong khoảng lề.
3. Nếu $y_n f(x_n) < 1$ thì điểm dữ liệu bị gán nhãn sai vì vậy hàm hinge loss sẽ trả lại 1 giá trị lớn tùy vào độ lệch của nhãn thật với nhãn dự đoán.

Ta cần tìm w, b làm cho tổng giá trị của hàm hinge loss qua toàn bộ điểm huấn luyện nhỏ nhất :

$$\min_{w, b} \sum_{n=1}^N \max(0, 1 - y_n f(x_n)) \quad (74)$$

Khi dữ liệu ta dùng huấn luyện là phân tách tuyến tính nên giá trị tối ưu có thể bằng 0. Nếu gọi (w^*, b^*) là nghiệm tối ưu thì nếu thay (cw^*, cb^*) ta có $y_n(1 - \langle cw^*, x_n \rangle + cb^*) < 0 \forall n = 1, \dots, N$, vậy (cw^*, cb^*) cũng là điểm tối ưu của bài toán. Việc giải bài toán tối ưu trên sẽ cho ta vô số điểm tối ưu.

Vậy chúng ta sẽ thêm vào hàm mục tiêu một đại lượng thể hiện sự ưu tiên cho một số w vì vậy làm cho điểm tối ưu trở nên duy nhất. Đại lượng này được gọi là *chính quy*. Ngoài động lực ta đã nêu bên trên việc thêm đại lượng chính quy còn cho ta một số tính chất:



Hình 8: Minh họa cho hàm hinge loss.

1. Việc thêm đại lượng chính quy sẽ đồng nhất với việc tối ưu lồi theo góc nhìn hình học.
2. Ngoài ra việc thêm đại lượng chính quy giúp cho bài toán được cải thiện tính tổng quát trong tập truy vấn bởi vì đại lượng chính quy không cho phép các điểm dữ liệu huấn luyện có ảnh hưởng lớn đến giá trị hàm mất mát.

Đại lượng thêm thường được sử dụng là $C\|w\|^2$. Để việc tính đạo hàm dễ dàng ta nhân thêm với đại lượng này $\frac{1}{2}$, cuối cùng ta có bài toán tối ưu :

$$\min_{w,b} \sum_{n=1}^N \max(0, 1 - y_n(f(x_n))) + \frac{C}{2} \|w\|^2 \quad (75)$$

Bài toán SVM biên mềm gốc và bài toán không ràng buộc là hai bài toán tương đương. Việc tối thiểu làm hinge loss chính là tối thiểu tổng các biến bù ứng với mỗi điểm dữ liệu (x_n, y_n) . Thật vậy:

$$\xi_i \geq 0 \text{ và } \xi_i \geq 1 - (\langle w, x_i \rangle + b) \quad (76)$$

$$\Leftrightarrow \xi_i \geq \max(0, 1 - (\langle w, x_i \rangle + b)) \quad (77)$$

Vậy nếu giả sử (w^*, b^*, ξ_i^*) là nghiệm tối ưu của phân loại biên mềm SVM thì từ (77) ta có : $\xi_i^* = \max(0, 1 - \langle w^*, x_i \rangle + b^*)$.

7.2 Áp dụng Stochastic Gradient Descent giải quyết bài toán

$$\frac{\partial}{\partial w} \ell = \begin{cases} -y_i x_i & \text{khi } y_i(\langle w, x_i \rangle + b) < 1 \\ 0 & \text{khi } y_i(\langle w, x_i \rangle + b) \geq 1 \end{cases} \quad (78)$$

$$\frac{\partial}{\partial b} \ell = \begin{cases} -y_i & \text{khi } y_i(\langle w, x_i \rangle + b) < 1 \\ 0 & \text{khi } y_i(\langle w, x_i \rangle + b) \geq 1 \end{cases} \quad (79)$$

Từ đây ta đạo hàm hàm mục tiêu $J(\mathbf{w}, b) = \sum_{n=1}^N \max(0, 1 - y_n f(\mathbf{x}_n)) + \frac{C}{2} \|\mathbf{w}\|^2$ theo 2 biến \mathbf{w}, b :

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b) = \begin{cases} -y_i \mathbf{x}_i - \frac{C}{2} \mathbf{w} & \text{khi } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) < 1 \\ -\frac{C}{2} \mathbf{w} & \text{khi } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \end{cases} \quad (80)$$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b) = \begin{cases} -y_i & \text{khi } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) < 1 \\ 0 & \text{khi } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \end{cases} \quad (81)$$

Áp dụng thuật toán Stochastic Gradient Descent để tìm \mathbf{w}^*, b^* :

Algorithm 6 SGD cho bài toán Soft Margin SVM không ràng buộc

Input: Tập dữ liệu $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$, tập nhãn tương ứng $\mathbf{Y} = [y_1, y_2, \dots, y_N]^T \in \mathbb{R}^{N \times 1}$, tốc độ học α , số vòng lặp T

Output: \mathbf{w}^* và b^*

```

1: Khởi tạo giá trị  $\mathbf{w}_0$  và  $b_0$ .
2: for  $i = 0$  to  $T$  do
3:   Chọn 1 điểm dữ liệu bất kỳ  $(\mathbf{x}_i, y_i)$ 
4:   if  $(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) < 1$  then
5:      $\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha(-y_i \mathbf{x}_i - \frac{C}{2} \mathbf{w}_i)$ 
6:      $b_{i+1} = b_i - \alpha(-y_i)$ 
7:   else
8:      $\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha(-\frac{C}{2} \mathbf{w}_i)$ 
9:      $b_{i+1} = b_i$ 
10:  end if
11: end for
12: return  $\mathbf{w}_{i+1}, b_{i+1}$ 

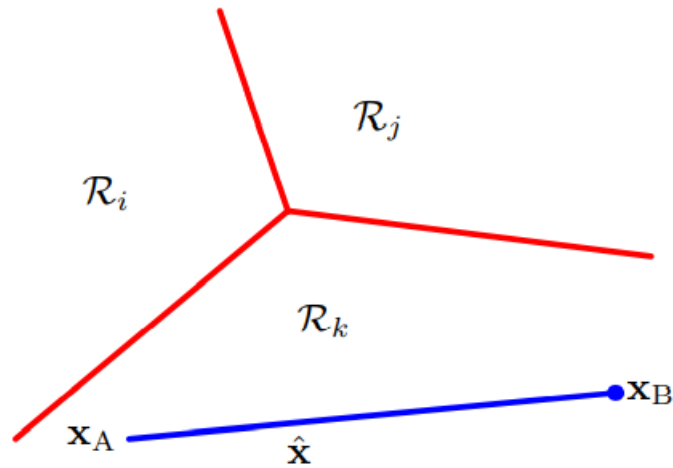
```

8 MULTI-CLASS SVM

Bài toán SVM mà ta xây dựng ở trên chỉ phục vụ phân loại nhị phân tức là phân chia dữ liệu thành 2 lớp. Bởi vì mỗi mô hình SVM chỉ phân biệt giữa 2 lớp nên một cách tự nhiên để mở rộng mô hình để phân chia nhiều lớp là sử dụng nhiều thuật toán SVM trong một mô hình đó chính là ý tưởng của kĩ thuật *one-vs-one* và *one-vs-rest*.

Trong *one-vs-one*, chúng ta sử dụng mỗi mô hình SVM để phân loại 2 lớp, bỏ qua các điểm của các lớp còn lại. Điều này có nghĩa sự phân tách chỉ quan tâm đến dữ liệu của 2 lớp cần phân chia. Vì vậy để phân chia hết m lớp chúng ta cần $\frac{m(m-1)}{2}$ mô hình SVM. Khi có một dữ liệu mới được đưa vào thì nó sẽ được đưa qua toàn bộ các mô hình SVM và cuối cùng nhãn của dữ liệu sẽ được quyết định dựa trên lớp mà nó được phân vào nhiều nhất.

Phương pháp thường được sử dụng nhiều hơn là *one-vs-rest*. Nếu có m nhóm chúng ta sẽ sử dụng m mô hình phân loại, mỗi một mô hình SVM sẽ tìm một siêu mặt phẳng để tách dữ liệu thuộc một lớp với các lớp khác. Chẳng hạn mô hình phân loại 1 sẽ chia dữ liệu thành hai phần thuộc lớp 1 và không thuộc lớp 1.



Hình 9: Miêu tả các vùng quyết định là kết nối với nhau và có tính lỗi

Nhưng khi dùng phương pháp trên dẫn tới một vùng chứa dữ liệu không được gán nhãn. Để tránh được vấn đề trên ta sẽ gộp cả m phương trình đường thẳng phân cách của m mô hình SVM bằng một phương trình duy nhất có dạng:

$$y_i(x) = \langle w_i, x \rangle + b_i, \quad \forall i = 1, 2, \dots, m. \quad (82)$$

Một điểm sẽ thuộc lớp k nếu $y_k(x) - y_j(x) > 0$ với $k \neq j$. Vậy siêu mặt phẳng phân tách lớp k và lớp j có dạng:

$$\langle w_k - w_j, x \rangle + (b_k - b_j) = 0. \quad (83)$$

Sau khi xây dựng lớp hàm như vậy ta thấy mọi mặt phẳng phân cách giờ đây đều liên kết và sẽ chia mặt phẳng thành m vùng có tính chất lỗi.

Chứng minh: Giả sử hai điểm x_A và x_B thuộc vùng \mathcal{R}_k . Bất kì điểm nào thuộc đường thẳng nối x_A và x_B sẽ được biểu diễn dưới dạng:

$$\hat{x} = \theta x_A + (1 - \theta)x_B, \text{ với } 0 \leq \theta \leq 1.$$

Nhờ sự tuyến tính của phương trình (83) ta có :

$$y_k(\hat{x}) = \theta y_k(x_A) + (1 - \theta)y_k(x_B), \text{ với } 0 \leq \theta \leq 1.$$

Vì ta giả sử x_A và x_B thuộc \mathcal{R}_k nên $y_k(x_A) > y_j(x_A)$ và $y_k(x_B) > y_j(x_B)$ vậy $y_k(\hat{x}) > y_j(\hat{x})$ tương tự thì $y_k(\hat{x}) > y_i(\hat{x})$, vậy \hat{x} thuộc lớp \mathcal{R}_k . ■

Vậy từ đây vấn đề nêu ở trên của phương pháp one-vs-rest đã được khắc phục.

9 THỰC HÀNH

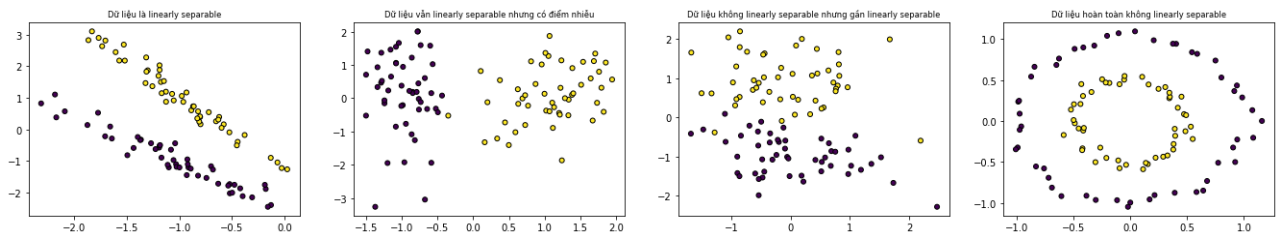
9.1 Chạy trên dữ liệu tự sinh

Dữ liệu được sinh từ sklearn.datasets gồm 4 bộ dữ liệu như sau:

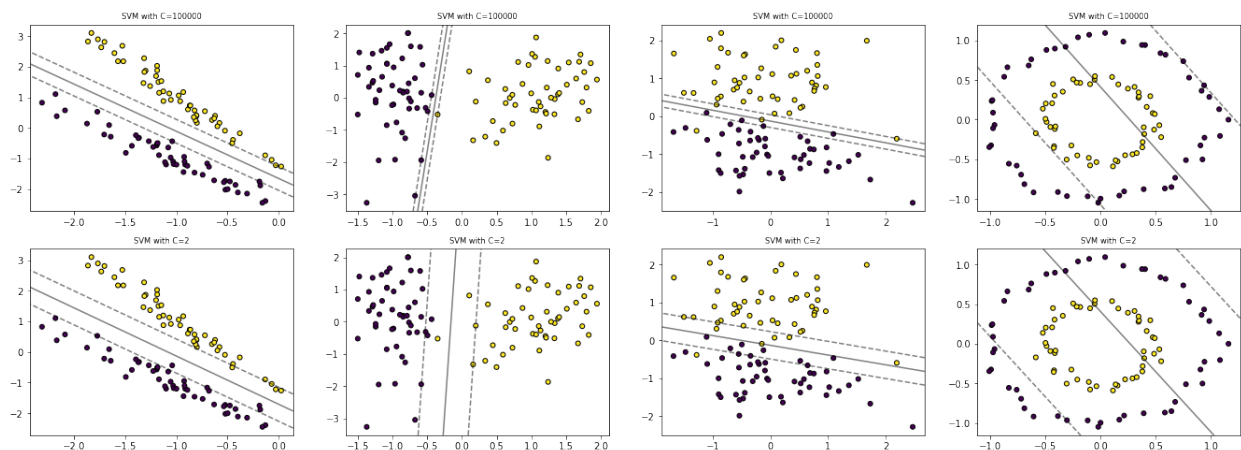
- Dữ liệu là tách biệt tuyến tính: Có đường thẳng phân tách dữ liệu thành 2 tập.

- Dữ liệu vẫn tách biệt tuyến tính nhưng có điểm nhiễu: Vẫn tồn tại đường thẳng phân tách dữ liệu thành 2 tập nhưng đường thẳng đó sẽ bị lệch về một phía.
- Dữ liệu gần tách biệt tuyến tính: Không tồn tại đường thẳng phân tách hoàn toàn nhưng vẫn phân tách được cho phần lớn điểm dữ liệu.
- Dữ liệu hoàn toàn không tách biệt tuyến tính: Hoàn toàn không tồn tại đường thẳng nào cho kết quả tốt.

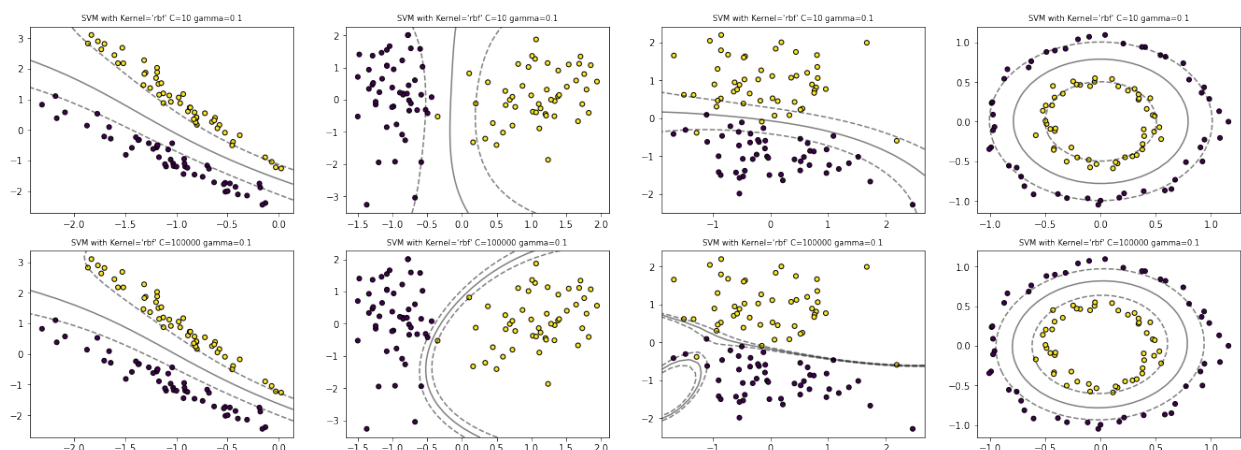
Mỗi tập dữ liệu gồm 100 điểm dữ liệu, có hình dạng như sau:



Chạy thử mô hình SVM thông thường lần lượt với $C=100000$ và $C=2$ thì ta được đường thẳng phân cách và margin như sau:



Còn khi chạy mô hình SVM với kernel rbf lần lượt với $C=10$ và $C=100000$, hằng số $\gamma=0.1$ thì ta được đường thẳng phân cách và lề như sau:



Ta rút ra được một số nhận xét như sau:

- SVM biên mềm hoạt động tốt khi dữ liệu có dạng tách biệt tuyến tính.

- Khi dữ liệu không tách biệt tuyến tính thì SVM với kernel='linear' không thể cho ra đáp án.
- Với SVM kernel mô hình chạy tốt với nhiều dạng dữ liệu, nhưng cần phải điều chỉnh các tham số một cách hợp lý.

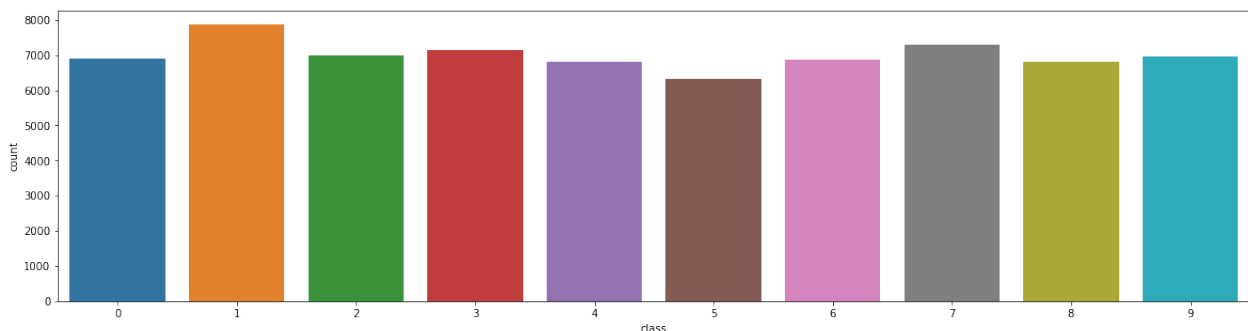
9.2 Chạy trên dữ liệu MNIST

Bài toán gốc: nhận diện số viết tay từ 0 đến 9

Dữ liệu MNIST gồm 70000 ảnh 28x28 đã được trải ra thành vector, với nhãn từ 0 đến 9 và có sự phân bố đều giữa các nhãn

	pixel1	pixel2	pixel3	pixel4	...	pixel782	pixel783	pixel784	class
0	0	0	0	0	...	0	0	0	5
1	0	0	0	0	...	0	0	0	0
2	0	0	0	0	...	0	0	0	4
3	0	0	0	0	...	0	0	0	1
4	0	0	0	0	...	0	0	0	9
...
69995	0	0	0	0	...	0	0	0	2
69996	0	0	0	0	...	0	0	0	3
69997	0	0	0	0	...	0	0	0	4
69998	0	0	0	0	...	0	0	0	5
69999	0	0	0	0	...	0	0	0	6

[70000 rows x 785 columns]



Để áp dụng SVM trong phân loại nhị phân ta sẽ lấy từ tập dữ liệu trên các điểm dữ liệu có nhãn là 8 và 9. Bài toán trở thành phân biệt số 8 và số 9.

Đầu tiên ta sẽ lấy từ tập dữ liệu các điểm dữ liệu có nhãn là 8 và 9. Vì dữ liệu đã được xử lý nhưng chưa được chuẩn hóa, nên ta sẽ chuẩn hóa dữ liệu để các cột có dữ liệu thì đều có mean=0 và std=1.

```
for col in X.columns.values:
    value = np.std(X[col])
    if (value > 0):
        X[col] = (X[col] - np.mean(X[col]))/value
```

Do không có tập test sẵn nên ta chia dữ liệu thành các tập train và test với tỉ lệ 7:3.

Chạy mô hình SVM trên tập train và đánh giá độ chính xác trên tập test.

Khi chạy SVM thông thường với các C lần lượt là 10^5 , 10, 0,01 ta được kết quả như sau:

```

C = 100000
      precision    recall  f1-score   support

      8       0.97       0.97       0.97        2050
      9       0.97       0.97       0.97        2085

   accuracy
 macro avg       0.97       0.97       0.97        4135
weighted avg       0.97       0.97       0.97        4135

[[1979   71]
 [   57 2028]]
C = 10
      precision    recall  f1-score   support

      8       0.97       0.97       0.97        2050
      9       0.97       0.97       0.97        2085

   accuracy
 macro avg       0.97       0.97       0.97        4135
weighted avg       0.97       0.97       0.97        4135

[[1979   71]
 [   57 2028]]
C = 0.01
      precision    recall  f1-score   support

      8       0.98       0.98       0.98        2050
      9       0.98       0.98       0.98        2085

   accuracy
 macro avg       0.98       0.98       0.98        4135
weighted avg       0.98       0.98       0.98        4135

[[2018   32]
 [   34 2051]]

```

Chạy với các kernel thông dụng với gamma là 10 và 10^{-3} ta được bảng kết quả như sau:

kernel = sigmoid gamma = 10					kernel = sigmoid gamma = 0.001				
	precision	recall	f1-score	support		precision	recall	f1-score	support
8	0.81	0.82	0.81	2050	8	0.98	0.98	0.98	2050
9	0.82	0.81	0.82	2085	9	0.98	0.98	0.98	2085
accuracy			0.82	4135	accuracy			0.98	4135
macro avg	0.82	0.82	0.82	4135	macro avg	0.98	0.98	0.98	4135
weighted avg	0.82	0.82	0.82	4135	weighted avg	0.98	0.98	0.98	4135
[[1675 375] [389 1696]]					[[2009 41] [51 2034]]				
kernel = poly gamma = 10					kernel = poly gamma = 0.001				
	precision	recall	f1-score	support		precision	recall	f1-score	support
8	0.99	1.00	0.99	2050	8	0.99	0.97	0.98	2050
9	1.00	0.99	0.99	2085	9	0.97	0.99	0.98	2085
accuracy			0.99	4135	accuracy			0.98	4135
macro avg	0.99	0.99	0.99	4135	macro avg	0.98	0.98	0.98	4135
weighted avg	0.99	0.99	0.99	4135	weighted avg	0.98	0.98	0.98	4135
[[2040 10] [17 2068]]					[[1988 62] [29 2056]]				
kernel = rbf gamma = 0.001					kernel = rbf gamma = 0.001				
	precision	recall	f1-score	support		precision	recall	f1-score	support
8	0.00	0.00	0.00	2050	8	0.98	0.99	0.99	2050
9	0.50	1.00	0.67	2085	9	0.99	0.98	0.99	2085
accuracy			0.50	4135	accuracy			0.99	4135
macro avg	0.25	0.50	0.34	4135	macro avg	0.99	0.99	0.99	4135
weighted avg	0.25	0.50	0.34	4135	weighted avg	0.99	0.99	0.99	4135
[[0 2050] [0 2085]]					[[2035 15] [34 2051]]				

Như vậy kernel = poly, gamma = 10 cho kết quả tốt nhất, kể đến là kernel = rbf, gamma = 0.001 và cả hai đều có accuracy rất cao là 0.99

9.3 Cài đặt SVM biên cứng và SVM biên mềm bằng qpsolvers và cvxopt

Code cho phần biên cứng:

```
# data generator
from __future__ import print_function
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
np.random.seed(22)

means = [[2, 2], [4, 2]]
cov = [[.3, .2], [.2, .3]]
N = 30
n = 60
X0 = np.random.multivariate_normal(means[0], cov, N)
X1 = np.random.multivariate_normal(means[1], cov, N)
X = np.concatenate((X0, X1), axis = 0)
y = np.concatenate((np.ones((N, 1)), -1*np.ones((N, 1))), axis = 0)
```

```
# Primal problem
from cvxopt import matrix, solvers
from qpsolvers import solve_qp
G0 = np.vstack((-X0.T[0], -X0.T[1], -np.ones(N))).T
G1 = np.vstack((X1.T[0], X1.T[1], np.ones(N))).T
G = np.concatenate((G0, G1), axis = 0)
h = -np.ones((n,))
P = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 0]], dtype=np.float64)
q = np.zeros((3,))
S = solve_qp(P, q, G, h, solver='cvxopt')
w_primal = S[:2]
w_primal = w_primal.reshape(1, 2)
b_primal = S[2]
```

```
# Dual problem
from cvxopt import matrix, solvers
# build P
V = np.concatenate((X0, -X1), axis = 0)
P = matrix(V.dot(V.T))
q = matrix(-np.ones((n, 1)))

# build A, b, G, h
G = matrix(-np.eye(n))
h = matrix(np.zeros((n, 1)))
A = matrix(y.T)
b = matrix(np.zeros((1, 1)))
solvers.options['show_progress'] = False
S = solvers.qp(P, q, G, h, A, b)
l = np.array(S['x'])
w_dual = matrix(np.zeros((1, 2)))
SV = 0
```

```

e = 1e-6
b_dual = 0
for i in range(0,n):
    if l[i] > e:
        SV = SV + 1
        w_dual = w_dual + l[i]*V[i]
for i in range(0,n):
    if l[i] > e:
        b_dual = b_dual + (y[i]-w_dual.dot(X[i]))
b_dual = b_dual / SV

```

Code cho phần biên mềm

```

# Data generator
from __future__ import division, print_function, unicode_literals
# list of points
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from matplotlib.backends.backend_pdf import PdfPages
np.random.seed(22)

means = [[2, 2], [4, 2]]
cov = [[.7, 0], [0, .7]]
N = 30
n = 60
X0 = np.random.multivariate_normal(means[0], cov, N)
X1 = np.random.multivariate_normal(means[1], cov, N)
X = np.vstack((X0, X1))
y = np.vstack((np.ones((N,1)), -np.ones((N,1)))).reshape((2*N,))

# Primal problem
from cvxopt import matrix, solvers
from qpsolvers import solve_qp
#build G
c = 100
G0 = np.vstack((-X0.T[0], -X0.T[1], -np.ones(N))).T
G1 = np.vstack((X1.T[0], X1.T[1], np.ones(N))).T
G = np.concatenate((G0, G1), axis = 0)
G2 = np.zeros((n,3))
G = np.concatenate((G,G2),axis = 0)
I = -np.eye(n)
I = np.concatenate((I,I),axis = 0)
G = np.concatenate((G,I),axis = 1)
#build h q P
h = np.concatenate((-np.ones((n,1)), np.zeros((n,1))),axis = 0)
q = np.concatenate((np.zeros((3,1)), c*np.ones((n,1))),axis = 0)
P = np.zeros((n+3,n+3))
P[0][0] = P[1][1] = 1
S = solve_qp(P,q,G,h,solver='cvxopt')
w_primal = S[:2].reshape((1,2))
b_primal = S[2]

```

```

# Dual problem
from cvxopt import matrix, solvers
c = 100

# build P
V = np.concatenate((X0, -X1), axis = 0)
P = matrix(V.dot(V.T))
q = matrix(-np.ones((n, 1)))

# build A, b, G, h
G = matrix(np.vstack((-np.eye(n), np.eye(n))))
h = matrix(np.concatenate((np.zeros((n, 1)), c*np.ones((n, 1))), axis = 0))
A = matrix(y.reshape((n, 1)).T)
b = matrix(np.zeros((1, 1)))
solvers.options['show_progress'] = False
S = solvers.qp(P, q, G, h, A, b)
l = np.array(S['x'])
w_dual = matrix(np.zeros((1, 2)))
SV = 0
e = 1e-5
b_dual = 0

for i in range(0, n):
    if l[i] > e:
        w_dual = w_dual + l[i]*V[i]

for i in range(0, n):
    if l[i] > e and c-l[i] > e:
        SV = SV + 1
        b_dual = b_dual + (y[i]-w_dual.dot(X[i]))

b_dual = b_dual / SV

```

Chi tiết phần thực hành có thể tham khảo thêm ở các notebook đính kèm.

TÀI LIỆU

- [1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [2] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.
- [3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [4] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

- [5] Vũ Hữu Tiệp. Machine learning cơ bản, 2017.