ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN HỌC MÁY
CSC14005_21KHMT2
ĐỒ ÁN CUỐI KỲ
Support vector machine (SVM)

21127592 – Nguyễn Minh Đạt
21127191 – Nguyễn Nhật Truyền

Contents

1. Thông tin nhóm

MSSV	Họ tên
21127592	Nguyễn Minh Đạt
21127191	Nguyễn Nhật Truyền

Phân công công việc

	Công việc	Người phụ trách
Lập trình	Huấn luyện SVM dùng linear kernel	Nguyễn Minh Đạt
	Huấn luyện SVM dùng Gaussian/RBF kernel	Nguyễn Nhật Truyền
	Đánh giá SVM	Nguyễn Nhật Truyền
	Cài đặt thuật toán Random Forest để so sánh kết quả	Nguyễn Nhật Truyền
Báo cáo và Video demo	Cơ sở lý thuyết	Nguyễn Minh Đạt
	Kết quả thực nghiệm	Nguyễn Nhật Truyền

2. Cơ sở lý thuyết

2.1 Cơ sở toán học

2.1.1 Quan hệ giữa điểm và siêu phẳng

Cho:

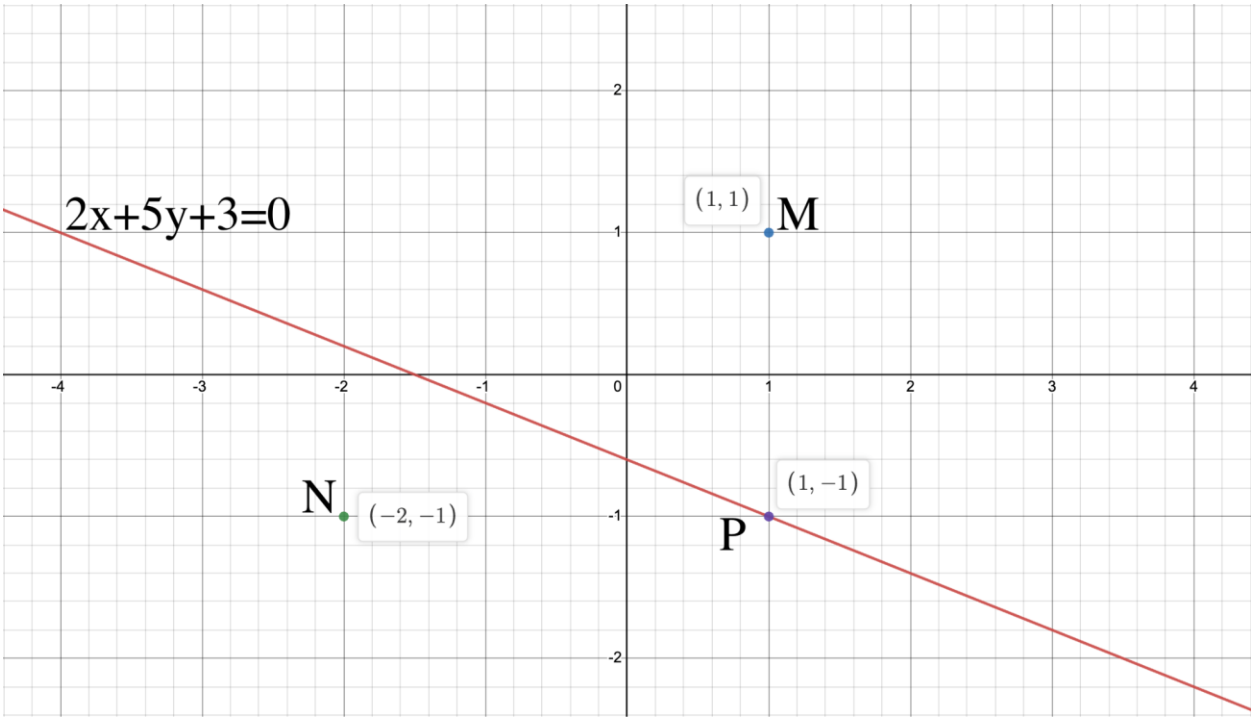
- Điểm $x \epsilon R^D$ có toạ độ $x = (x_1, x_2, \dots, x_D)$ trong không gian D chiều.
- Hàm số $f: R^D \to R$ có dạng $f(x) = w^T x + b$, với $w \epsilon R^D$ và $b \epsilon R$.

Với siêu phẳng $f(x) = 0$, ta có 3 trường hợp:

- Trường hợp 1: $f(x) = 0$. Điểm x **thuộc** siêu phẳng.
- Trường hợp 2: $f(x) > 0$. Điểm x nằm về phía **dương** của siêu phẳng.
- Trường hợp 3: $f(x) < 0$. Điểm x nằm về phía **âm** của siêu phẳng.

Ví dụ: Trong không gian 2 chiều Oxy, cho phương trình đường thẳng $f(x,y) = 2x + 5y + 3 = 0$ và 3 điểm $M(1,1)$, $N(-2,-1)$, $P(1,-1)$.
Ta có:

- Thay toạ độ điểm M vào phương trình, thu được: $f(1,1) = 20 > 0$
  → Điểm M nằm về phía **dương** của đường thẳng
- Thay toạ độ điểm N vào phương trình, thu được: $f(-2,-1) = -6 < 0$
  → Điểm N nằm về phía **âm** của đường thẳng
- Thay toạ độ điểm P vào phương trình, thu được: $f(1,-1) = 0$
  → Điểm P **thuộc** đường thẳng



2.1.2 Khoảng cách từ điểm đến siêu phẳng

Khoảng cách từ 1 điểm dữ liệu có toạ độ $x_a \epsilon D$ đến siêu phẳng $w^T x + b = 0$ được tính theo công thức:

$$distance = \frac{|w^T x_a + b|}{||w||_2}$$

với $||w||_2$ chính là norm-2 của vector w, được tính theo công thức: $||w||_2 = \sqrt{\sum_i^D \quad {w_i}^2}$

2.1.3 Ứng dụng vào bài toán phân lớp nhị phân

Cho điểm dữ liệu có toạ độ x_a, căn cứ vào dấu của $f(x_a) = w^T x_a + b$, chúng ta có thể dự đoán nhãn cho x_a theo quy tắc:

- Nếu $f(x_a) \geq 0$, thì dự đoán $y_a = 1$

- Nếu $f(x_a) < 0$, thì dự đoán $y_a = -1$

Giả sử: Dữ liệu khả tách tuyến tính và siêu phẳng phân loại dữ liệu một cách chính xác.

- Nếu $f(x_a) \geq 0$, thì $y_a = 1$
- Nếu $f(x_a) < 0$, thì $y_a = -1$

Nhận thấy: $f(x_a)$ và y_a cùng dấu. Như vậy, $|f(x_a)| = y_a\big(f(x_a)\big)$

Thay: $f(x_a) = w^T x_a + b$, ta có:

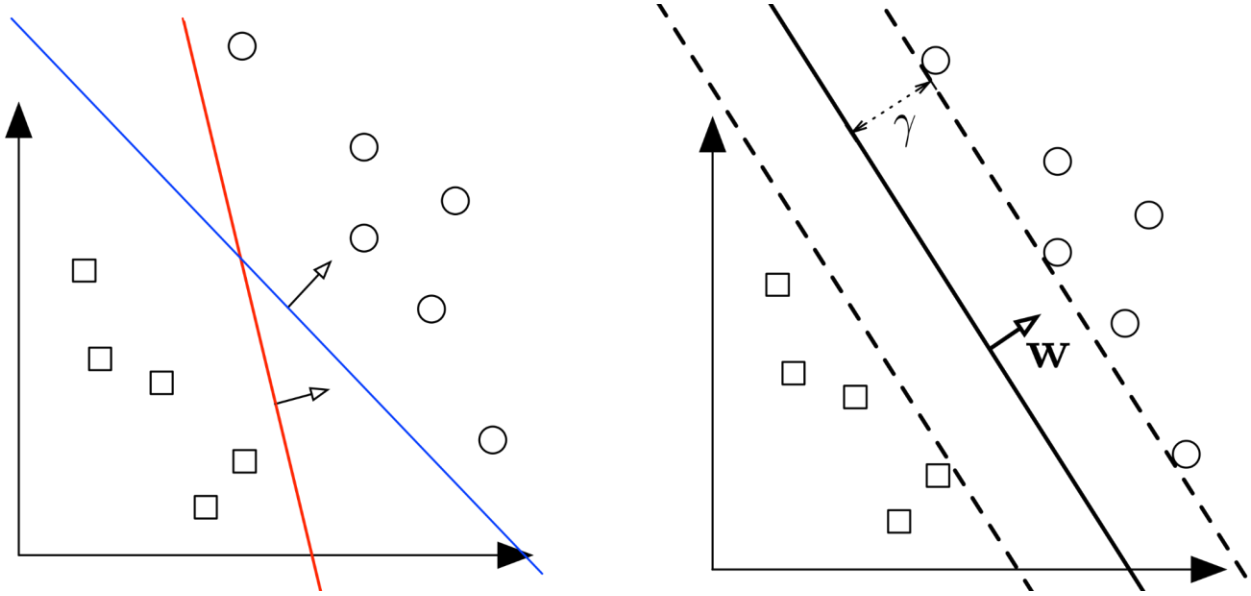
$$|w^T x_a + b| = y_a(w^T x_a + b)$$

Khi đó, khoảng cách từ điểm đến siêu phẳng có thể được tính bằng công thức:

$$distance = \frac{y_a(w^T x_a + b)}{||w||_2}$$

2.2 Thuật toán SVM trong bài toán phân lớp nhị phân

Nhắc lại, để giải bài toán phân lớp nhị phân, chúng ta có thể sử dụng thuật toán Perceptron Learning Algorithm (PLA). Tuy nhiên, đối với dữ liệu khả tách tuyến tính, có vô số siêu phẳng ứng viên có khả năng phân loại dữ liệu một cách chính xác. Như vậy, làm thế nào để tìm được siêu phẳng tốt nhất?



→ Thuật toán SVM xác định siêu phẳng tốt nhất là siêu phẳng làm cho **margin** đạt giá trị lớn nhất giữa hai lớp.

Khái niệm margin:

Margin là khoảng cách từ siêu phẳng đến điểm dữ liệu gần nhất, được tính bởi công thức:

$$margin = min_n\left(\frac{y_n(w^T x_n + b)}{||w||_2}\right) = \frac{1}{||w||_2} min_n(y_n(w^T x_n + b))$$

Chuẩn hoá w, b:

Ví dụ: Cho siêu phẳng có dạng: $2x + 5y + 3 = 0$.

Nhân 2 vế cho số thực $k = 2$, khi đó, $w = 2 \times (2,5) = (4,10)$ và $b = 2 \times 3 = 6$, phương trình siêu phẳng trở thành: $4x + 10y + 6 = 2 \times (2x + 5y + 3) = 0$.

→ Như vậy, siêu phẳng không thay đổi khi chúng ta thay đổi tỉ lệ của w và b theo các giá trị khác nhau.

2.2.1 Giải bài toán SVM sử dụng biên cứng

Bởi vì siêu phẳng không thay đổi khi chúng ta thay đổi tỉ lệ của w và b theo các giá trị khác nhau, chúng ta có thể thay đổi tỉ lệ của w và b sao cho:

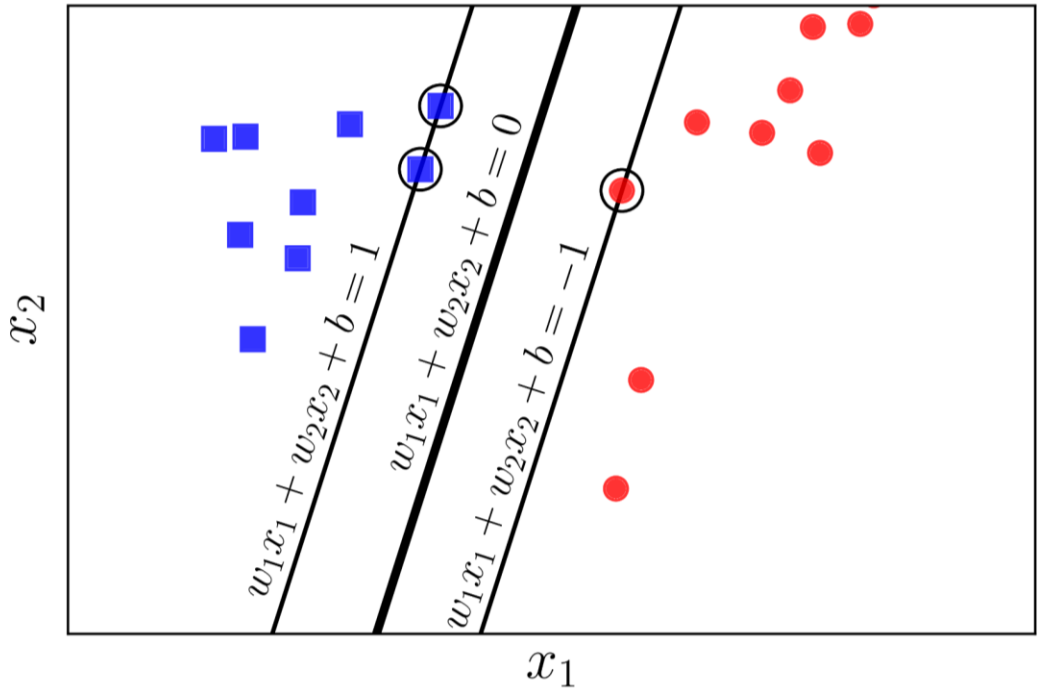
$$min_n(y_n(w^T x_n + b)) = 1$$

Khi đó, margin trở thành:

$$margin = \frac{1}{||w||_2}$$

Hơn nữa, với mọi điểm dữ liệu, ta có điều kiện:

$$y_n(w^T x_n + b) \geq 1, \forall n = 1,2,\ldots,N$$



Phát biểu bài toán:

Tìm w và b sao cho margin đạt giá trị lớn nhất.

$$(w, b) = argmax_{w,b}(margin) = argmax_{w,b}\left(\frac{1}{||w||_2}\right)$$

Thỏa mãn ràng buộc:

$$y_n(w^T x_n + b) \geq 1, \forall n = 1,2,\ldots,N$$

Nhận xét:

- Thay vì maximize $\left(\frac{1}{||w||_2}\right)$, chúng ta có thể minimize $(||w||_2)$. Hơn nữa, chúng ta có thể bình phương $(||w||_2)$ để thu được một hàm khả vi $\left(||w||_2^2\right)$ và nhân với hệ số $\frac{1}{2}$ để thuận lợi cho việc lấy đạo hàm $\left(\frac{\partial}{\partial w}\left(\frac{1}{2}||w||_2^2\right) = \frac{1}{2}\frac{\partial}{\partial w}\left(||w||_2^2\right) = \frac{1}{2}(2w) = w\right)$.
- Viết lại ràng buộc: $y_n(w^T x_n + b) \geq 1 \Leftrightarrow 1 - y_n(w^T x_n + b) \leq 0$

Như vậy, bài toán trở thành:

Primal SVM (hard margin):

$$(w, b) = argmin_{w,b}\left(\frac{1}{2}||w||_2^2\right)$$

subject to: $1 - y_n(w^T x_n + b) \leq 0, \forall n = 1,2,\ldots,N$

Sử dụng **Lagrange** để giải bài toán với trường hợp tổng quát:

$$min_{w,b} f(w, b)$$

subject to: $g_n(w, b) \leq 0, \forall n = 1,2,\ldots,N$

Lagrangian:

$$L(w, b, \lambda) = f(w, b) + \Sigma_n^N \lambda_n g_n(w, b)$$

Thay $f(w, b) = \frac{1}{2}||w||_2^2$ và $g_n(w, b) = 1 - y_n(w^T x_n + b)$, ta có:

Lagrangian của bài toán SVM:

$$L(w, b, \lambda) = \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n (1 - y_n(w^T x_n + b))$$

với: $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T$ và $\lambda_n \geq 0, \forall n = 1,2,\ldots,N$

Biến đổi bài toán trở thành bài toán đối ngẫu.

Đặt:

$$\theta_{primal}(w, b) = max_\lambda\big(L(w, b, \lambda)\big) = max_\lambda\left(f(w, b) + \Sigma_n^N \lambda_n g_n(w, b)\right)$$

Trường hợp 1: w và b thỏa mãn điều kiện.

Ta có: $\forall n, g_n(w, b) = 1 - y_n(w^T x_n + b) \leq 0$

Mà: $\forall n, \lambda_n \geq 0$

Như vậy: $\forall n, \lambda_n g_n(w, b) \leq 0$ (tích hai số trái dấu)

Suy ra: $\theta_{primal}(w, b) = max_\lambda\big(L(w, b, \lambda)\big) = f(w, b) = \frac{1}{2}||w||_2^2$

Trường hợp 2: w và b **không** thỏa mãn điều kiện.

Ta có: $\exists n, g_n(w, b) = 1 - y_n(w^T x_n + b) > 0$

Mà: $\forall n, \lambda_n \geq 0$

Như vậy: $\exists n, \lambda_n g_n(w, b) \geq 0$ (tích hai số cùng dấu)

Suy ra: $\theta_{primal}(w,b) = max_{\lambda}\big(L(w,b,\lambda)\big) = \infty$

Như vậy, trong trường hợp **w** và b thỏa mãn ràng buộc $(g_n(w,b) \le 0, \forall n)$, ta có:

$$\theta_{primal}(w,b) = f(w,b)$$

Tương đương: $min_{w,b}\left(\theta_{primal}(w,b)\right) = min_{w,b}\big(f(w,b)\big)$

$$\Leftrightarrow (w,b) = argmin_{w,b}\left(\frac{1}{2}||w||_2^2\right)$$

→ Do đó, bài toán gốc tương đương với bài toán: $min_{w,b}\left(\theta_{primal}(w,b)\right)$ trong điều kiện **w** và b thỏa mãn ràng buộc.

Tiếp tục đặt:

$$\theta_{dual}(\lambda) = min_{w,b}\big(L(w,b,\lambda)\big) = min_{w,b}\left(f(w,b) + \Sigma_n^N \lambda_n g_n(w,b)\right)$$

Ta có: $max_{\lambda}\big(\theta_{dual}(\lambda)\big) = max_{\lambda}\left(min_{w,b}\big(L(w,b,\lambda)\big)\right)$

Mà: $min_{w,b}\left(\theta_{primal}(w,b)\right) = min_{w,b}\left(max_{\lambda}\big(L(w,b,\lambda)\big)\right)$

Gọi: w^*, b^* là nghiệm của bài toán $min_{w,b}\left(\theta_{primal}(w,b)\right)$

λ^* là nghiệm của bài toán $max_{\lambda}\big(\theta_{dual}(\lambda)\big)$

Sao cho: w^*, b^*, λ^* thoả mãn **điều kiện KKT**:

- $g_n(w^*, b^*) = 1 - y_n(w^{*T}x_n + b^*) \le 0, \forall n = 1,2,\dots,N$
- $\lambda_n^* \ge 0, \forall n = 1,2,\dots,N$
- $\lambda_n^* g_n(w^*, b^*) = \lambda_n^*\big(1 - y_n(w^{*T}x_n + b^*)\big) = 0, \forall n = 1,2,\dots,N$
- $\frac{\partial L(w^*,b^*,\lambda^*)}{\partial w} = 0 \Leftrightarrow w - \Sigma_n^N \lambda_n y_n x_n = 0 \Leftrightarrow w = \Sigma_n^N \lambda_n y_n x_n$
- $\frac{\partial L(w^*,b^*,\lambda^*)}{\partial b} = 0 \Leftrightarrow -\Sigma_n^N \lambda_n y_n = 0 \Leftrightarrow \Sigma_n^N \lambda_n y_n = 0$

Giải điều kiện: $\lambda_n^*\big(1 - y_n(w^{*T}x_n + b^*)\big) = 0, \forall n = 1,2,\dots,N$

Trường hợp 1: $\lambda_n^* = 0$ và $y_n(w^{*T}x_n + b^*) > 1$. Tập hợp những điểm dữ liệu có khoảng cách tới siêu phẳng lớn hơn margin.

Trường hợp 2: $\lambda_n^* > 0$ và $y_n(w^{*T}x_n + b^*) = 1$. Tập hợp những điểm dữ liệu nằm gần siêu phẳng nhất, có khoảng cách tới siêu phẳng đúng bằng margin, còn được gọi là các **support vector**.

Ta được: $min_{w^*,b^*}\left(\theta_{primal}(w^*,b^*)\right) = max_{\lambda^*}\big(\theta_{dual}(\lambda^*)\big) = L\big(w^*,b^*,\lambda^*\big)$

→ Như vậy, chúng ta có thể giải bài toán $max_{\lambda}\big(\theta_{dual}(\lambda)\big)$ thay vì $min_{w,b}\left(\theta_{primal}(w,b)\right)$, mà bài toán $min_{w,b}\left(\theta_{primal}(w,b)\right)$ tương đương với bài toán

gốc trong điều kiện ràng buộc, do đó, nghiệm của bài toán $max_{\lambda}(\theta_{dual}(\lambda))$ cũng là nghiệm cần tìm của bài toán gốc.

Trước hết, chúng ta biến đổi lại hàm Lagrangian:

$$L(w,b,\lambda) = \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n (1 - y_n(w^T x_n + b))$$
$$= \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n (1 - y_n w^T x_n - y_n b))$$
$$= \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n - \Sigma_n^N \lambda_n y_n x_n w^T - \Sigma_n^N \lambda_n y_n b$$

Thay $w = \Sigma_n^N \lambda_n y_n x_n$ và $\Sigma_n^N \lambda_n y_n = 0$ vào Lagrangian, ta thu được hàm $\theta_{dual}(\lambda)$ theo λ:

$$\theta_{dual}(\lambda) = min_{w,b}(L(w,b,\lambda)) = \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n - w w^T - 0$$
$$= \frac{1}{2}||w||_2^2 + \Sigma_n^N \lambda_n - ||w||_2^2$$
$$= \Sigma_n^N \lambda_n - \frac{1}{2}||w||_2^2$$
$$= \Sigma_n^N \lambda_n - \frac{1}{2}\Sigma_n^N \Sigma_m^N \lambda_n \lambda_m y_n y_m x_n^T x_m$$

Như vậy, chúng ta có thể giải bài toán đối ngẫu (dual SVM) thay vì bài toán gốc (primal SVM). Với $g(\lambda) = \theta_{dual}(\lambda)$, bài toán đối ngẫu có dạng:

Dual SVM:

$$\lambda = argmax_{\lambda} g(\lambda)$$
$$\text{subject to:} \quad \lambda \succeq 0 \wedge \Sigma_n^N \lambda_n y_n = 0$$

Tìm w, b:

Sau khi giải bài toán đối ngẫu, tìm được λ, chúng ta tính w và b bằng cách:

- Từ λ vừa tìm được, tính w theo công thức: $w = \Sigma_n^N \lambda_n y_n x_n$
- Thay w vừa tìm được vào một điểm dữ liệu bất kỳ thỏa $\lambda_n > 0 \wedge y_n(w^T x_n + b) = 1$, từ đó tính $b = y_n - w^T x_n$.

Nhận xét:

- Bài toán gốc (primal SVM): Tham số cần tìm là **w** và b. Với **w** có cùng số chiều với x, mà $x \in \Re^D$, do đó, số chiều của **w** là D ($w = [w_1, w_2, \ldots, w_D]$), cùng với hệ số tự do b, như vậy, tổng số lượng tham số cần tìm là: $D + 1$.
- Bài toán đối ngẫu (dual SVM): Tham số cần tìm là λ. Mà chúng ta có N điểm dữ liệu, ứng với mỗi điểm dữ liệu, chúng ta có một nhân tử lagrange, do đó, số chiều của λ là N ($\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]$) như vậy, tổng số lượng tham số cần tìm là: N.
- Như vậy, bài toán đối ngẫu có lợi trong trường hợp số chiều dữ liệu lớn hơn số điểm dữ liệu có được trong training set ($D > N$). Hơn nữa, bài toán đối ngẫu cho

phép dùng kernel, thuận tiện cho việc tính toán và giải quyết trường hợp dữ liệu không khả tách.

2.2.2 Giải bài toán SVM sử dụng biên mềm

Nhắc lại khái niệm biên cứng: Tất cả các điểm dữ liệu được phân loại đúng, và có khoảng cách đến siêu phẳng lớn hơn hoặc bằng margin. Khi đó, chúng ta giải bài toán tối ưu tìm w và b sao cho margin đạt giá trị lớn nhất với ràng buộc cứng $y_n(w^T x_n + b) \geq 1, \forall n$:

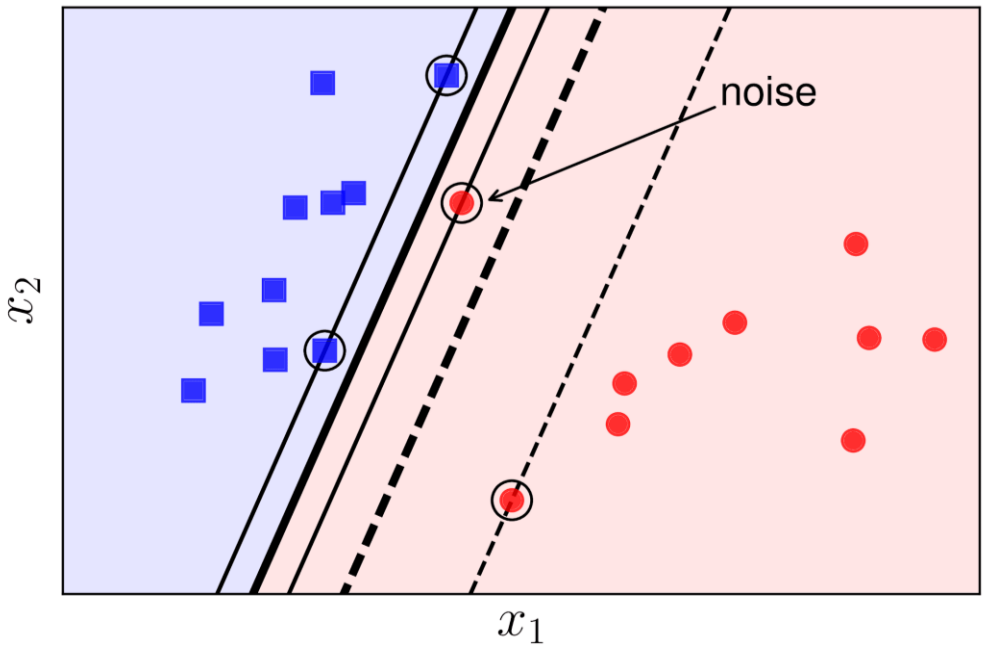
Hard margin:

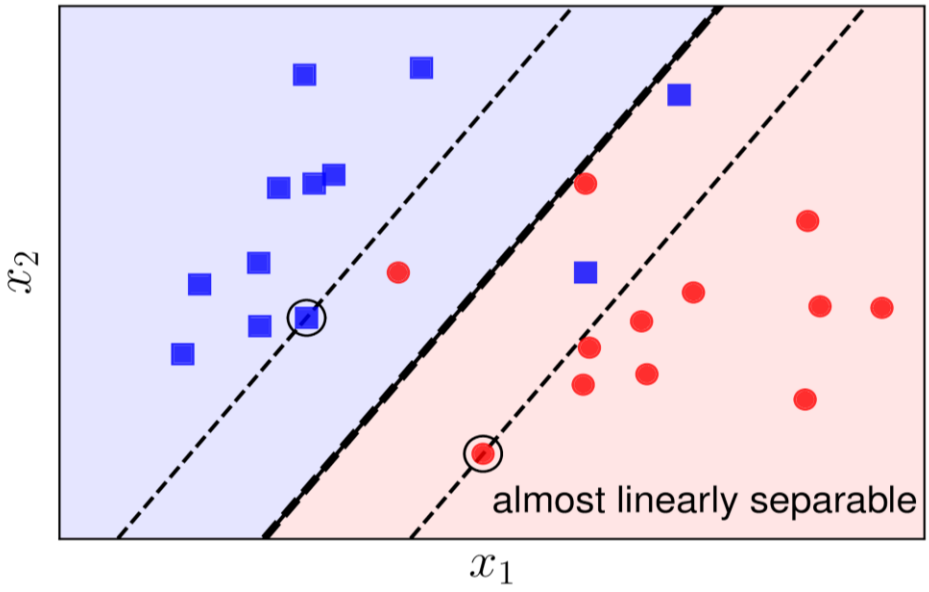
Minimize$_{w,b}$ $\frac{1}{2}||w||^2$

Subject to: $y_n(w^T x_n + b) \geq 1$ for all n = 1, …, N

Tuy nhiên, tồn tại một số trường hợp mà sử dụng bài toán hard margin để giải là không hiệu quả. Chẳng hạn như khi:

- Dữ liệu không khả tách (không tồn tại margin).
- Dữ liệu khả tách nhưng tồn tại một điểm dữ liệu thuộc lớp này nằm rất gần các điểm dữ liệu của lớp khác (margin rất nhỏ).

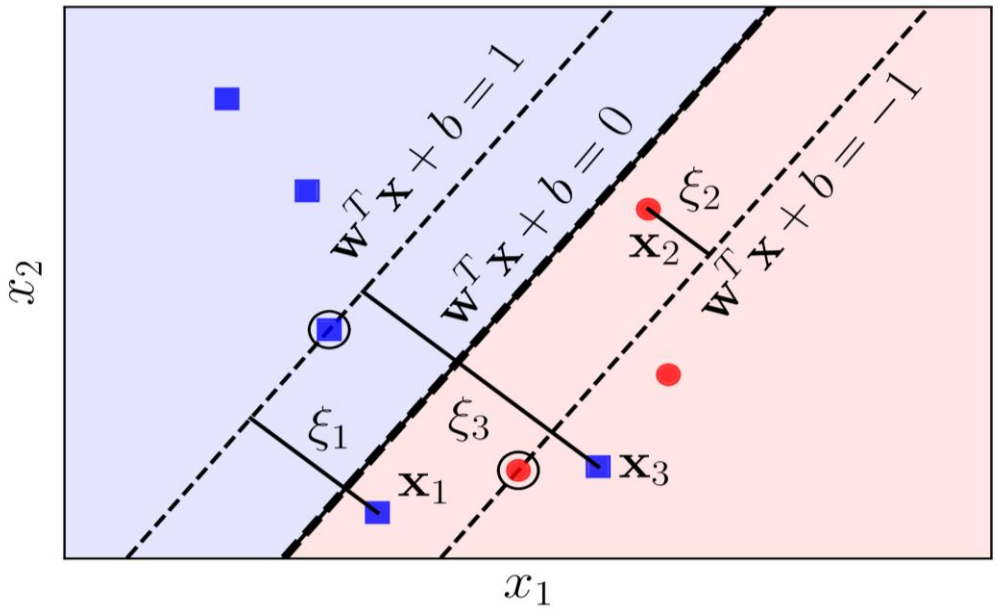




Biến slack:

Để giải bài toán SVM hiệu quả trong những trường hợp nêu trên, chúng ta cần phải "hy sinh", nghĩa là cho phép một số điểm dữ liệu rơi vào vùng margin, hoặc thậm chí bị phân loại sai (nằm phía bên kia siêu phẳng phân cách). Để có thể định lượng sự "hy sinh" này, mỗi điểm dữ liệu sẽ có một biến **slack** (kí hiệu: ξ), với các đặc điểm sau:

- Với những điểm dữ liệu được phân loại đúng, và có khoảng cách đến siêu phẳng lớn hơn hoặc bằng margin thì $\xi = 0$.
- Với những điểm dữ liệu được phân loại đúng, tuy nhiên có khoảng cách đến siêu phẳng bé hơn margin thì $0 < \xi < 1$.
- Với những điểm dữ liệu bị phân loại sai, thì $\xi > 1$.



Nhận xét:

- Đối với bài toán hard margin, để maximize margin, chúng ta minimize giá trị $\frac{1}{2}||w||^2$. Ở bài toán soft margin, bên cạnh minimize giá trị $\frac{1}{2}||w||^2$ để maximize margin, chúng ta cần phải minimize một giá trị nữa, đó chính là tổng hy sinh: $\sum_n^N \xi_n$. Như vậy, **hàm mục tiêu** của chúng ta trở thành: $\frac{1}{2}||w||^2 + C\sum_n^N \xi_n$, với C là hằng số được dùng để điều chỉnh tầm quan trọng giữa margin và sự hy sinh $(C > 0)$.
- Thay vì sử dụng ràng buộc cứng $y_n(w^T x_n + b) \geq 1, \forall n$. Chúng ta sử dụng ràng buộc mềm với sự xuất hiện của biến slack: $y_n(w^T x_n + b) \geq 1 - \xi_n, \forall n$ và ràng buộc phụ: $\xi_n \geq 0, \forall n$.

Soft margin:

Minimize$_{w,b,\xi}$ $\frac{1}{2}||w||^2 + C\sum_n^N \xi_n$

Subject to: $1 - \xi_n - y_n(w^T x_n + b) \leq 0$ and $-\xi_n \leq 0$ for all n = 1, …, N

Lagrangian:

$$L(w, b, \xi, \lambda, \mu) = \frac{1}{2}||w||^2 + C\sum_n^N \xi_n + \sum_n^N \lambda_n(1 - \xi_n - y_n(w^T x_n + b)) + \sum_n^N \mu_n(-\xi_n)$$

với: $\lambda \geq 0 \wedge \mu \geq 0$

Tương tự như giải bài toán hard margin:

Đặt:

$$\theta_{primal}(w, b, \xi) = max_{\lambda,\mu}\big(L(w, b, \xi, \lambda, \mu)\big)$$

Với w, b, ξ thoả: $(1 - \xi_n - y_n(w^T x_n + b) \leq 0) \wedge (-\xi_n \leq 0)$

Ta có: $\theta_{primal}(w, b, \xi) = max_{\lambda,\mu}\big(L(w, b, \xi, \lambda, \mu)\big) = \frac{1}{2}||w||^2 + C\sum_n^N \xi_n$

Như vậy: $min_{w,b,\xi}\left(\theta_{primal}(w, b, \xi)\right) = min_{w,b,\xi}\left(\frac{1}{2}||w||^2 + C\sum_n^N \xi_n\right)$

→ Bài toán soft margin tương đương với bài toán: $min_{w,b,\xi}\left(\theta_{primal}(w, b, \xi)\right)$

Tiếp tục đặt:

$$\theta_{dual}(\lambda, \mu) = min_{w,b,\xi}\big(L(w, b, \xi, \lambda, \mu)\big)$$

Ta có: $max_{\lambda,\mu}\big(\theta_{dual}(\lambda, \mu)\big) = max_{w,b,\xi}\left(min_{w,b,\xi}\big(L(w, b, \xi, \lambda, \mu)\big)\right)$

Mà: $min_{w,b,\xi}\left(\theta_{primal}(w, b, \xi)\right) = min_{w,b,\xi}\left(max_{\lambda,\mu}\big(L(w, b, \xi, \lambda, \mu)\big)\right)$

Gọi: λ^*, μ^* là nghiệm của bài toán $max_{\lambda,\mu}\big(\theta_{dual}(\lambda, \mu)\big)$

w^*, b^*, ξ^* là nghiệm của bài toán $min_{w,b,\xi}\left(\theta_{primal}(w, b, \xi)\right)$

Sao cho: $\lambda^*, \mu^*, w^*, b^*, \xi^*$ thoả điều kiện KKT

- $1 - \xi_n^* - y_n(w^{*T} x_n + b^*) \leq 0$
- $-\xi_n^* \leq 0$
- $\lambda_n^* \geq 0$
- $\mu_n^* \geq 0$
- $\lambda_n^*\left(1 - \xi_n^* - y_n(w^{*T} x_n + b^*)\right) = 0$
- $\mu_n^*(-\xi_n^*) = 0$
- $\frac{\partial L}{\partial w} = 0 \Leftrightarrow w - \Sigma_n^N \lambda_n y_n x_n = 0 \Leftrightarrow w = \Sigma_n^N \lambda_n y_n x_n$
- $\frac{\partial L}{\partial b} = 0 \Leftrightarrow -\Sigma_n^N \lambda_n y_n = 0 \Leftrightarrow \Sigma_n^N \lambda_n y_n = 0$
- $\frac{\partial L}{\partial \xi} = 0 \Leftrightarrow C - \lambda_n - \mu_n = 0 \Leftrightarrow \lambda_n = C - \mu_n$

Ta được:

$$max_{\lambda^*,\mu^*}\left(\theta_{dual}(\lambda^*, \mu^*)\right) = min_{w^*,b^*,\xi^*}\left(\theta_{primal}(w^*, b^*, \xi^*)\right) = L(w^*, b^*, \xi^*, \lambda^*, \mu^*)$$

→ Như vậy, chúng ta có thể giải bài toán đối ngẫu $max_{\lambda,\mu}\left(\theta_{dual}(\lambda, \mu)\right)$ thay vì bài toán soft margin gốc. Thay $w = \Sigma_n^N \lambda_n y_n x_n$, $\Sigma_n^N \lambda_n y_n = 0$ và $\lambda_n = C - \mu_n$ vào Lagrangian, ta được:

$$\theta_{dual}(\lambda, \mu) = \sum_n^N \lambda_n - \frac{1}{2}\sum_n^N \sum_m^N \lambda_n \lambda_m y_n y_m x_n^T x_m$$

Nhận xét: $\theta_{dual}(\lambda, \mu)$ là hàm theo λ và không phụ thuộc μ. Đặt $g(\lambda) = \theta_{dual}(\lambda, \mu)$, chúng ta phát biểu bài toán đối ngẫu:

Dual SVM (Soft margin):

$$\lambda = argmax_\lambda g(\lambda)$$

subject to: $0 \leq \lambda_n \leq C \wedge \Sigma_n^N \lambda_n y_n = 0$

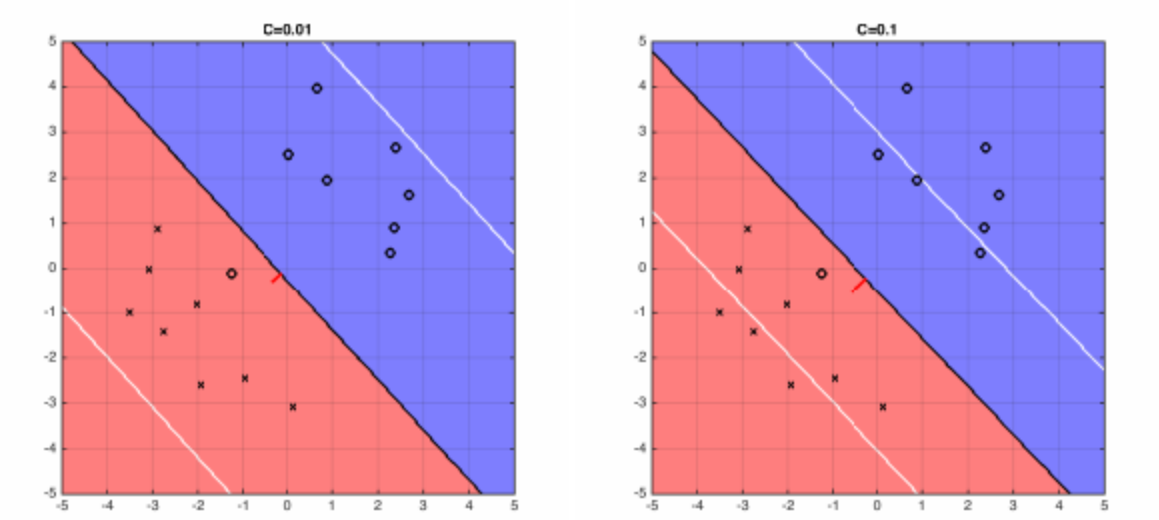
Tìm w, b,

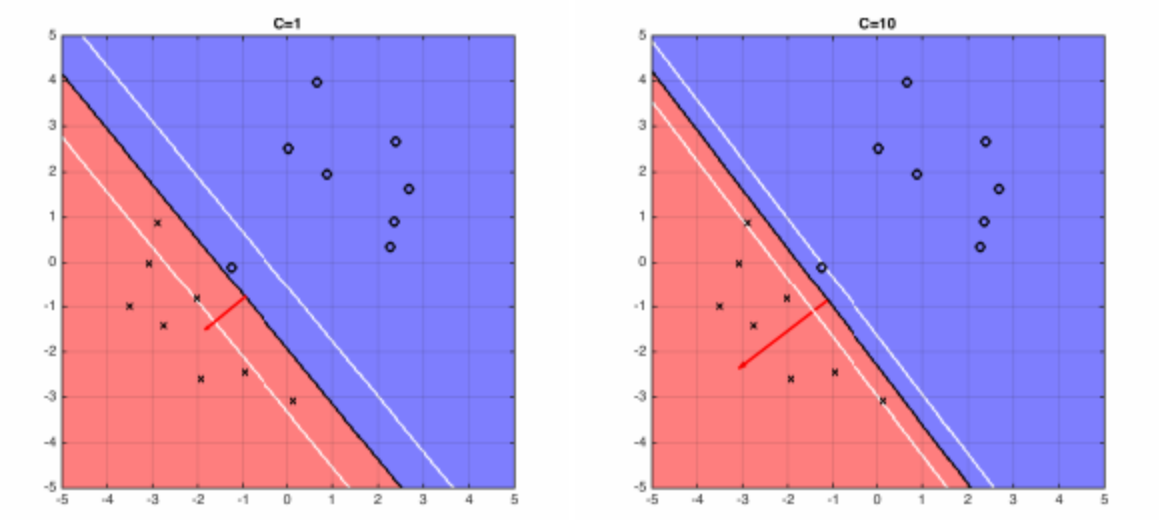
Giải bài toán đối ngẫu, chúng ta thu được λ, từ đó tính xác định w và b bằng cách:

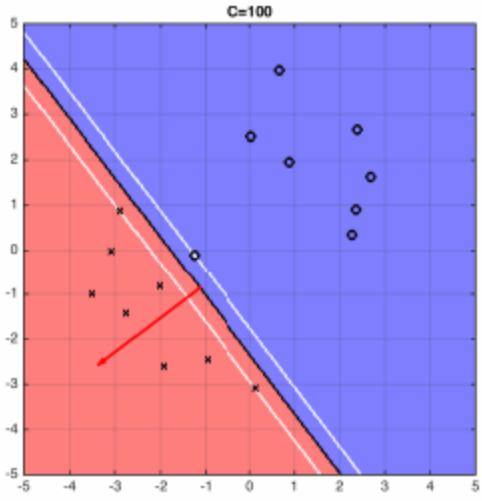
- $w = \Sigma_n^N \lambda_n y_n x_n$
- Thay w vừa tìm được vào một điểm dữ liệu bất kỳ thỏa $\lambda_n > 0$, từ đó tính $b = y_n - w^T x_n$.

Ý nghĩa tham số C (C > 0): C là tham số regularization,

- với C càng **lớn**, thuật toán SVM trở nên càng nghiêm ngặt, và cố gắng phân loại đúng tất cả các điểm dữ liệu,
- trái lại, với C càng **nhỏ**, thuật toán SVM trở nên càng lỏng lẻo và có thể cho cho phép phân loại sai một số điểm dữ liệu để thu được nghiệm đơn giản hơn (giá trị $||w||^2$ nhỏ hơn).


C=0.01
C=0.1


C=1
C=10


C=100

2.2.3 Giải bài toán SVM sử dụng Kernel

Xét bài toán đối ngẫu:

$$\lambda = argmax_{\lambda}\left(\sum_{n}^{N}\lambda_n - \frac{1}{2}\sum_{n}^{N}\sum_{m}^{N}\lambda_n\lambda_m y_n y_m x_n^T x_m\right)$$

$$\text{subject to:} \quad 0 \le \lambda_n \le C \wedge \Sigma_n^N \lambda_n y_n = 0$$

Thay $x_i = \phi(x_i)$, bài toán trở thành:

$$\lambda = argmax_{\lambda}\left(\sum_{n}^{N}\lambda_n - \frac{1}{2}\sum_{n}^{N}\sum_{m}^{N}\lambda_n\lambda_m y_n y_m \phi(x_n)^T\phi(x_m)\right)$$

$$\text{subject to:} \quad 0 \le \lambda_n \le C \wedge \Sigma_n^N \lambda_n y_n = 0$$

Nhận thấy:

- Bài toán chỉ thay đổi duy nhất một chi tiết: ở trong hàm mục tiêu, tích vô hướng $x_n^T x_m$ được thay thế bằng một tích vô hướng mới $\phi(x_n)^T\phi(x_m)$. Như vậy, nếu các điểm dữ liệu ban đầu **không** khả tách tuyến tính, chúng ta có thể tìm một ánh xạ ϕ để biến đổi dữ liệu sang không gian mới sao cho ở trong không gian đó, các điểm dữ liệu trở nên khả tách tuyến tính.
- Thậm chí, với mỗi điểm dữ liệu x_i, chúng ta không cần tính trực tiếp $\phi(x_i)$. Cho hai điểm dữ liệu ban đầu x_n và x_m, chúng ta có thể tính tích vô hướng trong không gian mới $\phi(x_n)^T\phi(x_m)$ sử dụng kỹ thuật kernel trick.

Định nghĩa hàm kernel:

$$k(x_n, x_m) = \phi(x_n)^T\phi(x_m)$$

Kernel trick:

Tính chất:

Xét ma trận K có kích thước $N \times N$, với mỗi phần tử $K_{n,m} = k(x_n, x_m)$. Ma trận K có hai tính chất: K là ma trận **đối xứng** và K là ma trận **nửa xác định dương**.

Một số kernel thông dụng:

- Linear:

  $$k(x, z) = x^{\mathrm{T}}z$$

- Polynomial:

  $$k(x, z) = (r + \gamma x^{\mathrm{T}}z)^{\mathrm{d}}$$

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

- Radial Basic Function (RBF):

RBF kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế và là kernel mặc định trong sklearn.

$k(x,z) = exp(-\gamma||x-z||_2^2)$, $\gamma > 0$

- Sigmoid:

$k(x,z) = tanh(\gamma x^{\mathrm{T}} z + r)$

2.3 Thuật toán SVM trong bài toán phân đa lớp

Cho bài toán phân loại k lớp, với dữ liệu huấn luyện $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, trong đó $y_n \in \{1,2,\ldots,k\}$ là lớp của x_n.

2.3.1 Phương pháp one-against-all

Ý tưởng:

Từ bộ dữ liệu huấn luyện đa lớp ban đầu, xây dựng các bộ dữ liệu huấn luyện ở dạng nhị phân, mỗi bộ dữ liệu dành riêng cho 1 lớp với các điểm dữ liệu thuộc lớp đó được gán nhãn dương, còn các điểm dữ liệu thuộc lớp khác gán nhãn âm. Từ đó, huấn luyện k mô hình SVM cho k bài toán phân loại nhị phân tương ứng.

Ví dụ:

Cho bộ dữ liệu $(x_1, A), (x_2, B), (x_3, A), (x_4, C), (x_5, C)$ với 3 lớp A, B, C.

Xây dựng 3 bộ dữ liệu ở dạng nhị phân tương ứng:

- $(x_1, A), (x_2, notA), (x_3, A), (x_4, notA), (x_5, notA)$
- $(x_1, notB), (x_2, B), (x_3, notB), (x_4, notB), (x_5, notB)$
- $(x_1, notC), (x_2, notC), (x_3, notC), (x_4, C), (x_5, C)$

Từ đó, huấn luyện 3 mô hình SVM cho bài toán phân loại nhị phân tương ứng:

- Mô hình thứ 1: Phân loại dữ liệu thuộc lớp A với các dữ liệu không thuộc lớp A.
- Mô hình thứ 2: Phân loại dữ liệu thuộc lớp B với các dữ liệu không thuộc lớp B.
- Mô hình thứ 3: Phân loại dữ liệu thuộc lớp C với các dữ liệu không thuộc lớp C.

Giải bài toán SVM sử dụng biên mềm cho mô hình thứ m-th ứng với lớp $y_n = m$:

$$min_{w^m, b^m, \xi^m} \left(\frac{1}{2} ||w^m||_2^2 + C \sum_n^N \xi_n^m \right)$$

subject to: $(w^{mT} \phi(x_n) + b^m) \geq 1 - \xi_n^m, if\ y_n = m$

$(w^{mT} \phi(x_n) + b^m) < 1 - \xi_n^m, if\ y_n \neq m$

$\xi_n^m \geq 0$

Như vậy, ứng với k mô hình, chúng ta thu được k hàm quyết định:

$$w^{1^T}\phi(x) + b^1$$
$$\dots$$
$$w^{k^T}\phi(x) + b^k$$

Lớp của x được xác định bởi hàm quyết định có giá trị lớn nhất, tính bằng công thức:

$$y = argmax_m\left(w^{m^T}\phi(x_n) + b^m\right)$$

2.3.2 Phương pháp one-against-one

Ý tưởng:

Giống với one-against-all, phương pháp one-against-one cũng xây dựng các bộ dữ liệu huấn luyện ở dạng nhị phân. Tuy nhiên, phương pháp one-against-one thực hiện bắt cặp hai lớp khác nhau để tạo thành một bộ dữ liệu nhị phân. Như vậy, với bộ dữ liệu huấn luyện ban đầu bao gồm k lớp, phương pháp one-against-one sẽ xây dựng $\frac{k(k-1)}{2}$ bộ dữ liệu nhị phân tương ứng.

Ví dụ:

Cho bộ dữ liệu $(x_1, A), (x_2, B), (x_3, A), (x_4, C), (x_5, C)$ với 3 lớp A, B, C.

Xây dựng $\frac{3\times2}{2} = 3$ bộ dữ liệu ở dạng nhị phân tương ứng theo phương pháp bắt cặp:

- Lớp A và Lớp B: $(x_1, A), (x_2, B), (x_3, A)$
- Lớp A và Lớp C: $(x_1, A), (x_3, A), (x_4, C), (x_5, C)$
- Lớp B và Lớp C: $(x_2, B), (x_4, C), (x_5, C)$

Từ đó, chúng ta cũng huấn luyện 3 mô hình SVM cho bài toán phân loại nhị phân tương ứng.

- Mô hình thứ 1: Phân loại dữ liệu thuộc lớp A với dữ liệu thuộc lớp B.
- Mô hình thứ 2: Phân loại dữ liệu thuộc lớp A với dữ liệu thuộc lớp C.
- Mô hình thứ 3: Phân loại dữ liệu thuộc lớp B với dữ liệu thuộc lớp C.

Giải bài toán SVM sử dụng biên mềm cho mô hình ứng với hai lớp y_i, y_j:

$$min_{w^{ij}, b^{ij}, \xi^{ij}}\left(\frac{1}{2}||w^{ij}||_2^2 + C\sum_n^N \xi_n^{ij}\right)$$

subject to:
$$(w^{ij^T}\phi(x_n) + b^{ij}) \geq 1 - \xi_n^{ij}, if\ y_n = i$$
$$(w^{ij^T}\phi(x_n) + b^{ij}) < 1 - \xi_n^{ij}, if\ y_n = j$$
$$\xi_n^{ij} \geq 0$$

Để xác định lớp cho điểm dữ liệu x, chúng ta sử dụng chiến lược "bỏ phiếu", nghĩa là mỗi mô hình SVM liên quan sẽ có 1 dự đoán về lớp của x, từ đó, x được dự đoán thuộc về lớp có nhiều phiếu bầu nhất.

3. Kết quả thực nghiệm

Tác vụ: Huấn luyện SVM để phân lớp ảnh thời trang:

Mô tả dữ liệu:

- Bộ dữ liệu được sử dụng là bộ Fashion MNIST.

- Kích thước:
  - Bộ dữ liệu bao gồm tổng cộng 70,000 hình ảnh.
  - 60,000 hình ảnh được sử dụng để huấn luyện mô hình.
  - 10,000 hình ảnh được sử dụng để kiểm tra và đánh giá hiệu suất mô hình.
- Loại hình ảnh:
  - Mỗi hình ảnh trong bộ dữ liệu là ảnh xám (grayscale).
  - Kích thước của mỗi hình ảnh là 28x28 pixel.
- Danh mục phân loại:
  - Có tổng cộng 10 danh mục phân loại khác nhau, tương ứng với 10 loại sản phẩm thời trang.
  - Các danh mục bao gồm: áo thun, quần dài, áo len, váy, áo khoác, dép, túi xách, ổ cắm giày, đồng hồ và áo sơ mi.

- Mục tiêu:
    - Mục tiêu của bài toán là phân loại đúng loại sản phẩm thời trang từ hình ảnh.

Cài đặt SVM:

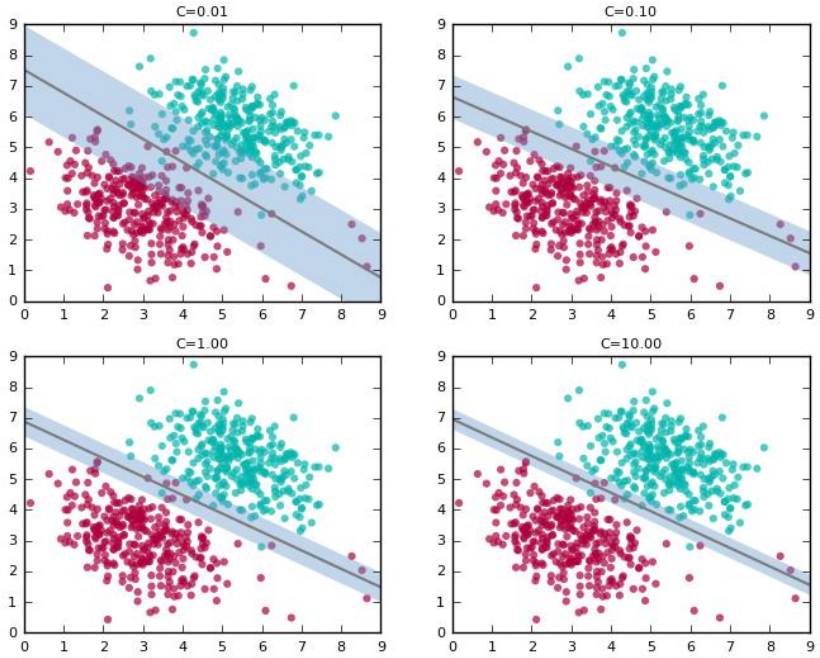
- Sử dụng SVM đã được cài đặt sẵn trong thư viện scikit-learn.
- Thực hiện trên Jupyter Notebook (anaconda3).

3.1 Huấn luyện SVM:

Ở lần này chúng em sẽ thử nghiệm mô hình SVM với 2 kernel (linear kernel và RBF kerel).

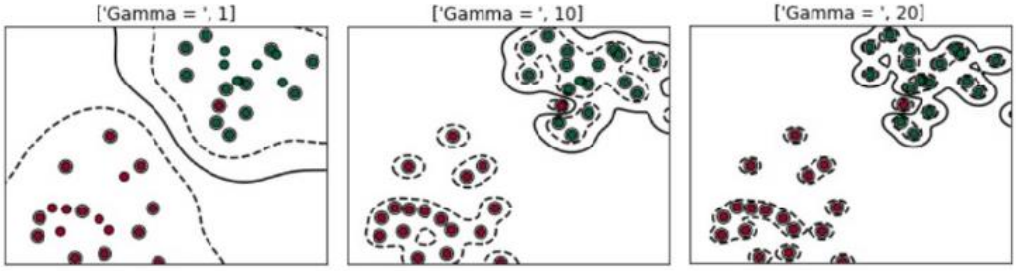
Thử nghiệm với các giá trị khác nhau của siêu tham số C và γ; với mỗi giá trị C và γ, ghi nhận lại: độ lỗi trên tập training, độ lỗi trên tập validation, thời gian huấn luyện.

- Tham số C:
    + C là tham số kiểm soát độ nới lỏng của ranh giới hỗ trợ (margin) trong mô hình SVM.
    + Khi C lớn, mô hình sẽ cố gắng giữ cho mọi điểm dữ liệu đúng mức có thể, tạo ra một ranh giới hỗ trợ chặt chẽ.
    + Khi C nhỏ, mô hình sẽ chấp nhận sự vi phạm của một số điểm dữ liệu để tạo ra ranh giới hỗ trợ tổng quát hơn.



- Tham số gamma:
    + Gamma kiểm soát hình dạng của hàm nhân (kernel) và ảnh hưởng đến sự nhạy cảm của mô hình đối với các điểm dữ liệu.

+ Giá trị gamma lớn có thể dẫn đến một mô hình phức tạp với khả năng overfitting, đặc biệt là khi có ít dữ liệu.
+ Giá trị gamma nhỏ giúp mô hình tạo ra ranh giới hỗ trợ mềm mại và phù hợp với dữ liệu đa dạng.



- Linear kernel:
  - Tham số thử nghiệm: thử nghiệm với các giá trị khác nhau của siêu tham số C. Với 2 mô hình SVC và LinearSVC.
  - Kết quả:

    SVC

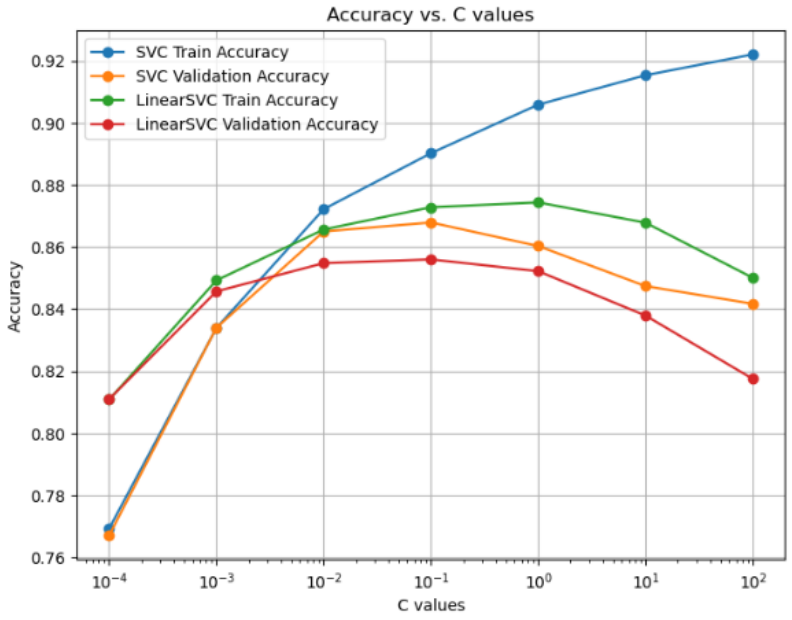
C	accuracy_validation	train_time(s)
1e-4	0.7671	611.832612991333
1e-3	0.834	291.33587098121643
1e-2	0.865	183.40004897117615
1e-1	0.8679	155.61584377288818
1.00	0.8603	195.36884450912476
10.00	0.8474	472.7568690776825
100.00	0.8417	3441.5078756809235

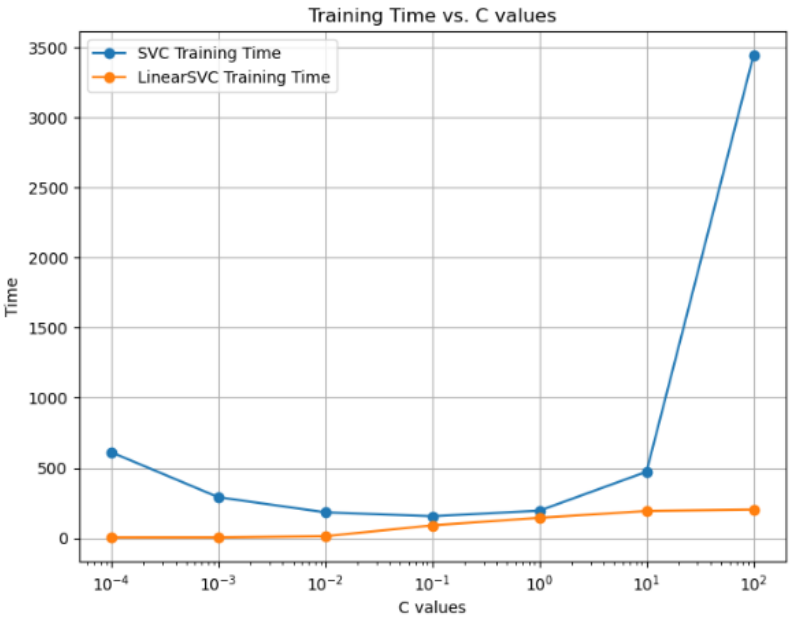
    LinearSVC:

C	accuracy_validation	train_time
1e-4	0.811	4.1716320514678955
1e-3	0.8457	4.404699325561523
1e-2	0.8548	13.205413103103638

1e-1	0.856	90.53393483161926
1.00	0.8522	144.46867680549622
10.00	0.8379	192.79570245742798
100.00	0.8175	203.49650645256042

- Visualize:





- Nhận xét:
    + Tăng C giảm độ chính xác trên tập validation:
      Có thể thấy rằng ban đầu khi tăng C giá trị accuracy có tăng. Tuy nhiên hầu hết độ chính xác trên tập validation giảm khi giá trị của C tăng lên mức cao. Điều này có thể xuất phát từ việc mô hình trở nên quá phức tạp với giá trị C lớn, dẫn đến việc nó quá fit với dữ liệu huấn luyện và không tốt trên dữ liệu mới.
    + Hiệu suất tốt ở giá trị C trung bình:
      Giá trị C khoảng 1e-2 hoặc 1e-1 có vẻ cho kết quả tốt nhất trên tập validation với độ chính xác khoảng từ 0.865 đến 0.8679 (Với mô hình SVC).
    + Thời gian huấn luyện giảm khi C giảm:
      Thời gian huấn luyện (train time) giảm khi giảm giá trị của C. Điều này là lợi ích của việc giảm phức tạp của mô hình khi giá trị C nhỏ.
    + Giảm hiệu suất khi C quá nhỏ:
      Khi giá trị của C quá nhỏ (ví dụ, 1e-4), độ chính xác trên tập validation giảm đáng kể. Điều này có thể là dấu hiệu của việc mô hình quá đơn giản và không đủ mạnh mẽ để khái quát hóa tốt trên dữ liệu mới.
    + Thời gian huấn luyện tăng đột ngột ở giá trị C cao (SVC):
      Khi giá trị C tăng lên (ví dụ, 100.00), thời gian huấn luyện tăng đáng kể. Điều này có thể là do mô hình trở nên quá phức tạp và yêu cầu nhiều thời gian để tìm ra giải pháp tối ưu trong quá trình huấn luyện.

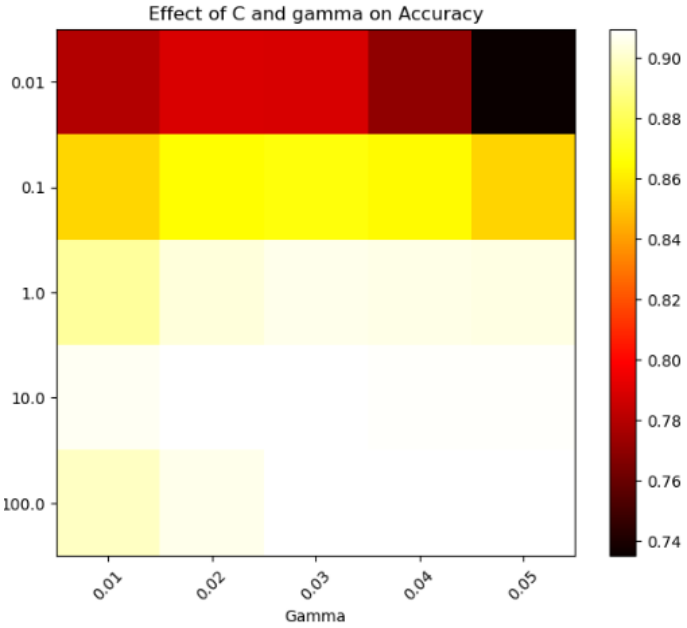
- RBF kernel:
    - Tham số thử nghiệm: thử nghiệm với các giá trị khác nhau của siêu tham số C và gamma. Trên mô hình SVC.
    - Kết quả:

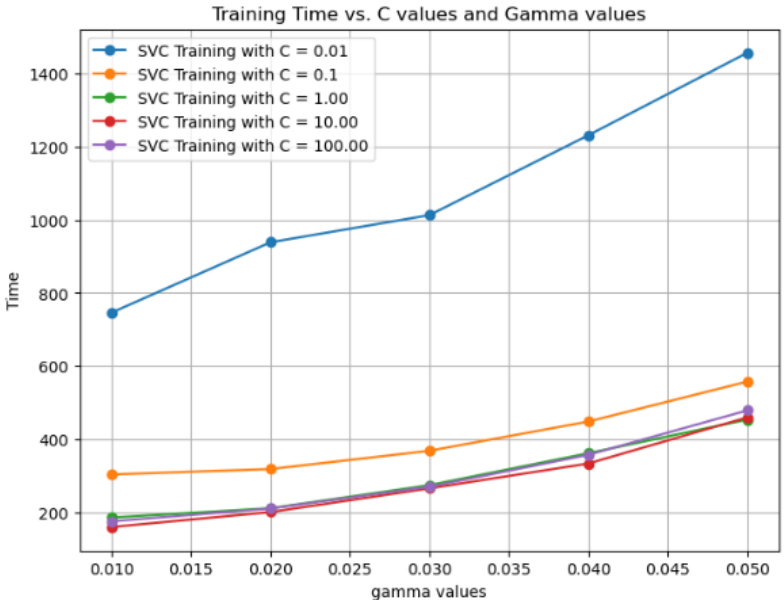
Tham số		accuracy_train	accuracy_validation	train_time(s)
C = 0.01	gamma = 0.01	0.7802	0.779	746.2768030166626
	gamma = 0.02	0.79006	0.7891	938.7034120559692
	gamma = 0.03	0.78712	0.7883	1012.9219031333923

	gamma = 0.04	0.772	0.7699	1231.4561281204224
	gamma = 0.05	0.74172	0.7351	1456.9363102912903
C = 0.1	gamma = 0.01	0.85394	0.8546	303.2974109649658
	gamma = 0.02	0.86798	0.8648	318.1141231060028
	gamma = 0.03	0.8725	0.8671	368.07842540740967
	gamma = 0.04	0.87358	0.8643	448.0881128311157
	gamma = 0.05	0.86862	0.8537	557.3114976882935
C = 1.00	gamma = 0.01	0.91042	0.8918	185.3846881389618
	gamma = 0.02	0.93472	0.9032	210.80760216712952
	gamma = 0.03	0.9505	0.9058	273.9119906425476
	gamma = 0.04	0.961	0.9053	361.33776354789734
	gamma = 0.05	0.9691	0.9041	453.3405635356903
C = 10.00	gamma = 0.01	0.97132	0.9072	159.54243516921997
	gamma = 0.02	0.99378	0.9089	200.40108156204224
	gamma = 0.03	0.99892	0.9091	265.26370453834534
	gamma = 0.04	0.99992	0.9082	333.1737928390503
	gamma = 0.05	0.99998	0.9083	459.3322823047638
C = 100.00	gamma = 0.01	0.99974	0.8987	175.10702395439148

	gamma = 0.02	1.0	0.9054	210.16954684257507
	gamma = 0.03	1.0	0.9094	269.7428960800171
	gamma = 0.04	1.0	0.9091	478.42067790031433
	gamma = 0.05	1.0	0.9089	478.42067790031433

- Visualize:





- Nhận xét:
    + Hiệu suất trên tập validation:
      Mô hình có hiệu suất tốt trên tập validation ở mức độ cao của C và gamma, đặc biệt là khi C = 10.0 và gamma = 0.03, 0.04, 0.05, với độ chính xác lên đến 90.91%.
    + Tăng C và gamma cũng tăng độ chính xác:
      Đối với các giá trị C và gamma tăng lên, độ chính xác trên cả tập huấn luyện và tập validation cũng tăng.
    + Thời gian huấn luyện tăng khi C và gamma tăng:
      Thời gian huấn luyện tăng đáng kể khi giá trị của C và gamma tăng. Điều này là do mô hình trở nên phức tạp hơn và yêu cầu nhiều thời gian hơn để tìm ra giải pháp tối ưu.
    + Hiệu suất giảm khi gamma quá lớn:
      Có dấu hiệu của hiệu suất giảm khi gamma quá lớn (ví dụ, gamma = 0.05). Điều này có thể là dấu hiệu của việc mô hình trở nên quá nhạy cảm đối với các điểm dữ liệu và không tốt trên dữ liệu mới.
    + Độ chính xác trên tập huấn luyện và tập validation:
      Có sự chênh lệch nhỏ giữa độ chính xác trên tập huấn luyện và tập validation, điều này cho thấy mô hình không bị overfitting mạnh và vẫn khá tốt khi áp dụng cho dữ liệu mới.

- Chọn hàm dự đoán có độ lỗi nhỏ nhất trên tập validation là hàm dự đoán cuối cùng:
    - Hàm dự đoán có độ lỗi nhỏ nhất trên tập validation là:
      SVC(C=100.00, gamma=0.03)
      Với accuracy_validation = 0.9094
    - Áp dụng mô hình lên test_set:

```
predicted_test = best_classifier.predict(test_set)
accuracy_test = accuracy_score(test_labels, predicted_test)
print("\naccuracy score with train_set: ", accuracy_test)
```

```
accuracy score with train_set:  0.8988
```

*accuracy score with test_set : 0.8988

3.2 Đánh giá SVM:

Ở đây nhóm chúng em có cài đặt mô hình random forest để so sánh. Với mô hình random forest chúng em thử nghiệm 2 tham số n_estimators=100, max_features='sqrt'. Trong đó n_estimators là số lượng cây và max_features là số lượng các đặt trưng được xem xét khi tìm kiếm phấn chia tốt nhất trong 1 cây.

Thu được kết quả:

	SVM	Random Forest
accuracy	0.8988	0.8739
training_time	269.7428960800171	102.97511744499207

Nhận xét:

- Mô hình SVM có độ chính xác cao hơn so với mô hình Random Forest (89.88% so với 87.39%). Tuy nhiên, điều này đi kèm với chi phí thời gian huấn luyện lớn hơn (269.74 giây so với 102.98 giây của Random Forest). Sự lựa chọn giữa độ chính xác và thời gian huấn luyện phụ thuộc vào yêu cầu cụ thể của ứng dụng.
- SVM thường hiệu quả khi có các biên phân loại phức tạp trong không gian đặc trưng. Nó có khả năng tốt trong việc xử lý các mối quan hệ phi tuyến tính giữa các đặc trưng. Ngược lại, Random Forest cũng là một lựa chọn mạnh mẽ, nhất là khi có sự tương tác phức tạp giữa các đặc trưng.
- Việc Random Forest có thời gian huấn luyện ngắn hơn có thể là dấu hiệu của việc nó không quá phức tạp, nhưng cũng có thể dẫn đến khả năng overfitting thấp hơn so với SVM.

Tài liệu tham khảo

"Mathematics for Machine Learning" by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong.
https://mml-book.github.io/

Bài 18: Duality - Machine Learning Cơ Bản
https://machinelearningcoban.com/2017/04/02/duality/

Bài 19: Support Vector Machine - Machine Learning Cơ Bản
https://machinelearningcoban.com/2017/04/09/smv/

Bài 20: Soft Margin Support Vector Machine - Machine Learning Cơ Bản
https://machinelearningcoban.com/2017/04/13/softmarginsmv/

Bài 21: Kernel Support Vector Machine - Machine Learning Cơ Bản
https://machinelearningcoban.com/2017/04/22/kernelsmv/

Bài 22: Multi-class Support Vector Machine - Machine Learning Cơ Bản
https://machinelearningcoban.com/2017/04/28/multiclasssmv/

Support Vector Machines — scikit-learn 1.3.2 documentation
https://scikit-learn.org/stable/modules/svm.html

RBF SVM parameters - scikit-learn documentation
https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Support Vector Machine - Cornell CS 4/5780
https://www.cs.cornell.edu/courses/cs4780/2022sp/notes/LectureNotes13.html

Lesson 5: Lagrange multipliers and constrained optimization - Khan Academy
https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/lagrange-multipliers-and-constrained-optimization/v/constrained-optimization-introduction

Lecture 14 - Support Vector Machines - Yaser Abu Mostafa - Caltech
Lecture 15 - Kernel Methods - Yaser Abu Mostafa - Caltech
Lecture 16 - Radial Basis Functions - Yaser Abu Mostafa - Caltech
https://www.youtube.com/playlist?list=PLnIDYuXHkit4LcWjDe0EwlE57WiGlBs08