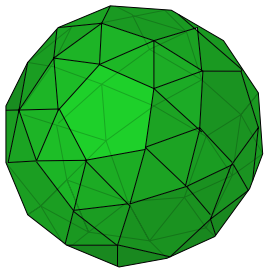
Projects in Mathematics and Applications

SUPPORT VECTOR MACHINE

Phạm Quốc Việt *, Nguyễn Hà Minh †
Nguyễn Đình Bảo Ngọc ‡, Nguyễn Huy Anh §

Ngày 27 tháng 8 năm 2018



* Trường THPT Chuyên Amsterdam
† Trường THPT Nguyễn Bỉnh Khiêm
‡ Trường Phổ Thông Năng Khiếu - ĐHQG TP.HCM
§ Campbell County Comprehensive High School

Lời cảm ơn

Chúng em xin chân thành cảm ơn Ban tổ chức PiMA vì đã mang đến một trại hè Toán học vô cùng thú vị và đầy ý nghĩa cho chúng em cũng như là các bạn trại sinh còn lại. Không những chúng em được tìm hiểu sâu và rộng hơn về Toán, chúng em còn được làm quen và học hỏi từ các bạn có cùng chung đam mê dành cho môn Toán từ khắp nơi trên mọi miền đất nước. Tham gia PiMA, chúng em được giới thiệu với các mảng Toán ở Đại học, được học lập trình, nghiên cứu và viết báo cáo, và trên hơn cả là rèn luyện kĩ năng làm việc nhóm. Chúng em rất trân trọng trải nghiệm 10 ngày vừa qua và hy vọng rằng PiMA sẽ tiếp tục truyền cảm hứng và nuôi giữ ngọn lửa đam mê cho các bạn học sinh tham gia như PiMA đã thực hiện thành công trong 3 năm vừa qua.

Chúng em cũng xin gửi lời cảm ơn đến Ban Giám Đốc Đại Học Khoa Học Tự Nhiên TP.HCM và các nhà tài trợ đã tạo điều kiện để trại hè PiMA 2018 được diễn ra thành công.

Chúng em xin cảm ơn đến các thầy diễn giả: GS.TS Hồ Tú Bảo, PGS.TS Đinh Điền, TS. Nguyễn Đức Hoàng Hạ, TS. Nguyễn Minh Triết, PGS.TS Lý Quốc Ngọc. Qua những lời chia sẻ nhiệt huyết từ các thầy, chúng em và các bạn trại sinh còn lại đã được mở mang và tiếp xúc với nhiều khía cạnh khác nhau trong môn Toán, đặc biệt là đối với mảng máy học.

Chúng em xin chân thành gửi lời cảm ơn đến sự hỗ trợ của anh Vũ Lê Thế Anh, là người đã trực tiếp hướng dẫn tụi em trong quá trình tìm hiểu và chuẩn bị bài báo cáo này, cùng sự góp ý của anh Dương Đình Trọng cho bài báo cáo này được hoàn hảo hơn.

Và xin cảm ơn đến các bạn trại sinh còn lại vì đã đồng hành cùng chúng mình trong 10 ngày vừa qua.

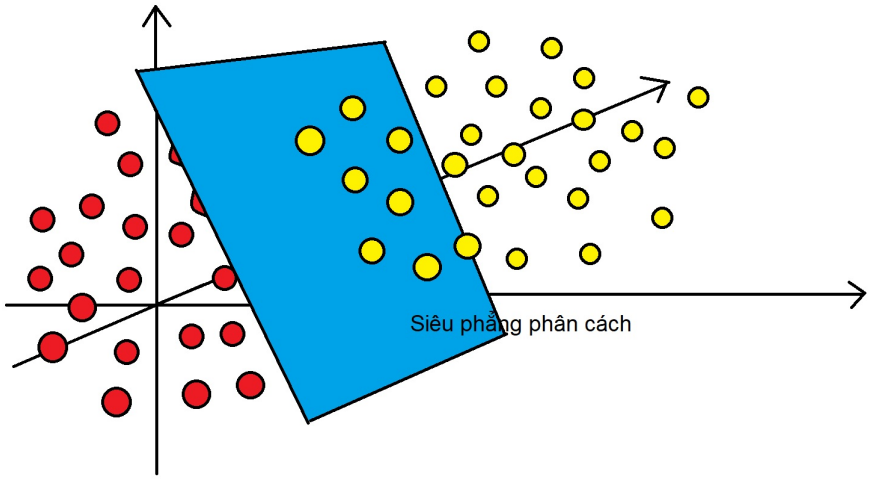
Tóm tắt nội dung

Trong khuôn khổ bài báo cáo này, chúng em sẽ trình bày về thuật toán Support Vector Machine và ứng dụng của nó trong các bài toán, đặc biệt là trong hệ thống phân lớp nhị phân. Bài báo cáo được xây dựng trên cơ sở toán học, Soft-margin và Hard-margin SVM, cách tối ưu hệ thống, và mô phỏng thuật toán này trên mô hình dự đoán giao thức tín dụng.

Mục lục

1 Tổng quan

Thuật toán Support Vector Machine chính thức được giới thiệu bởi Bernhard E. Boser, Isabelle M. Guyon và Vladimir N. Vapnik vào năm 1992. Thuật toán Support Vector Machine bắt đầu trở nên nổi tiếng khi được áp dụng vào kĩ thuật nhận dạng con số viết bằng tay. Ngày nay, Support Vector Machine có vai trò quan trọng trong máy học và được biết đến như là một ví dụ điển hình của phương pháp kernel.
Support Vector Machine giúp hỗ trợ giải quyết một trong những nhiệm vụ phổ biến của máy học: phân cách dữ liệu. Cụ thể hơn, Support Vector Machine giúp ta phân loại một điểm dữ liệu mới vào một trong hai lớp dữ liệu đã được hình thành từ những điểm dữ liệu cũ bằng cách gắn với mỗi điểm dữ liệu đó một vector p chiều và đi tìm một siêu phẳng $p-1$ chiều để phân cách các vector trên sao cho khoảng cách từ lề đến siêu phẳng đó là lớn nhất.
Hình sau là một ví dụ của siêu phẳng phân cách trong không gian ba chiều.



2 Một số định nghĩa và tính chất

Định nghĩa 2.1. Một tập hợp A được gọi là lồi nếu với hai phần tử x_i và x_j bất kì thuộc A, phần tử $x_k = \varepsilon x_i + (1-\varepsilon)x_j$ cũng thuộc $A, \forall \varepsilon \in [0,1]$.

Tính chất 2.2.

- Một siêu phẳng là một tập lồi.
- Giao của các tập lồi là một tập lồi.

Định nghĩa 2.3. Một hàm số $f : \mathbb{R}^n \to \mathbb{R}$ là hàm lồi nếu tập xác định (dom) D là một tập lồi và:

$$f(\varepsilon \boldsymbol{x}_i + (1-\varepsilon)\boldsymbol{x}_j) \leq \varepsilon f(\boldsymbol{x}_i) + (1-\varepsilon) f(\boldsymbol{x}_j), \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in D, 0 \leq \varepsilon \leq 1.$$

Tính chất 2.4. Hàm số $f : \mathbb{R}^n \to \mathbb{R}$ là một hàm lõm nếu $-f$ là một hàm lồi.

Tính chất 2.5. Hàm norm là một hàm lồi.

Định nghĩa 2.6. Một hàm số $f : \mathbb{R}^n \to \mathbb{R}$ là hàm lồi chặt nếu f là hàm lồi và dấu bằng không xảy ra $\forall \boldsymbol{x}_i \neq \boldsymbol{x}_j$.

Định nghĩa 2.7. Một bài toán tối ưu lồi là một bài toán tối ưu có dạng:

$$\boldsymbol{x} = \underset{x}{\operatorname{argmin}}\, f_0, \tag{1}$$

với

$$\begin{aligned} & f_i(\boldsymbol{x}) \leq 0 && \forall i = 1, 2, \ldots, n, \\ & h_j(\boldsymbol{x}) = \boldsymbol{a}_j^T \boldsymbol{x} - b_j = 0 && \forall j = 1, 2, \ldots, m, \\ & \boldsymbol{x} \in \mathcal{D} = (\bigcap_{i=0}^{n} \operatorname{dom} f_i) \cap (\bigcap_{j=1}^{n} \operatorname{dom} h_j). \end{aligned}$$

Trong đó, f_i là các hàm lồi $\forall i = 0, 1, ..., n$.

Định nghĩa 2.8. Gọi $\boldsymbol{x}_0$ là nghiệm toàn cục của bài toán tối ưu lồi (1) nếu:

$$\boldsymbol{x}_0 = \underset{x}{\operatorname{argmin}}\, f_0,$$

$$\text{và} \quad f_i(\boldsymbol{x}_0) \leq 0 \quad \forall i = 1, 2, ...n, \quad h_j(\boldsymbol{x}_0) = 0, \quad \forall j = 1, 2, \ldots, m.$$

Định nghĩa 2.9. Gọi $\boldsymbol{x}_0$ là nghiệm cục bộ lân cận ε của bài toán tối ưu lồi (1) nếu:

$$\boldsymbol{x}_0 = \underset{x}{\operatorname{argmin}}\, f_0,$$

$$\text{và} \quad f_i(\boldsymbol{x}_0) \leq 0 \quad \forall i = 1, 2, ...n, \quad h_j(\boldsymbol{x}_0) = 0 \quad \forall j = 1, ..., m, \ \|\boldsymbol{x} - \boldsymbol{x}_0\|_2 \leq \varepsilon.$$

Tính chất 2.10. Nghiệm cục bộ lân cận của một bài toán tối ưu lồi cũng chính là nghiệm toàn cục của bài toán đó.

Chứng minh bằng phản chứng: Giả sử $\boldsymbol{x}_0$ là một nghiệm cục bộ lân cận ε nhưng không là cực trị toàn cục. Gọi $\boldsymbol{y}_0$ là cực trị toàn cục, tức là $f(\boldsymbol{y}_0) \leq f(\boldsymbol{x}_0)$, $\|\boldsymbol{y}_0 - \boldsymbol{x}_0\|_2 > \varepsilon$.
Mặt khác, luôn tồn tại $\boldsymbol{z}_0$ sao cho:

$$\begin{cases} \boldsymbol{z}_0 = \theta \boldsymbol{y}_0 + (1 - \theta)\boldsymbol{x}_0 \text{ với } \theta \in [0, 1] \text{ đủ nhỏ}, \\ \|\boldsymbol{z}_0 - \boldsymbol{x}_0\|_2 < \varepsilon, \\ \boldsymbol{z}_0 \text{ thỏa mãn các hàm điều kiện ràng buộc.} \end{cases}$$

Khi đó:

$$\begin{aligned} f_0(\boldsymbol{z}_0) &= f_0(\theta \boldsymbol{y}_0 + (1 - \theta)\boldsymbol{x}_0) \\ &\leq \theta f_0(\boldsymbol{y}_0) + (1 - \theta) f_0(\boldsymbol{x}_0) \\ &< \theta f_0(\boldsymbol{x}_0) + (1 - \theta) f_0(\boldsymbol{x}_0) \\ &= f_0(\boldsymbol{x}_0). \end{aligned}$$

Điều này mâu thuẫn do $\boldsymbol{x}_0$ là nghiệm cục bộ lân cận ε.

Định nghĩa 2.11. Hàm Lagrangian của bài toán tối ưu lồi (1) là hàm:

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

với,

$$\begin{aligned} & \boldsymbol{x} \in \mathcal{D}, \\ & \boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_n]^T, \boldsymbol{\lambda} \succeq 0, \\ & \boldsymbol{\nu} = [\nu_1, \nu_2, ..., \nu_n]^T. \end{aligned}$$

Định nghĩa 2.12. Hàm đối ngẫu của bài toán tối ưu lồi (1) với $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ là hàm:

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}).$$

Nhận xét 2.13. *Hàm đối ngẫu trên là một hàm lõm.*

Tính chất 2.14. $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\boldsymbol{x}_0)$ với $\boldsymbol{x}_0$ là nghiệm toàn cục của bài toán tối ưu lồi (1).

Định nghĩa 2.15. Bài toán đối ngẫu Lagrange của bài toán (1) là:

$$(\boldsymbol{\lambda}_0, \boldsymbol{\nu}_0) = \underset{\boldsymbol{\lambda}, \boldsymbol{\nu}}{\operatorname{argmax}}\, g(\boldsymbol{\lambda}, \boldsymbol{\nu}).$$

Định nghĩa 2.16. Đối ngẫu chặt xảy ra khi $g(\boldsymbol{\lambda}_0, \boldsymbol{\nu}_0) = f_0(\boldsymbol{x}_0)$.

Nhận xét 2.17. *Khi đối ngẫu chặt xảy ra, nghiệm của bài toán đối ngẫu chính là nghiệm của bài toán tối ưu gốc.*

Định lý 2.18 (Tiêu chuẩn Slater)**.** *Khi bài toán tối ưu gốc là lồi, và tồn tại $\boldsymbol{x}^* \in \mathcal{D}$ sao cho: $f_i(\boldsymbol{x}^*) < 0 \quad \forall i = 1, 2, ..., n, \quad h_j(\boldsymbol{x}^*) = 0 \quad \forall j = 1, 2, ..., m$ thì đối ngẫu chặt xảy ra.*

Định lý 2.19 (Hệ điều kiện KKT)**.** *Điều kiện cần và đủ KKT cho nghiệm $(\boldsymbol{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\nu}_0)$ bài toán đối ngẫu là:*

$$\begin{aligned}
f_i(\boldsymbol{x}_0) &\leq 0, \forall i = 1, 2, ..., n, \\
h_j(\boldsymbol{x}_0) &= 0, \forall j = 1, 2, ..., m, \\
\lambda_i^0 &\geq 0, \forall i = 1, 2, ..., n, \\
\lambda_i^0 f_i(\boldsymbol{x}_0) &= 0, \forall i = 1, 2, ..., n, \\
\frac{\partial f_0}{\partial \boldsymbol{x}_0} + \sum_{i=1}^{n} \lambda_i^0 \frac{\partial f_i}{\partial \boldsymbol{x}_0} + \sum_{j=1}^{m} v_j^0 \frac{\partial h_j}{\partial \boldsymbol{x}_0} &= 0.
\end{aligned}$$

3 Phát biểu bài toán

Xét tập dữ liệu $D = \{(\boldsymbol{x_i}, \boldsymbol{y_i}); \boldsymbol{x_i} \in \mathbb{R}^m; y_i \in \{1, -1\} \mid i = 1, 2, ..., n\}$ với n là số cặp dữ liệu và m là số chiều của dữ liệu (số lượng đặc trưng). Trong đó:

$$y_i = \begin{cases} 1 & \text{nếu cặp dữ liệu thứ } i \text{ thuộc nhóm 1,} \\ -1 & \text{nếu cặp dữ liệu thứ } i \text{ thuộc nhóm 2.} \end{cases}$$

Ta cần xác định siêu phẳng $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ phân chia 2 nhóm sao cho lề là lớn nhất, với định nghĩa lề trong bài toán SVM là giá trị của:

$$\min_{i=1,2,...n} \frac{|\boldsymbol{w}^T\boldsymbol{x_i} + b|}{\|\boldsymbol{w}\|_2}$$

Bài toán trên tương đương với bài toán tối ưu sau:

$$(\boldsymbol{w}, b) = \underset{\boldsymbol{w}, b}{\operatorname{argmax}} \left\{ \min_i \frac{|\boldsymbol{w}^T\boldsymbol{x_i} + b|}{\|\boldsymbol{w}\|_2} \right\} \tag{2}$$

Nhận xét 3.1. *Ta luôn chọn được $(\boldsymbol{w}, b)$ sao cho y_i và $(\boldsymbol{w}^T\boldsymbol{x_i} + b)$ cùng dấu vì nếu không chỉ cần đổi dấu của $y_i, \forall i = 1, 2, ..., n$.*

Khi đó, (2) tương đương với tìm:

$$(\boldsymbol{w}, b) = \underset{\boldsymbol{w}, b}{\operatorname{argmax}} \left\{ \min_i \frac{y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)}{\|\boldsymbol{w}\|_2} \right\} \tag{3}$$

4 Các phương pháp giải bài toán SVM

4.1 Tập dữ liệu có thể phân cách tuyến tính: Hard-margin SVM

4.1.1 Bài toán tương đương

Ta có: luôn tồn tại siêu phẳng $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ phân chia hoàn toàn hai nhóm. Gọi $(\boldsymbol{x}_0, y_0)$ là:

$$(\boldsymbol{x}_0, y_0) = \underset{(\boldsymbol{x}_i, y_i)}{\operatorname{argmin}} \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_2}$$

Nhận xét 4.1. *Nếu thay* $(\boldsymbol{w}, b) = (k\boldsymbol{w}, kb)$ *thì* $\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_2}$ *không đổi,* $\forall i = 1, 2, ..., n$. *Từ đó, không mất tính tổng quát, giả sử* $y_0(\boldsymbol{w}^T\boldsymbol{x}_0 + b) = 1$.

Khi đó (3) trở thành tìm:

$$(\boldsymbol{w}, b) = \underset{\boldsymbol{w}, b}{\operatorname{argmax}} \frac{1}{\|\boldsymbol{w}\|_2}, \tag{4}$$

$$\text{với } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq y_0(\boldsymbol{w}^T\boldsymbol{x}_0 + b) = 1, \ \forall i = 1, 2, ..., n.$$

(4) tương đương với tìm:

$$(\boldsymbol{w}, b) = \underset{\boldsymbol{w}, b}{\operatorname{argmax}} \frac{1}{2}\|\boldsymbol{w}\|_2^2, \tag{5}$$

$$\text{với } 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \leq 0, \ \forall i = 1, 2, ..., n.$$

Như vậy, sau khi xác định được siêu phẳng $\boldsymbol{w}^T\boldsymbol{x} + b = 0$, nhóm của $\boldsymbol{x}_i \in \mathbb{R}^m$ bất kì trùng với dấu của $\boldsymbol{w}^T\boldsymbol{x}_i + b = 0$.

4.1.2 Bài toán đối ngẫu Lagrange

Kiểm tra tiêu chuẩn Slater: Vì hai lớp dữ liệu là có thể phân cách tuyến tính, tức bài toán có nghiệm, nên luôn tồn tại cặp $(\boldsymbol{w}_o, b_o)$ sao cho:

$$\begin{aligned} 1 - y_i(\boldsymbol{w}_o^T\boldsymbol{x}_i + b_o) &\leq 0, \forall i = 1, 2, ..., n, \\ \Leftrightarrow 2 - y_i(2\boldsymbol{w}_o^T\boldsymbol{x}_i + 2b_o) &\leq 0, \forall i = 1, 2, ..., n. \end{aligned}$$

Chọn $\boldsymbol{w}_1 = 2\boldsymbol{w}_o$ và $b_1 = 2b_o$, suy ra:

$$1 - y_i(\boldsymbol{w}_1^T\boldsymbol{x}_i + b_1) \leq -1 < 0, \forall i = 1, 2, ..., n$$

Vậy tiêu chuẩn Slater thỏa mãn.

Hàm Lagrangian của bài toán (5) là:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} \lambda_i(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

với $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_n]^T$ và $\lambda_i \geq 0, \forall i = 1, 2, ..., n$.

Hàm đối ngẫu Lagrange là

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) \text{ với } \boldsymbol{\lambda} \succeq 0.$$

Xét $(\boldsymbol{w}_0, b_0, \boldsymbol{\lambda}_0) = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda})$. Khi đó:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_0} = \boldsymbol{w}_0 - \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i = 0 \Rightarrow \boldsymbol{w}_0 = \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i.$$
$$\frac{\partial \mathcal{L}}{\partial b_0} = \sum_{i=1}^{n} \lambda_i y_i = 0.$$

Thay vào biểu thức của $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda})$, ta được:

$$g(\boldsymbol{\lambda}) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \boldsymbol{x}_n^T \boldsymbol{x}_m \tag{6}$$

Từ đó, kết hợp hàm đối ngẫu Lagrange và các điều kiện ràng buộc của $\boldsymbol{\lambda}$, ta sẽ thu được **bài toán đối ngẫu** Lagrange của bài toán (26.3) có dạng

$$\boldsymbol{\lambda} = arg \max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda})$$

với:

$$\boldsymbol{\lambda} \succeq 0$$
$$\sum_{n=1}^{N} \lambda_n y_n = 0$$

Vì đây là một bài toán lồi và đối ngẫu chặt xảy ra, **hệ điều kiện KKT** là điều kiện cần và đủ cho bộ $(\boldsymbol{w}_0, b_0, \boldsymbol{\lambda}_0)$:

$$\begin{aligned}
1 - y_i(\boldsymbol{w_0}^T \boldsymbol{x}_i + b_0) &\leq 0, \forall i = 1, 2, ..., n \\
\lambda_i &\geq 0, \forall i = 1, 2, ..., n \\
\lambda_i(1 - y_i(\boldsymbol{w_0}^T \boldsymbol{x_i} + b_0)) &= 0 \ \forall i = 1, 2, ..., n \qquad (7) \\
\boldsymbol{w}_0 &= \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x_i} \\
\sum_{n=1}^{N} \lambda_n y_n &= 0 \qquad (8)
\end{aligned}$$

Từ điều kiện (7) ta thấy hoặc $\lambda_i = 0$ hoặc $y_i(\boldsymbol{w_0}^T \boldsymbol{x_i} + b_0) = 1$. Khi đó, hai đường thẳng $\boldsymbol{w_0}^T \boldsymbol{x}_i + b = \pm 1$ chia tách hoàn toàn hai lớp dữ liệu.
Từ đây ta có định nghĩa sau:

Định nghĩa 4.2. Hai đường biên của bài toán SVM là hai đường thẳng $\boldsymbol{w_0}^T \boldsymbol{x}_i + b = \pm 1$.

Nhận xét 4.3. *Số điểm nằm trên cả hai siêu phẳng là nhỏ hơn rất nhiều so với tổng sô điểm của tập huấn luyện.*

Đặt $S = \{n : \lambda_n \neq 0\}$ và N_S là số phần tử của tập S
Theo (8), ta tính $\boldsymbol{w}$:

$$\begin{aligned} \boldsymbol{w} &= \sum_{m \in S} \lambda_m y_m \boldsymbol{x}_m, \\ b &= \frac{1}{N_S} \sum_{n \in S} \left(y_n - \sum_{m \in S} \lambda_m y_m \boldsymbol{x}_m^T \boldsymbol{x}_n \right) \end{aligned}$$

Về việc tính $\boldsymbol{\lambda}_0$ thế nào, mô hình sẽ được dựng bằng ngôn ngữ lập trình Python.

4.2 Trường hợp tập dữ liệu không thể phân cách tuyến tính: Soft-margin SVM và Kernel Trick

4.2.1 Bài toán tương đương: Soft-margin SVM

Trong thực tế, tập dữ liệu huấn luyện ta có gần như không thể *phân cách tuyến tính*. Khi đó, nếu dùng thuật toán Hard-margin SVM sẽ dẫn đến việc kết quả *lề* rất nhỏ, thậm chí không tìm được siêu phẳng phân cách.
Với suy nghĩ như vậy, thuật toán *Soft-margin SVM* ra đời. Lúc này, để thu được *lề* lớn hơn, ta chấp nhận một vài điểm dữ liệu bị *sai* rơi vào bên trong hoặc sang hẳn phía bên kia của *lề*. Dễ thấy sự chấp nhận *sai* cần được hạn chế, nếu không, ta có thể tạo ra một *lề* rất lớn bằng cách chấp nhận hầu hết các điểm bị *sai*.
Từ đó, hàm mục tiêu nên là một sự kết hợp để tối đa *lề* cũng như tối thiểu sự *sai*.
Như đã nói trong *hard-margin SVM*, việc tối đa *lề* có thể đưa về việc tối thiểu $|\boldsymbol{w}\|_2^2$. Để tính sự *sai*, định nghĩa thêm một biến đo độ *sai* với mỗi điểm $\boldsymbol{x}_i$ trong tập dữ liệu huấn luyện.

Định nghĩa 4.4. α_i là biến *sai* của biến x_i nếu như:

$$\alpha_i = \begin{cases} 0 & \text{nếu điểm dữ liệu } x_i \text{ nằm trong biên của nó,} \\ |\boldsymbol{w}^T \boldsymbol{x}_i + b - y_i| & \text{nếu điểm dữ liệu } x_i \text{ không nằm trong biên của nó.} \end{cases}$$

Nhận xét 4.5. *Ta có thể định nghĩa biến sai như trên do α_i chỉ cần tỉ lệ với khoảng cách từ x_i đến siêu phẳng*

Với *soft-margin SVM*, hàm mục tiêu sẽ có thêm một số hạng nữa giúp tối thiểu tổng độ *sai*. Khi đó, hàm mục tiêu mới là:

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_{n=1}^{N} \alpha_n,$$

với C là một hằng số dương.
Nhận thấy $|y_i| = 1, \forall i = 1, 2, ..., n$, ta có:

$$\begin{aligned} \alpha_i &= \left| \boldsymbol{w}^T \boldsymbol{x}_i + b - y_i \right| \\ &= \left| \boldsymbol{w}^T \boldsymbol{x}_i + b - y_i \right| |y_i| \\ &= \left| y_i (\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \right| \\ &\geq 1 - y_i (\boldsymbol{w}^T \boldsymbol{x}_i + b) \end{aligned}$$

Từ đó ta thu được điều kiện ràng buộc mới sau, $\forall i = 1, 2, ..., n$:

$$\begin{aligned} & 1 - \alpha_i - y_i (\boldsymbol{w}^T \boldsymbol{x}_i + b) \leq 0, \\ \text{và} \quad & \alpha_i \geq 0. \end{aligned}$$

Như vậy, bài toán tối ưu gốc cho *Soft-margin SVM* phát biểu như sau, $\forall i = 1, 2, ..., n$:

$$\begin{aligned}(\boldsymbol{w}_0, b_0, \alpha_0) = \underset{\boldsymbol{w},b,\alpha}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\alpha_i,\\ \text{với: } 1 - \alpha_i - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \leq 0,\\ -\alpha_n \leq 0.\end{aligned} \tag{9}$$

4.2.2 Hướng giải quyết thứ nhất: Bài toán đối ngẫu Lagrange của Soft-margin SVM

Tương tự bài toán Hard-margin SVM, ta quan tâm đến bài toán đối ngẫu của nó.
Kiểm tra tiêu chuẩn Slater: $\forall i = 1, 2, ..., n$ và $\forall(\boldsymbol{w}, b)$, luôn tồn tại các số dương α_i, đủ lớn sao cho:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \alpha_i > 1.$$

Vì vậy, bài toán tối ưu này thỏa mãn tiêu chuẩn Slater.
Hàm Lagrangian của bài toán Soft-margin SVM là:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}\lambda_i(1 - \alpha_i - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) - \sum_{i=1}^{n}\nu_i\alpha_i,\\ \text{với } \boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_n]^T \succeq 0 \text{ và } \boldsymbol{\nu} = [\nu_1, \nu_2, ..., \nu_n]^T \succeq 0.\end{aligned} \tag{10}$$

Bài toán đối ngẫu của Soft-margin SVM là:

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \min_{\boldsymbol{w},b,\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\nu})$$

Để giải bài toán trên, với mỗi cặp $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, xét bộ $(\boldsymbol{w}_0, b, \boldsymbol{\alpha}_0)$ thỏa mãn:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_0} = 0 \Leftrightarrow \boldsymbol{w}_0 = \sum_{i=1}^{n}\lambda_i y_i \boldsymbol{x}_i \tag{11}$$

$$\frac{\partial\mathcal{L}}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{n}\lambda_i y_i = 0 \tag{12}$$

$$\frac{\partial\mathcal{L}}{\partial\alpha_0} = 0 \Leftrightarrow \lambda_i = C - \nu_i \ \forall i = 1, 2, ..., n \tag{13}$$

Từ (13) suy ra chỉ cần quan tâm tới các cặp $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ thỏa mãn $\lambda_i = C - \nu_i$.
Từ đây ta cũng suy ra $0 \leq \lambda_i, \nu_i \leq C, \forall i = 1, 2, ..., n$.
Thay vào biểu thức Lagrangian, ta thu được hàm mục tiêu của bài toán đối ngẫu:

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j. \tag{14}$$

Nhận xét 4.6. *Hàm này không phụ thuộc vào* $\boldsymbol{\nu}$ *và từ* (13) *suy ra* $0 \leq \lambda_i \leq C, \forall i = 1, 2, ..., n$

Khi đó, bài toán đối ngẫu trở thành tìm:

$$\boldsymbol{\lambda}_0 = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\, g(\boldsymbol{\lambda})$$

$$\text{với: } \sum_{i=1}^{n}\lambda_i y_i = 0, \tag{15}$$

$$0 \leq \lambda_i \leq C, \forall i = 1, 2, ..., n. \tag{16}$$

Hệ điều kiện KKT của bài toán tối ưu *Soft-margin SVM* là, $\forall i = 1, 2, ..., n$:

$$1 - \alpha_i^0 - y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0) \leq 0, \tag{17}$$
$$-\alpha_i^0 \leq 0, \tag{18}$$
$$\lambda_i^0 \geq 0, \tag{19}$$
$$\nu_i^0 \geq 0, \tag{20}$$
$$\lambda_i^0(1 - \alpha_i^0 - y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0)) = 0, \tag{21}$$
$$\nu_i^0 \alpha_i^0 = 0, \tag{22}$$

Ngoài ra còn có các điều kiện (11), (12), (13).
Khi $\lambda_i^0 > 0$, theo (21):

$$y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0) = 1 - \alpha_i^0. \tag{23}$$

Nếu $0 < \lambda_i^0 < C$, theo (13), $\nu_i^0 = C - \lambda_i^0 > 0$; ngoài ra theo (22) ta thu được $\alpha_i^0 = 0$. Tiếp tục theo (23), suy ra $y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0) = 1$, hay:

$$\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0 = y_i, \forall i : 0 < \lambda_i^0 < C.$$

Tóm lại, nếu $0 < \lambda_i^0 < C$, các điểm $\boldsymbol{x}_i$ nằm chính xác trên các lề. Khi đó, $(\boldsymbol{w}_0, b_0)$ được tính theo:

$$\begin{aligned} \boldsymbol{w}_0 &= \sum_{i=1}^{n} \lambda_i^0 y_i \boldsymbol{x}_i, \\ b_0 &= \frac{1}{N_{\mathcal{S}}} \sum_{i \in \mathcal{S}} (y_i - \boldsymbol{w}_0^T \boldsymbol{x}_i) \end{aligned} \tag{24}$$

với $\mathcal{S} = \{m : 0 < \lambda_m < C\}$ và $N_{\mathcal{S}}$ là số phần tử của $\mathcal{S}$.
Khi đó, nhóm của $\boldsymbol{x}^*$ chính là dấu của:

$$\boldsymbol{w}_0^T \boldsymbol{x}^* + b_0 = \sum_{i \in M} \lambda_i^0 y_i \boldsymbol{x}_i^T \boldsymbol{x}^* + \frac{1}{N_{\mathcal{S}}} \sum_{j \in \mathcal{S}} \left(y_j - \sum_{i \in M} \lambda_j^0 y_j \boldsymbol{x}_j^T \boldsymbol{x}^* \right) \tag{25}$$

Với $\mathcal{M} = \{n : 0 < \lambda_n\}$ và $N_{\mathcal{M}}$ là số phần tử của $\mathcal{M}$.

4.2.3 Hướng giải quyết thứ 2: Bài toán tối ưu không ràng buộc cho soft-margin SVM

Bài toán tối ưu không ràng buộc tương đương: Xét điều kiện ràng buộc đầu tiên:

$$1 - \alpha_i - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \leq 0 \Leftrightarrow \alpha_i \geq 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

Mặt khác, $\alpha_i \geq 0$ nên ta có bài toán ràng buộc tương đương với bài toán (9) như sau:

$$\begin{aligned} (\boldsymbol{w}_0, b_0, \alpha_0) = & arg \min_{\boldsymbol{w}, b, \alpha} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{i=1}^{n} \alpha_i \\ & \text{thỏa mãn: } \alpha_i \geq \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)), \forall i = 1, 2, ..., n \end{aligned} \tag{26}$$

Mệnh đề 4.7. Nếu $(\boldsymbol{w}_0, b_0, \alpha_0)$ là nghiệm của bài toán tối ưu (26), thì:

$$\alpha_i^0 = \max(0, 1 - y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0)), \forall i = 1, 2, ..., n \tag{27}$$

Chứng minh: Giả sử tồn tại i sao cho

$$\alpha_i > \max(0, 1 - y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0)),$$

Khi đó, chọn $\alpha_i' = \max(0, 1 - y_i(\boldsymbol{w}_0^T \boldsymbol{x}_i + b_0))$, ta được một giá trị nhỏ hơn của hàm mục tiêu, trong khi tất cả các ràng buộc vẫn được thỏa mãn (mâu thuẫn).
Vậy mệnh đề được chứng minh.
Từ đó, bài toán tối ưu trở thành:

$$\begin{aligned}(\boldsymbol{w}_0, b_0, \alpha_0) = arg \min_{\boldsymbol{w},b,\alpha} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) \\ \text{thỏa mãn: } \alpha_i = \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)), \forall i = 1, 2, ..., n,\end{aligned} \tag{28}$$

hay:

$$(\boldsymbol{w}, b) = arg \min_{\boldsymbol{w},b} \left\{ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) \right\} \tag{29}$$

Nhận xét 4.8. *Bài toán này có thể giải được bằng các phương pháp gradient descent.*

Định nghĩa 4.9. Hàm Hinge loss có dạng:

$$J_i(\boldsymbol{w}, b) = \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))$$

Xây dựng hàm mất mát

Định nghĩa 4.10. Hàm Hinge loss trung bình có công thức:

$$J(\boldsymbol{w}, b) = \frac{1}{n}\sum_{i=1}^{n} J_i = \frac{1}{n}\sum_{i=1}^{n} \ max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))$$

Nhận xét 4.11. *Xét $(\boldsymbol{w}_0, b_0)$ là nghiệm bài toán Khi tập dữ liệu có thể phân cách tuyến tính, giá trị tối ưu của $J(\boldsymbol{w}, b) = 0$. Khi đó dễ thấy $\forall c > 1$ là hằng số, $(c\boldsymbol{w}_0, cb_0)$ cũng là nghiệm bài toán dẫn tới nghiệm có thể lớn tùy ý.*

Vì vậy, ta có định nghĩa sau:

Định nghĩa 4.12. Hàm mất mát tổng cộng có công thức là:

$$J(\boldsymbol{w}, b) = L(\boldsymbol{w}, b) + \beta R(\boldsymbol{w}, b),$$

với $\beta > 0$, $R(\boldsymbol{w}, b) = \frac{1}{n}\frac{\beta}{2}\|\boldsymbol{w}\|_2^2$.

Nhận xét 4.13. *Hàm mất mát tổng cộng là một hàm lồi theo $(\boldsymbol{w}, b)$.*

Tối ưu hàm mất mát: Vì việc tối ưu hàm mất mát dựa trên gradient descent, việc chính của mục này là tính đạo hàm của hàm mất mát theo $\boldsymbol{w}$ và b.

Nhận xét 4.14. *Đạo hàm của hàm mất mát tổng cộng không quá phức tạp.*
Vì thế ta dùng Gradient Descent theo $\boldsymbol{w}$ và b để tính giá trị tối ưu hàm mất mát.

Phần tính toán cụ thể, sẽ đề cập trong code Python. Đạo hàm của phần *hinge loss* không quá phức tạp:

$$\frac{\partial(\max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)))}{\partial \boldsymbol{w}} = \begin{cases} -y_i x_i & \text{nếu } 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 0 \\ 0 & \text{nếu } 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0 \end{cases}$$

$$\frac{\partial(\max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)))}{\partial b} = \begin{cases} -y_i & \text{nếu } 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 0 \\ 0 & \text{nếu } 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0 \end{cases}$$

Phần *regularization* cũng có đạo hàm tương đối đơn giản:

$$\frac{\partial\left(\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2\right)}{\partial \boldsymbol{w}} = \lambda \boldsymbol{w}$$

$$\frac{\partial\left(\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2\right)}{\partial b} = 0$$

Nếu cập nhật bằng gradient descent thông qua chỉ một điểm dữ liệu $(\boldsymbol{x}_i, y_i)$ (*stochastic gradient descent*). Nếu $1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0$, ta không cập nhật gì và chuyển sang điểm tiếp theo. Ngược lại biểu thức cập nhật cho $\boldsymbol{w}$, b được cho bởi

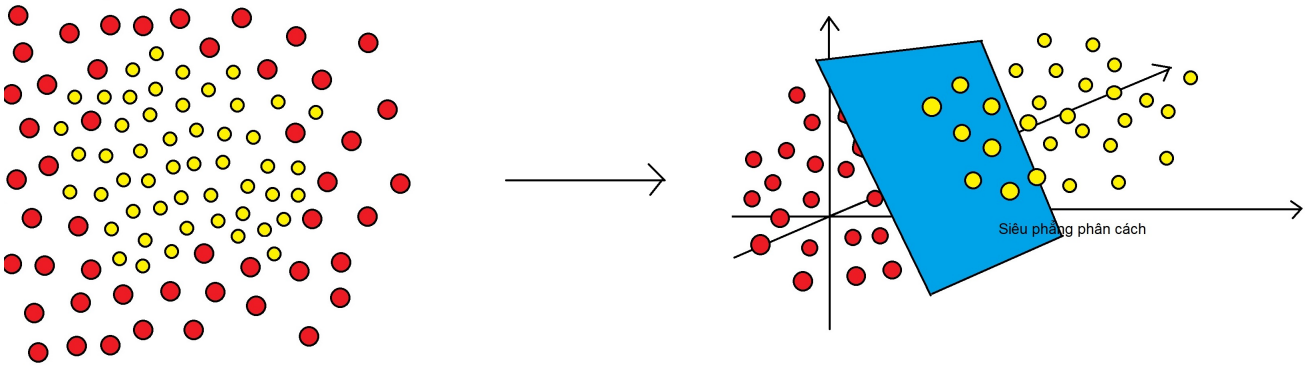
$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta(-y_i x_i + \lambda \boldsymbol{w})$; $b \leftarrow b + \eta y_i$ nếu $1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 0$

$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\lambda \boldsymbol{w}$; $b \leftarrow b$ nếu $1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0$

với η là *learning rate*.

4.2.4 Kernel trick

Để áp dụng thuật toán Support Vector Machine lên các dữ liệu không phân biệt tuyến tính, người ta cần biến các điểm dữ liệu này lên không gian nhiều chiều sao cho nó phân biệt tuyến tính hoặc gần như phân biệt tuyến tính trên không gian đó. Ở không gian này, bài toán có thể được giải quyết bằng hard/soft-margin SVM. Dựa trên suy nghĩ đó, ta đi tìm một hàm số $\phi(x)$ nhằm biến đổi dữ liệu $\boldsymbol{x}$ để thỏa các tính chất trên. Nhằm tăng hiệu năng tính toán và tiết kiệm bộ nhớ, các hàm Kernel được sử dụng để mô tả mối quan hệ giữa hai điểm dữ liệu bất kì trong không gian mới, thay vì tính toán trực tiếp biến đổi của chúng.
Ví dụ:



Ý tưởng cho hàm Kernel Xét hàm số $\phi()$ sao cho sau khi được biến đổi sang không gian mới, mỗi $\boldsymbol{x}$ trở thành $\phi(x)$, và khi đó tập dữ liệu mới trở nên gần phân biệt tuyến tính. Khi đó bài toán đối ngẫu của Soft-margin SVM trở thành:

$$\begin{aligned} \boldsymbol{\lambda} &= arg\max_{\boldsymbol{\lambda}} \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j), \\ \text{thỏa mãn: } & \sum_{i=1}^{n} \lambda_i y_i = 0, \\ & 0 \le \lambda_i \le C, \forall i = 1, 2, ..., n, \end{aligned} \tag{30}$$

và nhóm của một điểm dữ liệu mới $\boldsymbol{x}^*$ chính là dấu của biểu thức:

$$\boldsymbol{w}^T\phi(\boldsymbol{x}^*) + b = \sum_{j\in S} \lambda_j y_j \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}^*) + \frac{1}{N_{\mathcal{M}}} \sum_{i\in\mathcal{M}} \left(y_i - \sum_{j\in S} \lambda_j y_j \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_i) \right). \tag{31}$$

Nhận xét 4.15. *Trong (30) và (31), ta chỉ cần tính $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ với hai điểm dữ liệu $\boldsymbol{x}_i, \boldsymbol{x}_j$ bất kì. Vì thế ta chỉ cần xác định một hàm $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$.*

Khi đó, (30) và (31) trở thành:

$$\begin{aligned} \boldsymbol{\lambda} &= arg\max_{\boldsymbol{\lambda}} \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{thỏa mãn: } & \sum_{i=1}^{n} \lambda_i y_i = 0, \\ & 0 \le \lambda_i \le C, \forall i = 1, 2, ..., n, \end{aligned} \tag{32}$$

và nhóm của một điểm dữ liệu mới $\boldsymbol{x}^*$ chính là dấu của biểu thức:

$$\boldsymbol{w}^T\phi(\boldsymbol{x}^*) + b = \sum_{j\in S} \lambda_j y_j k(\boldsymbol{x}_i, \boldsymbol{x}^*) + \frac{1}{N_{\mathcal{M}}} \sum_{i\in\mathcal{M}} \left(y_i - \sum_{j\in S} \lambda_j y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right). \tag{33}$$

Tính chất của hàm Kernel:
+ Tính chất đối xứng.
+ Thỏa mãn điều kiện Mercer:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{j}_n) c_i c_j \ge 0, \forall c_k \in \mathbb{R}, k = 1, 2, ..., n. \tag{34}$$

Nhận xét 4.16. *Điều kiện Mercer giúp đảm bảo bài toán đối ngẫu (33) là lồi.*

Một số hàm Kernel thông dụng

- Hàm linear (tuyến tính): $k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^T\boldsymbol{z}$.
- Hàm RBF (radial basic function/Gaussian kernel): $k(\boldsymbol{x}, \boldsymbol{z}) = exp(-\gamma\|\boldsymbol{x} - \boldsymbol{z}\|_2^2), \gamma > 0$
- Hàm sigmoid: $k(\boldsymbol{x}, \boldsymbol{z}) = tanh(\gamma\boldsymbol{x}^T\boldsymbol{z} + r)$
- Hàm polynomial (đa thức): $k(\boldsymbol{x}, \boldsymbol{z}) = (r + \gamma\boldsymbol{x}^T\boldsymbol{z})^d$

5 Áp dụng vào mô hình thực tế

Bài toán Support Vector Machine trên có thể áp dụng vào mô hình dự đoán lừa đảo qua thẻ tín dụng. Ta sẽ áp dụng thuật toán phân lớp nhị phân vào mô hình này. Mục đích của thuật toán là cho biết giao dịch là lừa đảo (**positive class**) hay tin cậy (**negative class**).
Phần code chi tiết thuật toán bằng Python được đề cập ở mục Phụ lục.

5.1 Mô tả dữ liệu

Dữ liệu đầu vào gồm các giao dịch qua thẻ tín dụng thu thập vào 09/2013 tại Châu Âu bao gồm:

- Có 492 giao dịch lừa đảo trong tổng số 284,807 (chiếm 0.172%).
- 28 đặc trưng ($\vec{v} = \vec{v}_1, \vec{v}_2, ..., \vec{v}_{28}$) đã được giảm chiều dữ liệu qua phương thức phân tích thành phần chính (Principal Component Analysis).
- 2 đặc trưng mô tả thời gian và số tiền chuyển giữa hai giao dịch kế nhau.
- Lớp của dữ liệu: là lừa đảo hay tin cậy.

Do dữ liệu về giao dịch lừa đảo quá ít so với tổng số dữ liệu, nên việc dự đoán các giao dịch lừa đảo không được chú trọng. Để giải quyết vấn đề này, ta chỉ giữ lại một số dữ liệu của các giao thức tin cậy sao cho tập dữ liệu cân bằng. Kĩ thuật này gọi là **Undersampling**.
Việc mô phỏng thuật toán SVM trên tập dữ liệu này được thực hiện trên môi trường Python với sự hỗ trợ của thư viện **scikit-learn**.

5.2 Các thuật ngữ quan trọng

Để đánh giá một hệ thống phân lớp, người ta thường phân tập dữ liệu thành 3 phần chính:

- Tập **training**: dùng để huấn luyện mô hình.
- Tập **validation**: tối ưu các tham số khác cho mô hình, sử dụng $\boldsymbol{w}$ được huấn luyện từ tập **training**.
- Tập **test**: dùng để đánh giá thuật toán sau khi đã chọn ra tham số mong muốn .

Việc này nhằm mục đích chọn các tham số tối ưu và đánh giá hệ thống trên một tập dữ liệu chưa thấy (**unseen data**).

Định nghĩa 5.1. Độ chính xác là tỉ lệ giữa số điểm dữ liệu đoán đúng trên tổng số dữ liệu.

Nhận xét 5.2. *Ta thấy rằng với các bộ dữ liệu phân lớp không cân bằng (dữ liệu chẩn đoán có bệnh, xác định phần tử khủng bố), độ chính xác không là thang đo phù hợp để đánh giá hệ thống (do sự thiếu hụt dữ liệu dẫn đến kết quả sai).*

Để khắc phục điều này, người ta đưa ra hai thang đo sau.

Định nghĩa 5.3. Precision là tỉ lệ số điểm được dự đoán đúng **positive class** trên tổng số các điểm được dự đoán **positive class**.

Định nghĩa 5.4. Recall là tỉ lệ số điểm được dự đoán đúng **positive class** trên tổng số các điểm thật sự thuộc **positive class**.

Nhận xét 5.5. *Precision cao đồng nghĩa với việc độ chính xác của các điểm được dự đoán* **positive class** *là cao. Recall cao đồng nghĩa với việc tỉ lệ bỏ sót các điểm thực sự thuộc* **positive class** *là thấp.*

5.3 Tham số C và kernel trong thuật toán SVM

Tham số C là tham số quyết định việc tổng quát hóa của thuật toán với dữ liệu. Ở góc nhìn khác, tham số C xác định việc xem xét các dữ liệu nhiễu gần siêu phẳng phân cách của thuật toán có đáng kể hay không.

Nhận xét 5.6. *Nếu C có giá trị đủ lớn, thuật toán Soft-margin SVM trở thành Hard-margin SVM.*

Thư viện **scikit-learn** cho phép ta cung cấp cụ thể các kernel khác nhau nhằm mục đích chọn ra kernel thích hợp để tối ưu hóa (ở bài toán này là ưu tiên tối đa giá trị recall để dự đoán không bỏ sót các giao dịch lừa đảo).
Quá trình chọn hai tham số trên được thực hiện lần lượt như sau:

- Thực hiện thuật toán trên tập **training** để chọn ra tham số $\boldsymbol{w}$.
- Lần lượt thử $\boldsymbol{w}$ với các tham số C và kernel khác nhau trên tập **validation** và chọn bộ tham số tối ưu nhất.

Ở mô hình này ta chọn $C \in \{0.01, 1, 100\}$ và kernel $\in$ {'linear','rbf','sigmoid'}. Các kết quả đánh giá với các tham số khác nhau:

Kernel = 'linear'	0.01	1	100
A.Precision	0.983	0.985	0.981
A.Accuracy	0.898	0.898	0.901
A.Recall	0.81	0.808	0.819
Standard deviation of recall	0.0475	0.0413	0.049

Kernel = 'rbf'	0.01	1	100
A.Precision	0.326	0.61	0.607
A.Accuracy	0.483	0.598	0.606
A.Recall	0.666	0.577	0.625
Standard deviation of recall	0.471	0.0623	0.0327

Kernel = 'sigmoid'	0.01	1	100
A.Precision	0.326	0.493	0.462
A.Accuracy	0.483	0.483	0.488
A.Recall	0.666	0.666	0.67
Standard deviation of recall	0.471	0.468	0.465

Sau quá trình thử ta chọn được C = 100 với kernel là hàm tuyến tính (linear).

Nhận xét 5.7. *Ta thấy rằng việc dùng Hard-margin SVM là hợp lí, vì việc dự đoán sót các giao dịch lừa đảo nghiêm trọng hơn việc dự đoán nhầm các giao dịch tin cậy. Lí giải cho việc hàm đa thức và hàm sigmoid không thực hiện hiệu quả bằng hàm tuyến tính, có thể là do tình huống* **Overfitting** *trên tập* **training**.

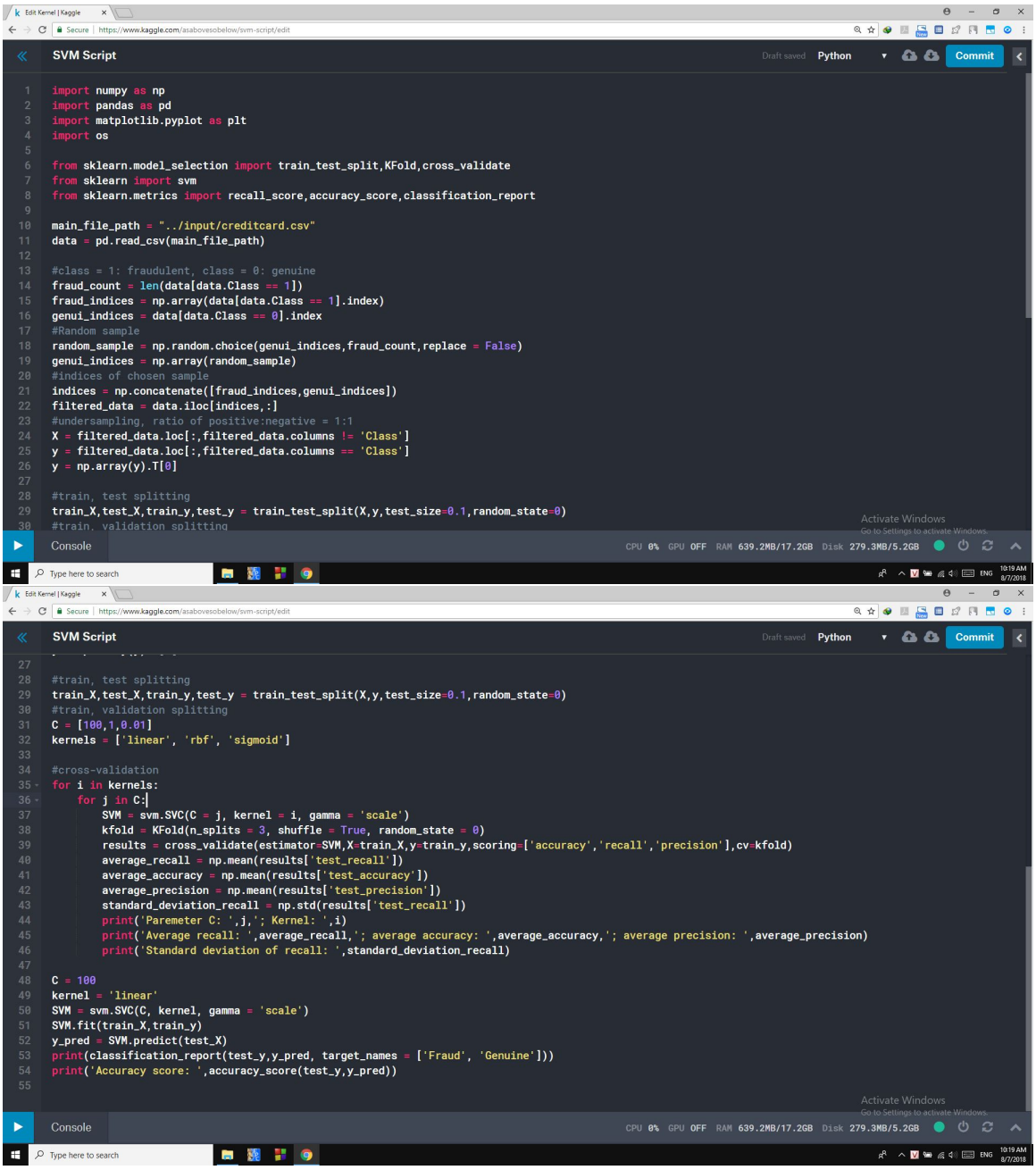
5.4 Đánh giá mô hình thực tế

Thuật toán dự đoán các giao thức lừa đảo đạt được các kết quả sau trên tập **test**:

- Precision đạt được 0.88.
- Recall đạt được 1.00.

	Precision	Recall
Giao dịch lừa đảo	0.88	1.00
Giao dịch tin cậy	1.00	0.85
Độ chính xác	0.929	

6 Phụ lục



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

from sklearn.model_selection import train_test_split,KFold,cross_validate
from sklearn import svm
from sklearn.metrics import recall_score,accuracy_score,classification_report

main_file_path = "../input/creditcard.csv"
data = pd.read_csv(main_file_path)

#class = 1: fraudulent, class = 0: genuine
fraud_count = len(data[data.Class == 1])
fraud_indices = np.array(data[data.Class == 1].index)
genui_indices = data[data.Class == 0].index
#Random sample
random_sample = np.random.choice(genui_indices,fraud_count,replace = False)
genui_indices = np.array(random_sample)
#indices of chosen sample
indices = np.concatenate([fraud_indices,genui_indices])
filtered_data = data.iloc[indices,:]
#undersampling, ratio of positive:negative = 1:1
X = filtered_data.loc[:,filtered_data.columns != 'Class']
y = filtered_data.loc[:,filtered_data.columns == 'Class']
y = np.array(y).T[0]

#train, test splitting
train_X,test_X,train_y,test_y = train_test_split(X,y,test_size=0.1,random_state=0)
#train, validation splitting
C = [100,1,0.01]
kernels = ['linear', 'rbf', 'sigmoid']

#cross-validation
for i in kernels:
    for j in C:
        SVM = svm.SVC(C = j, kernel = i, gamma = 'scale')
        kfold = KFold(n_splits = 3, shuffle = True, random_state = 0)
        results = cross_validate(estimator=SVM,X=train_X,y=train_y,scoring=['accuracy','recall','precision'],cv=kfold)
        average_recall = np.mean(results['test_recall'])
        average_accuracy = np.mean(results['test_accuracy'])
        average_precision = np.mean(results['test_precision'])
        standard_deviation_recall = np.std(results['test_recall'])
        print('Paremeter C: ',j,'; Kernel: ',i)
        print('Average recall: ',average_recall,'; average accuracy: ',average_accuracy,'; average precision: ',average_precision)
        print('Standard deviation of recall: ',standard_deviation_recall)

C = 100
kernel = 'linear'
SVM = svm.SVC(C, kernel, gamma = 'scale')
SVM.fit(train_X,train_y)
y_pred = SVM.predict(test_X)
print(classification_report(test_y,y_pred, target_names = ['Fraud', 'Genuine']))
print('Accuracy score: ',accuracy_score(test_y,y_pred))
```

Tài liệu

[1] Vũ Hữu Tiệp. *Machine Learning Cơ Bản*. 2017.

[2] Machine Learning Group ULB. Credit card fraud detection dataset, 2013.