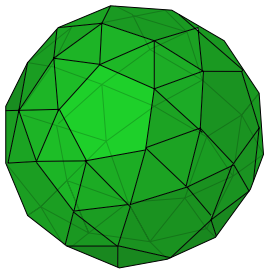
Projects in Mathematics and Applications

SUPPORT VECTOR MACHINES

Ngày 27 tháng 8 năm 2018

Nguyễn Minh Hải [*]
Trần Thanh Thiện [‡]
[†]Trần Minh Thảo
[§]Đắc Tùng Dương



[*]Trường THPT Chuyên Quốc Học, Huế
[†]Trường THPT Phạm Văn Đồng, Quảng Ngãi
[‡]Trường Phổ Thông Năng Khiếu, Đại học Quốc gia Thành phố Hồ Chí Minh
[§]Trường THPT Chuyên Khoa Học Tự Nhiên, Hà Nội

Lời cảm ơn

Trong suốt thời gian học tập tại Trại hè Toán và Ứng dụng Projects in Mathematics and Applications (PiMA) 2018, chúng tôi đã nhận được rất nhiều sự quan tâm, giúp đỡ của nhiều cá nhân và các đoàn thể. Với lòng biết ơn sâu sắc nhất, chúng tôi xin gửi lời cám ơn đến các anh chị mentors, đặc biệt là anh Trần Hoàng Bảo Linh với tất cả sự nhiệt tình, tâm huyết để truyền đạt vốn kiến thức quý báu cho chúng tôi. Bên cạnh đó, chúng tôi cũng xin chân thành cám ơn đến Trường Đại học Khoa học Tự nhiên Thành phố Hồ Chí Minh đã tạo mọi điều kiện tốt nhất về cở sở vật chất, cũng như những buổi nói chuyện, thảo luận về lĩnh vực Học Máy. Vì kiến thức và kinh nghiệm thực tiễn còn hạn chế, trong quá trình thực tập, thực hiện bài báo cáo này, chúng tôi không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ quý anh chị mentors và bạn đọc để dự án có thể hoàn thiện tốt hơn.

Tóm tắt nội dung

Support Vector Machines (SVMs) đã trở thành một công cụ ngày càng phổ biến trong bài toán phân lớp dữ liệu được áp dụng trong cả bài toán phân loại lẫn hồi quy. Xây dựng một SVM đòi hỏi phải giải quyết một bài toán lớn – quadratic programming (QP). Tuy nhiên, bài toán tối ưu gốc này khó áp dụng trực tiếp do hạn chế về bộ nhớ. Do đó, bài toán đối ngẫu thường được giải để vượt qua các thiếu sót trên. Thực ra, bài toán đối ngẫu cũng là một QP nhưng vector nghiệm là sparse nên trong nhiều trường hợp thực tế có thể được giải nhanh hơn bởi máy tính.
SVM thuần chỉ làm việc khi dữ liệu của hai classes là linearly separable. Với những bài toán mà dữ liệu gần linearly separable hoặc nonlinearly separable có những phương pháp bổ trợ cho SVM để thích nghi với dữ liệu đó, một trong số chúng là sử dụng các *Kernel*. Hơn thế nữa, phương pháp Kernel còn nâng cao tầm quan trọng của bài toán đối ngẫu.
Cấu trúc của bản báo cáo như sau:

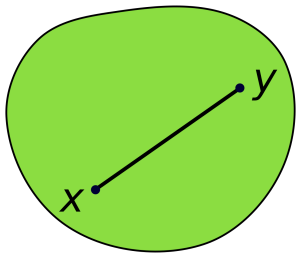
- Phần 1: Giới thiệu bài toán tối ưu lồi và các khái niệm bổ trợ như tập hợp lồi, hàm lồi và một số tính chất, ví dụ về hàm lồi.
- Phần 2: Giới thiệu bài toán SVM, tính lồi của nó và các khái niệm nền tảng như bài toán phân loại nhị phân, siêu phẳng chia tuyến tính.
- Phần 3: Giới thiệu bài toán đối ngẫu SVM và ý nghĩa của nó.
- Phần 4, 5: Giới thiệu mô hình cải tiến, Kernel SVM, và một số ví dụ của hàm Kernel.
- Phần 6: Mô tả ứng dụng của SVM vào một bài toán phân loại với dữ liệu cụ thể và kết quả thu được.
-

Mục lục

1 Giới thiệu bài toán tối ưu lồi

1.1 Tập lồi, hàm lồi, hàm lồi chặt

Định nghĩa 1.1. Một tập hợp được gọi là *tập lồi* (convex set) nếu đoạn thẳng nối hai điểm *bất kỳ* trong tập hợp đó nằm trọn vẹn trong tập hợp đó.



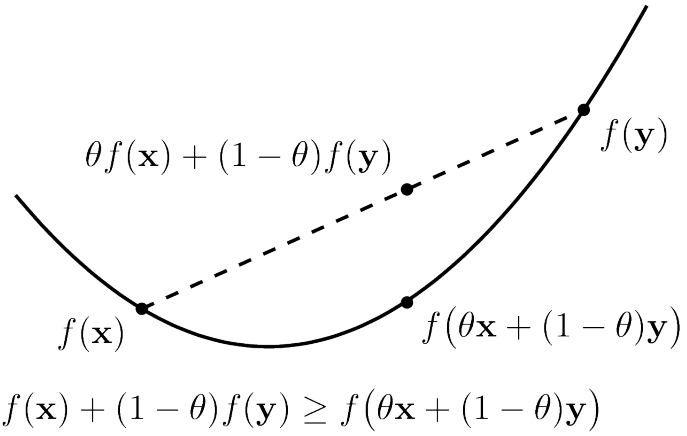
Hình 1: Ví dụ về tập lồi

Định nghĩa 1.2. Một tập hợp $\mathcal{C}$ được gọi là lồi nếu với hai điểm bất kỳ $\mathbf{x_1}, \mathbf{x_2} \in \mathcal{C}$, điểm $\mathbf{x}_\theta = \theta\mathbf{x_1} + (1-\theta)\mathbf{x_2}$ cũng nằm trong $\mathcal{C}$ với bất kỳ $0 \leqslant \theta \leqslant 1$.
Có thể thấy rằng, tập hợp các điểm có dạng $\theta\mathbf{x_1} + (1-\theta)\mathbf{x_2}$ chính là *đoạn thẳng* nối hai điểm $\mathbf{x_1}$ và $\mathbf{x_2}$.

Định nghĩa 1.3. Một hàm số $f : \mathbb{R}^n \to \mathbb{R}$ được gọi là một *hàm lồi* (convex function) nếu tập xác định của f (dom f) là một tập lồi, và:

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leqslant \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$$

với mọi $\mathbf{x}$, $\mathbf{y} \in$ dom $f, 0 \leqslant \theta \leqslant 1$.



Hình 2: Hàm lồi

Định nghĩa 1.4. Một hàm số f được gọi là lõm nếu $-f$ là lồi.

Định nghĩa 1.5. Một hàm số $f : \mathbb{R}^n \to \mathbb{R}$ được gọi là một *hàm lồi chặt* (strictly convex function) nếu tập xác định của f là một tập lồi, và:

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) < \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$$

với mọi $\mathbf{x}$, $\mathbf{y} \in$ dom $f, \mathbf{x} \neq \mathbf{y}, 0 < \theta < 1$.

1.2 Một số tính chất của hàm lồi

Định lý 1.6. *Nếu một hàm số là hàm lồi chặt và có điểm cực trị địa phương, thì điểm cực trị địa phương đó là điểm cực tiểu duy nhất và cũng là cực tiểu toàn cục.*

Chứng minh. Gọi $\mathbf{x_0}$ là điểm cực trị địa phương của f, tức là $f(\mathbf{x_0})$ là cực tiểu trong một lân cận nào đó $\implies \exists R > 0$ để:

$$f(\mathbf{x_0}) = inf\left\{f(\mathbf{x})|\left\|\mathbf{x}-\mathbf{x_0}\right\|_2 < R\right\}$$

Giả sử $\mathbf{x_0}$ không phải điểm cực trị toàn cục, suy ra:

$$\exists \mathbf{y} : f(\mathbf{y}) < f(\mathbf{x_0})(\mathbf{y} \text{ không thuộc lân cận đang xét})$$

Đặt $\mathbf{z} = (1-\alpha)\mathbf{x_0} + \alpha\mathbf{y}$.
Ta có $\left\|\mathbf{z}-\mathbf{x_0}\right\|_2 = \left\|\alpha(\mathbf{y}-\mathbf{x_0})\right\|_2$ nên có thể chọn $\alpha > 0$ đủ nhỏ để $\left\|\mathbf{z}-\mathbf{x_0}\right\|_2 < R$.

$$\begin{aligned}\implies f(\mathbf{x}_0) &\leqslant f(\mathbf{z}) = f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{y}) \\ &< (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{y}) \\ &< (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_0) \\ &= f(\mathbf{x}_0)(\text{Vô lý})\end{aligned}$$

Vậy giả sử sai, ta có điều phải chứng minh. □

1.2.1 Kiểm tra tính chất lồi dựa vào đạo hàm

Để nhận biết một hàm số khả vi có là hàm lồi hay không, ta dựa vào các đạo hàm bậc nhất của nó.
Với hàm nhiều biến, đặt $\nabla f(\mathbf{x}_0)$ là gradient của hàm số f tại điểm $\mathbf{x}_0$, phương trình mặt tiếp tuyến được cho bởi:

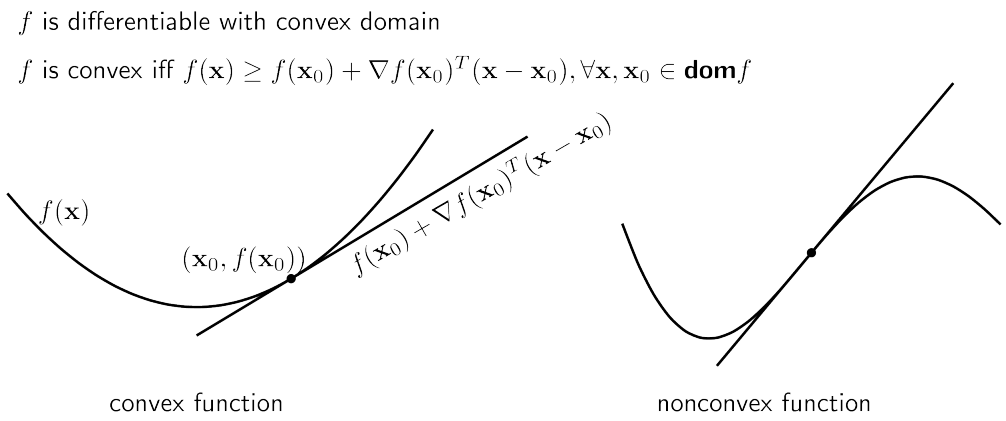
$$y = \nabla f(\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0) + f(\mathbf{x}_0)$$

Định lý 1.7 (Điều kiện bậc nhất)**.** *Giả sử hàm số f có $D = dom\ f$ là một tập lồi, khả vi trên toàn D. Khi đó, hàm số f là lồi* ***nếu và chỉ nếu***

$$f(\mathbf{x}) \geq \nabla f(\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0) + f(\mathbf{x}_0) \quad \forall \mathbf{x}, \mathbf{x_0} \in D \tag{1}$$

Tương tự như thế, một hàm số là stricly convex nếu dấu bằng trong (1) xảy ra khi và chỉ khi $\mathbf{x} = \mathbf{x_0}$.

Nói một cách trực quan hơn, một hàm số là lồi nếu đường tiếp tuyến tại một điểm bất kỳ trên đồ thị của hàm số đó nằm dưới đồ thị đó.



Hình 3: Hàm lồi (bên trái), hàm không lồi (bên phải)

1.2.2 Xây dựng các hàm lồi/lõm từ các hàm lồi/lõm

Các tính chất cơ bản sau của hàm lồi/lõm cho phép ta xây dựng thêm những hàm lồi/lõm khác từ một số hàm lồi/lõm cho trước:

Tính chất 1.8. *Cho f và g là 2 hàm số trên cùng miền xác định D và số thực $a \in \mathbb{R}$. Khi đó:*

- *Giả sử f là hàm lồi/lõm. Khi đó nếu $a > 0$ thì $a \cdot f$ là lồi/lõm, nếu $a < 0$ thì $a \cdot f$ là lõm/lồi.*
- *Nếu f và g cùng lồi/lõm thì tổng $f + g$ cũng lồi/lõm.*

Một tính chất nâng cao hơn, nhưng sẽ được dùng đến khi nói tới tính lồi của bài toán đối ngẫu SVM, như sau:

Định lý 1.9. *Cho I là một tập chỉ số (có thể vô hạn không đếm được), và tập hợp các hàm lồi $\{f_i \mid i \in I\}$ có cùng tập xác định D. Khi đó hàm số $f_{\sup} : D \to \mathbb{R}$ thỏa mãn*

$$f_{\sup}(x) = \sup\{f_i(x) \mid i \in I\}$$

cũng là hàm lồi trên D.

Hệ quả 1.10. Với I và D như trên, nếu ngược lại $\{f_i \mid i \in I\}$ là các hàm lõm thì hàm số $f_{\inf} = \inf\{f_i \mid i \in I\}$ cũng là hàm lõm.

1.3 Một số ví dụ của hàm lồi

Như sẽ nói tới sau đây, bài toán SVM là một bài toán *quy hoạch bậc 2* (*Quadratic Programming*), tức là hàm mục tiêu và ràng buộc là các hàm số có bậc không quá 2. Do vậy ta quan tâm tới tính lồi/lõm của các hàm số nhiều biến có bậc không quá 2, mà cụ thể là hàm tuyến tính và hàm norm bình phương $\|.\|_2^2$.

Định lý 1.11. *Từ định nghĩa 1.3 và định nghĩa 1.4, ta chứng minh được các hàm tuyến tính vừa lồi, vừa lõm.*

Định nghĩa 1.12. Ma trận đối xứng $\mathbf{A}_{n\times n}$ được gọi là *xác định dương (postive definite)* nếu với mọi vector $\mathbf{x} \in \mathbb{R}^n \setminus \{0\}$, ta có:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$$

Định lý 1.13. *Nếu ma trận đối xứng* $\boldsymbol{A}$ *là xác định dương thì hàm số* $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ *là hàm lồi*

Chứng minh. Ta có

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^T\mathbf{A}\mathbf{x} \\ &= \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix} \begin{bmatrix} \sum_{i=1}^n a_{1i}x_i \\ \sum_{i=1}^n a_{2i}x_i \\ \vdots \\ \sum_{i=1}^n a_{ni}x_i \end{bmatrix} \\ &= \sum_{j=1}^n (\sum_{i=1}^n a_{ij}x_i)x_j = \sum_{\substack{i=1 \\ j=1}}^n a_{ji}x_jx_i \end{aligned}$$

Do đó

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \sum_{\substack{j=1 \\ j\neq i}}^n a_{ji}x_j + \sum_{\substack{k=1 \\ k\neq i}}^n a_{ik}x_k + 2a_{ii}x_i = \sum_{j=1}^n a_{ji}x_j + \sum_{k=1}^n a_{ik}x_k$$

$$\begin{aligned} \nabla\mathbf{f} &= \begin{bmatrix} \sum_{j=1}^n a_{j1}x_j \\ \sum_{j=1}^n a_{j2}x_j \\ \vdots \\ \sum_{j=1}^n a_{jn}x_j \end{bmatrix} + \begin{bmatrix} \sum_{k=1}^n a_{1k}x_k \\ \sum_{k=1}^n a_{2k}x_k \\ \vdots \\ \sum_{k=1}^n a_{nk}x_k \end{bmatrix} \\ &= \mathbf{A}^T\mathbf{x} + \mathbf{A}\mathbf{x} \\ &= (\mathbf{A} + \mathbf{A^T})\mathbf{x} \\ &= 2\mathbf{A}\mathbf{x} \end{aligned}$$

Do $\mathbf{A}$ là ma trận đối xứng nên $\mathbf{A} = \mathbf{A}^T$
Vậy first-order condition có thể viết dưới dạng:

$$\begin{aligned} &\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 2(\mathbf{A}\mathbf{x_0})^T(\mathbf{x} - \mathbf{x_0}) + \mathbf{x_0}^T\mathbf{A}\mathbf{x_0} \\ \iff &\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 2(\mathbf{x_0})^T\mathbf{A}\mathbf{x} - \mathbf{x_0}^T\mathbf{A}\mathbf{x_0} \\ \iff &(\mathbf{x} - \mathbf{x_0})^T\mathbf{A}(\mathbf{x} - \mathbf{x_0}) \geq 0 \end{aligned}$$

Bất đẳng thức cuối cùng là đúng dựa trên định nghĩa 1.12. Vậy hàm số $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ là hàm lồi. □

Hệ quả 1.14. Hàm norm bình phương $\mathbf{x} \mapsto \|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{I}\mathbf{x}$ là hàm lồi.

1.4 Bài toán tối ưu lồi

Bài toán tối ưu lồi tổng quát

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

thỏa mãn

$$\begin{cases} f_n(\mathbf{x}) \leqslant 0, \forall n = 1, 2, ..., m \\ \mathbf{a}_n^T \mathbf{x} + b_n = 0, \forall n = 1, 2, ..., p \end{cases}$$

trong đó f và $f_n, \forall n = 1, 2, ..., m$ là các hàm lồi.

Tính chất quan trọng của bài toán tối ưu lồi đó chính là bất kỳ điểm cực tiểu địa phương nào cũng chính là điểm cực tiểu toàn bộ.
Tính chất này có được dựa vào hàm mục tiêu $f(\mathbf{x})$ là hàm lồi và Định lý 1.6.

2 Bài toán SVM gốc

Bài toán SVM gốc là bài toán phân loại nhị phân, mục tiêu là xây dựng một siêu phẳng phân chia tuyến tính từ dữ liệu của hai tập được gán nhãn cho trước.

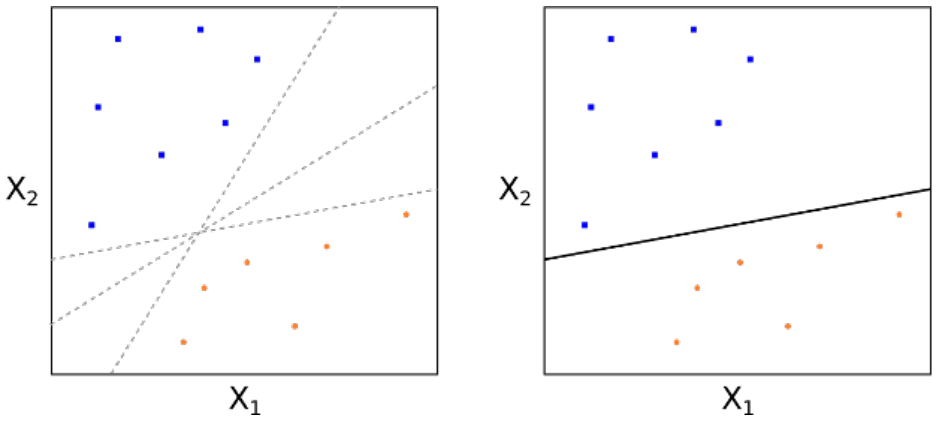
2.1 Bài toán phân loại nhị phân

Phát biểu bài toán: Dữ liệu đầu vào gồm các vector $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$ trong không gian d chiều là $\mathbb{R}^d$. Mỗi vector được gán một nhãn y_i tương ứng, ta được n cặp dữ liệu

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ...(\mathbf{x_n}, y_n)$$

trong đó $y_i = 1$ nếu $\mathbf{x_i}$ thuộc class thứ nhất, $y_i = -1$ nếu $\mathbf{x_i}$ thuộc class thứ hai.

Bài toán của ta là xác định class cho một điểm dữ liệu đầu vào mới bất kỳ.
SVM là một công cụ giúp chúng ta giải quyết bài toán trên. Chúng ta giả định rằng thông qua một phương tiện chưa xác định, có thể xây dựng được một hyperplane phân tích dữ liệu một cách hoàn hảo nhất theo nhãn của chúng.

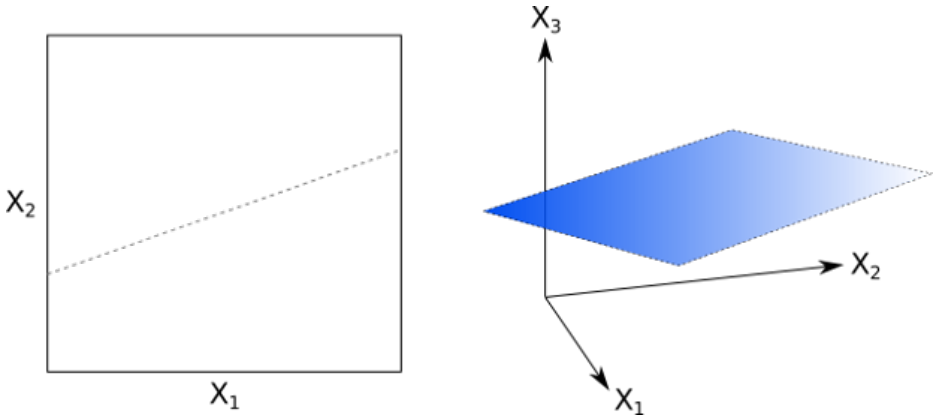


Hình 4: Các siêu phẳng (bên trái), Siêu phẳng phân chia dữ liệu hoàn hảo (bên phải)

Vậy làm thế nào để xây dựng siêu phẳng phân chia dữ liệu hoàn hảo này. Trong phần tiếp theo chúng ta sẽ tiếp cận với phương pháp này cũng như giới thiệu khái niệm maximal margin hyperplane và một mô hình phân loại được xây dựng dựa trên nó, maximal margin classifier.

2.2 Siêu phẳng chia tuyến tính

Định nghĩa 2.1. Siêu phẳng chia tuyến tính (Linear Separating Hyperplane) là một khái niệm quan trọng trong SVM. Với đầu vào là dữ liệu nhiều chiều thì linear separating hyperplane là một đối tượng có ít chiều hơn, có khả năng chia không gian dữ liệu thành hai vùng.
Linear separating hyperplane không cần phải đi qua điểm gốc, nghĩa là nó không bao gồm vector $\overrightarrow{0}$. Xét một không gian n chiều, linear separating hyperplane là một affine có $n-1$ chiều. Từ đó suy ra trong không gian hai chiều, hyperplane này chính là một đường thẳng, trong không gian ba chiều, nó là một mặt phẳng.

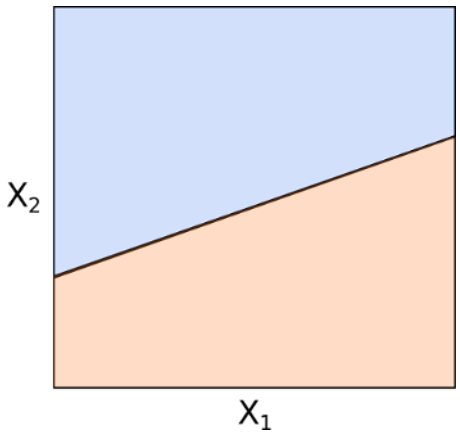


Hình 5: Siêu phẳng một chiều (bên trái) và siêu phẳng hai chiều (bên phải)

Định nghĩa 2.2. Một hyperplane (siêu mặt phẳng) trong không gian d chiều là tập hợp các điểm thỏa mãn phương trình:

$$a_1x_1 + a_2x_2 + ... + a_nx_n + b = \mathbf{a}^T\mathbf{x} + b = 0$$

Với $b, a_i, i = 1, 2, ..., n$ là các số thực.
Hyperplane này chia không gian thành hai miền (Hình 6).



Hình 6: Siêu phẳng chia cắt không gian n chiều

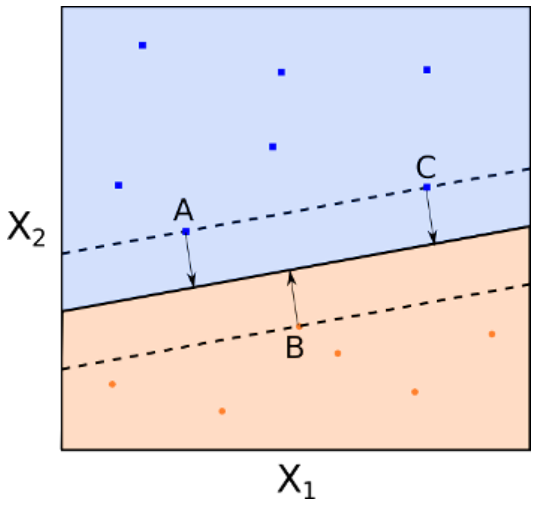
Những phần tử $\mathbf{x}$ nằm trên siêu phẳng có dạng $\mathbf{a}^T\mathbf{x} + b > 0$, trong khi các phần tử nằm dưới có dạng $\mathbf{a}^T\mathbf{x} + b < 0$. Như vậy, chúng ta có thể xác định một điểm dữ liệu mới $\mathbf{x}$ nằm trên hay nằm dưới siêu phẳng ban đầu bằng cách tính toán dấu của biểu thức $\mathbf{a}^T\mathbf{x} + b$. Khái niệm này sẽ là cơ sở của SVM.

2.3 Xây dựng Maximal Margin Classifier

Sự phân loại phân chia chính xác hai lớp dữ liệu trên, nhưng nó có thể cho chúng ta vô số nghiệm như hình bên trái của hình 4.
Vì vậy, chúng ta không chỉ đơn giản xây dựng một siêu phẳng như vậy mà phải làm sao để thu được siêu phẳng tối ưu nhất. Điều này giúp ta đi đến khái niệm Maximal Margin Hyperplane (MMH) là separating hyperplane có khoảng cách xa nhất từ bất kỳ điểm dữ liệu nào.

Vậy làm thế nào để tìm được Maximal Margin Hyperplane? Đầu tiên, chúng ta tính khoảng cách từ mỗi điểm dữ liệu $\mathbf{x}_i$ đến một separating hyperplane. Khoảng cách ngắn nhất được gọi là margin. MMH là separating hyperplane có margin lớn nhất. Điều này đảm bảo rằng, nó là khoảng cách tối thiểu xa nhất với các điểm dữ liệu. Sau đó, chúng ta xác định các điểm dữ liệu rơi vào bên nào của hyperplane.

Sau khi xây dựng được MMH, việc xác định class của một diểm dữ liệu đầu vào mới bất kì trở nên vô cùng đơn giản. Phương pháp phân loại này được gọi là Maximinal Margin Classifier (MMC).
Một trong những đặc điểm chính của MMC (và sau đó là SVC và SVM) là vị trí của MMH chỉ phụ thuộc vào support vectors, đó là tập hợp các điểm nằm trên ranh giới của margin (xem điểm A, B và C trên hình). Điều này có nghĩa là vị trí của MMH **không** phụ thuộc vào các dữ liệu đầu vào còn lại.



Hình 7: Maximal Margin Hyperplane và Suport Vetors

Nhận xét quan trọng: Nhược điểm của MMC và MMH là nó phụ thuộc hoàn toàn vào vị trí của support vectors (trong nhiều trường hợp thì support vectors không mang tính chất chung của các dữ liệu đầu vào). Tuy nhiên, điều này cũng khiến SVM trở thành một công cụ hấp dẫn trong machine learning, vì chúng ta chỉ cần có support vectors (tức là các giá trị a_i được cố định).

Định lý 2.3. *Khoảng cách từ một điểm (vector) có tọa độ* $\mathbf{x_0}$ *tới siêu phẳng có phương trình* $\boldsymbol{a}^T\boldsymbol{x} + b = 0$ *được xác định bởi:*

$$\frac{\left|\boldsymbol{a}^T\boldsymbol{x}_0 + b\right|}{\|\boldsymbol{a}\|_2}$$

Với $\|\boldsymbol{a}\|_2 = \sqrt{\sum_{i=1}^{d} a_i^2}$ *với* d *là số chiều của không gian.*

Với cặp dữ liệu $(\mathbf{x_n}, y_n)$, khoảng cách từ điểm đó đến siêu phẳng phân chia, tức margin ứng với cặp $(\mathbf{a}, b)$ là:

$$\text{margin}(\mathbf{a}, b) = \min_n \frac{y_n \left(\mathbf{a}^T \mathbf{x_n} + b\right)}{\|\mathbf{a}\|_2}$$

Chứng minh. y_n luôn cùng dấu với $\mathbf{x_n}$ nên y_n cùng dấu với $(\mathbf{a}^T\mathbf{x_n} + b)$, và tử số luôn là một số không âm. Từ đó áp dụng công thức tính khoảng cách (đã nêu ở trên), ta tìm được margin. □

Như vậy MMH là nghiệm của phương trình:

$$(\mathbf{a}, b) = \arg\max_{\mathbf{a},b} \text{ margin } = \arg\max_{\mathbf{a},b} \min_n \frac{y_n \left(\mathbf{a}^T \mathbf{x_n} + b\right)}{\|\mathbf{a}\|_2}$$

Đây cũng chính là bài toán tối ưu trong SVM. Tuy nhiên, việc giải trực tiếp bài toán này sẽ rất phức tạp, vì thế ta cần phải đơn giản hóa nó.

2.4 Bài toán tối ưu trong SVM

Ta có một nhận xét quan trọng là nếu thay $\mathbf{a}, b$ lần lượt bởi $k\mathbf{a}, kb$ (với $k \neq 0$) thì siêu phẳng H của ta là không đổi, do đó ta hoàn toàn có thể giả sử $\min y_n(\mathbf{a}^T\mathbf{x_n} + b) = 1$.
Từ đó ta có

$$y_n(\mathbf{a}^T\mathbf{x_n} + b) \geqslant 1, \forall i = 1, 2, ..., N$$

Từ đó bài toán tối ưu SVM của ta trở thành

$$(\mathbf{a}, b) = \arg\max_{\mathbf{a},b} \frac{1}{\|\mathbf{a}\|_2} \qquad \text{với điều kiện} \qquad y_n(\mathbf{a}^T\mathbf{x_n} + b) \geqslant 1$$

Bằng một số biến đổi đơn giản, ta đưa bài toán trở thành

Bài toán 2.4 (Bài toán SVM gốc)**.**

$$(\mathbf{a}, b) = \arg\min_{\mathbf{a},b} \frac{1}{2} \|\mathbf{a}\|_2^2 \qquad \text{với điều kiện} \qquad 1 - y_n(\mathbf{a}^T\mathbf{x_n} + b) \geqslant 0$$

(Mục đích là ta thu được một hàm khả vi và có đạo hàm đẹp hơn)
Sau khi xác định được siêu phẳng H, để xác định class của một điểm dữ liệu mới $\mathbf{x}$, ta sẽ tính $\mathbf{a}^T\mathbf{x} + b$, nếu giá trị này âm thì sẽ thuộc class -1, nếu dương thì thuộc class 1.

Nhận xét 2.5. Trong bài toán tối ưu SVM trên, hàm mục tiêu là một *norm* nên là một hàm lồi, các hàm bất đẳng thức ràng buộc là hàm tuyến tính nên cũng là các hàm lồi, do đó bài toán của ta là một bài toán tối ưu lồi và là một *Quadratic Programming*. Và với các Quadratic Programming thì đã có thuật toán giải và các thư viện như CVXOPT, tuy nhiên trong phạm vi bài viết này ta sẽ không đề cập đến.

Ta sẽ quan tâm nhiều hơn đến bài toán đối ngẫu của SVM và lí giải vì sao người ta lại giải bài toán đối ngẫu thay vì giải bài toán gốc.

3 Bài toán đối ngẫu cho SVM

Bài toán SVM là một bài toán tối ưu lồi có ràng buộc và các bài toán tối ưu lồi có ràng buộc nói chung đều có bài toán đối ngẫu. Các bài toán đối ngẫu đều có thể giải được bằng cách tối ưu hóa hàm số đối ngẫu Lagrange thông qua Lagrangian. Trong bài toán đối ngẫu SVM, *đối ngẫu mạnh* xảy ra và ta tìm được nghiệm của bài toán thông qua *hệ điều kiện KKT.*

3.1 Lagrangian

Lagrangian của bài toán tối ưu SVM là

$$\mathcal{L}(\mathbf{a}, b, \lambda) = \frac{1}{2}\|\mathbf{a}\|_2^2 + \sum_{n=1}^{N} \lambda_n \left(1 - y_n(\mathbf{a}^T\mathbf{x_n} + b)\right)$$

trong đó $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N] \succeq 0$ (hay là $\lambda_n \geqslant 0, \forall n = 1, 2, .., N$)

3.2 Hàm số đối ngẫu Lagrange

Hàm đối ngẫu Lagrange được định nghĩa là

$$g(\lambda) = \min_{\mathbf{a},b} \mathcal{L}(\mathbf{a}, b, \lambda)$$

trong đó $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N] \succeq 0$ (hay là $\lambda_n \geqslant 0, \forall n = 1, 2, .., N$)

Ta sẽ tìm cách tính toán cho hàm $g(\lambda)$ này.
Xét theo biến $\mathbf{a}, b$ thì $\mathcal{L}(\mathbf{a}, b, \lambda)$ là tổng của một norm và các hàm tuyến tính nên là một hàm lồi. Theo **Định lí 1.6** thì điểm cực tiểu của bài toán chính là nghiệm của hệ điều kiện

$$\begin{cases} \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{a}, b, \lambda) = 0 \\ \nabla_b\mathcal{L}(\mathbf{a}, b, \lambda) = 0 \end{cases}$$

Ta có

$$\nabla_{\mathbf{a}}\mathcal{L}(\mathbf{a}, b, \lambda) = \mathbf{a}^T - \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n = \mathbf{a}^T - \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n$$

$$\nabla_b\mathcal{L}(\mathbf{a}, b, \lambda) = -\sum_{n=1}^{N} \lambda_n y_n$$

Từ đó ta có

$$\begin{cases} \mathbf{a}^T = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n \\ \sum_{n=1}^{N} \lambda_n y_n = 0 \end{cases}$$

Từ đó ta tính được

$$g(\lambda) = \frac{1}{2}\|\mathbf{a}\|_2^2 + \sum_{n=1}^{N} \lambda_n + b\sum_{n=1}^{N} \lambda_n y_n - \sum_{n=1}^{N} \mathbf{a}^T \lambda_n y_n \mathbf{x}_n = \sum_{n=1}^{N} \lambda_n - \frac{1}{2}\|\mathbf{a}\|_2^2$$

Lại có

$$\|\mathbf{a}\|_2^2 = \sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m$$

Nên

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

Đặt

$$\mathbf{V} = [y_1 \mathbf{x}_1, ..., y_N \mathbf{x}_N]$$

thì ta có $\mathbf{a} = \mathbf{V}\lambda$ do đó $\|\mathbf{a}\|_2^2 = \lambda^T \mathbf{V}^T \mathbf{V} \lambda$
Từ đó ta có thể viết gọn

$$g(\lambda) = \mathbf{1}^T \lambda - \frac{1}{2} \lambda^T \mathbf{V}^T \mathbf{V} \lambda$$

(Ta kí hiệu $\mathbf{1}$ là vector hàng gồm toàn số 1)

3.3 Bài toán đối ngẫu Lagrange

Kết hợp hàm đối ngẫu Lagrange và các điều kiện ràng buộc của λ ta có bài toán đối ngẫu Lagrange sau

Bài toán 3.1 (Bài toán đối ngẫu SVM)**.**

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

thỏa mãn

$$\begin{cases} \lambda \succeq 0 \\ \sum_{n=1}^{N} \lambda_n y_n = 0 \end{cases}$$

Nhận xét rằng theo biến λ thì $\mathcal{L}(\mathbf{a}, b, \lambda)$ là một hàm lõm, mà $g(\lambda)$ là inf của các hàm lõm do đó $g(\lambda)$ cũng là một hàm lõm. Từ đó bài toán đối ngẫu của SVM chính là một bài toán tối ưu lồi.

Định lý 3.2. *Nghiệm của bài toán đối ngẫu SVM thỏa mãn hệ điều kiện KKT.*

Ý tưởng chứng minh. Nếu bài toán này thỏa mãn điều kiện Slater thì *đối ngẫu mạnh* sẽ xảy ra và nghiệm của bài toán sẽ là nghiệm của *hệ điều kiện KKT*.

Ta sẽ chứng minh bài toán thỏa mãn điều kiện Slater, tức là tồn tại điểm *strictly feasible*. Ta cần chứng minh tồn tại $(\mathbf{a}, b)$ sao cho

$$1 - y_n(\mathbf{a}^T \mathbf{x}_n + b) < 0, \forall n = 1, 2, ..., N$$

Nhận xét rằng training data của ta là phân biệt tuyến tính (linearly separable) nên luôn tồn tại siêu phẳng H phân chia dữ liệu thành hai class, tức là tồn tại $(\mathbf{a}_0, b_0)$ thỏa mãn

$$1 - y_n(\mathbf{a}_0^T \mathbf{x}_n + b_0) \leqslant 0, \forall n = 1, 2, ..., n$$

Tương đương với

$$2 - y_n(2\mathbf{a}_0^T \mathbf{x}_n + 2b_0) \leqslant 0$$

Chọn $\mathbf{a} = 2\mathbf{a}_0$ và $b = 2b_0$ thì ta có

$$1 - y_n(\mathbf{a}^T \mathbf{x}_n + b) = -1 < 0$$

Vậy bài toán của ta thỏa mãn điều kiện Slater, do đó *đối ngẫu mạnh* xảy ra, tức là ta có

$$\min \frac{1}{2} \|\mathbf{a}\|_2^2 = g(\lambda)$$

Nghiệm của bài toán là nghiệm của *hệ điều kiện KKT* sau

$$\begin{cases} 1 - y_n(\mathbf{a}^T\mathbf{x}_n + b) & \leqslant 0 \quad , n = 1, 2, \ldots, N \\ \lambda_n & \geqslant 0 \quad , n = 1, 2, \ldots, N \\ \lambda_n(1 - y_n(\mathbf{a}^T\mathbf{x}_n + b)) & = 0 \quad , n = 1, 2, \ldots, N \\ \mathbf{a}^T & = \sum_{n=1}^N \lambda_n y_n \mathbf{x}_n \\ \sum_{n=1}^N \lambda_n y_n & = 0 \end{cases} \tag{2}$$

□

Từ phương trình thứ ba thì ta nhận xét rằng với mỗi n thì ta luôn có $\lambda_n = 0$ hoặc $y_n(\mathbf{a}^T\mathbf{x}_n + b) = 1$. Ta chỉ quan tâm tới những điểm mà $\lambda_n \neq 0$, vì khi đó $y_n(\mathbf{a}^T\mathbf{x}_n + b) = 1$ và $\mathbf{x}_n$ nằm trên siêu phẳng $\mathbf{a}^T\mathbf{x} + b = \pm 1$.
Đặt $\mathcal{S} = \{n \mid \lambda_n \neq 0\}$. Với mỗi $n \in \mathcal{S}$, ta có

$$y_n(\mathbf{a}^T\mathbf{x}_n + b) = 1 \quad \iff \quad (\mathbf{a}^T\mathbf{x}_n + b) = y_n \quad \iff \quad b = y_n - \mathbf{a}^T\mathbf{x}_n$$

và

$$\mathbf{a}^T = \sum_{n \in \mathcal{S}} \lambda_n y_n \mathbf{x}_n$$

Với một giá trị n thì ta đã tính được b khi đã có $\mathbf{a}$, tuy nhiên kết quả ổn định và thường được sử dụng hơn là lấy trung bình của tất cả các cách tính b

$$b = \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(y_n - \mathbf{a}^T\mathbf{x}_n\right) = \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T \mathbf{x}_n\right)$$

Và để xác định class của một điểm dữ liệu $\mathbf{x}$ mới, ta xác định dấu của

$$\mathbf{a}^T\mathbf{x} + b = \sum_{n \in \mathcal{S}} \lambda_n y_n \mathbf{x}_n^T \mathbf{x} + \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T \mathbf{x}_n\right)$$

3.4 Ý nghĩa của bài toán đối ngẫu SVM

Bài toán đối ngẫu thường được quan tâm hơn bài toán gốc dù trong vài trường hợp (không nhiều), nó *khó giải* hơn bài toán gốc, vì một số lý do sau:

1. **Vector λ là một *sparse vector***
   Quan sát hệ điều kiện KKT, ở phương trình thứ 3 ta nhận thấy rằng với mỗi n thì ta luôn có $\lambda_n = 0$ hoặc $y_n(\mathbf{a}^T\mathbf{x}_n + b) = 1$.
   Mà ta đã biết các điểm dữ liệu $\mathbf{x}_n$ thỏa mãn $y_n(\mathbf{a}^T\mathbf{x}_n + b) = 1$ nằm trên hai siêu phẳng $\mathbf{a}^T\mathbf{x} + b = \pm 1$. Những điểm này còn được gọi là *support vector* và thường thì số lượng những điểm này là không nhiều, do đó hầu hết $\lambda_n = 0$ nên λ là một sparse vector. Và do đó làm giảm số chiều của bài toán (do λ có N chiều nhưng hầu hết lại bằng 0), việc này mang lại hiệu quả cao trong việc tính toán.
   Ngược lại, khi giải bài toán gốc thì ta không có thông tin gì về λ.

2. **Dữ liệu không phân biệt tuyến tính**
   Hơn nữa, trong trường hợp dữ liệu là không phân biệt tuyến tính hoặc gần phân biệt tuyến tính thì việc giải bài toán gốc không mang lại hiệu quả cao.
   Khi giải bài toán đối ngẫu, ta nhận được hướng tiếp cận cho trường hợp này từ việc có tính vô hướng giữa các vector $\mathbf{x}_n^T\mathbf{x}_m$. Đó chính là phương pháp Kernel SVM (sẽ được trình bày ở phần tiếp theo)

3. **Bộ nhớ được sử dụng hiệu quả**
   Chỉ một tập con của dữ liệu đầu vào được lưu vào bộ nhớ (những điểm dữ liệu nằm trên support vectors).

4 Kernel Support Vector Machine

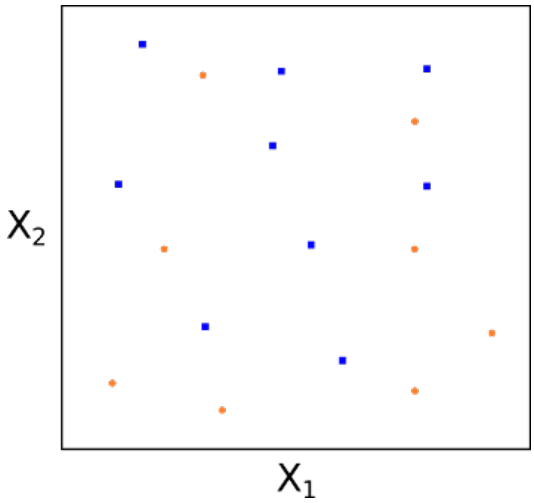
Bài toán đối ngẫu trong Hard Margin SVM là:

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

thỏa mãn

$$\begin{cases} \lambda \succeq 0 \\ \sum_{n=1}^{N} \lambda_n y_n = 0 \end{cases}$$

Tuy nhiên trong thực tế, dữ liệu rất hiếm khi là phân biệt tuyến tính hoặc gần phân phân biệt tuyến tính do đó bài toán trên khó có thể tìm ra một mặt phân chia tốt. (Hình 8)



Hình 8: Các điểm dữ liệu không thể phân biệt tuyến tính

Ý tưởng của Kernel SVM là chuyển dữ liệu của ta sang một không gian nhiều chiều hơn để dữ liệu của ta là phân biệt tuyến tính.

Giả sử ta tìm được hàm $\mathrm{F}(\mathbf{x})$ sao cho dữ liệu mới là $\mathrm{F}(\mathbf{x}_1), \mathrm{F}(\mathbf{x}_2), ..., \mathrm{F}(\mathbf{x}_N)$ là phân biệt tuyến tính.
Trong không gian mới, bài toán trở thành:

$$\lambda = \arg\max_{\lambda} g(\lambda) = \arg\max_{\lambda} \sum_{n=1}^{N} \lambda_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n\lambda_m y_n y_m \mathrm{F}(\mathbf{x})_n^T\mathrm{F}(\mathbf{x}_m)$$

thỏa mãn

$$\begin{cases} \lambda \succeq 0 \\ \sum_{n=1}^{N} \lambda_n y_n = 0 \end{cases}$$

Việc tính toán trực tiếp $\mathrm{F}(\mathbf{x}_n)$ cho mỗi điểm dữ liệu là tốn rất nhiều thời gian và bộ nhớ khi số điểm dữ liệu N là có thể rất lớn và số chiều của $\mathrm{F}(\mathbf{x}_n)$ có thể là rất lớn (hoặc vô hạn). Hơn nữa, việc xác định nhãn của một điểm dữ liệu mới cũng sẽ tốn rất nhiều thời gian và bộ nhớ, do đó thuật toán của ta sẽ không đạt hiệu quả cao. Để khắc phục vấn đề này, người ta đề xuất việc sử dụng hàm kernel, được định nghĩa như sau:

$$k(\mathbf{x}, \mathbf{y}) = \mathrm{F}(\mathbf{x})^T \mathrm{F}(\mathbf{y})$$

Hàm kernel xuất phát từ quan sát rằng trong các biểu thức cần tính thì ta chỉ cần tính tích vô hướng của hai vector mà không cần phải tính chi tiết từng vector, do đó hàm kernel khắc phục được vấn đề tính toán này.
Lúc này ta có bài toán đối ngẫu là:

$$\lambda = \arg\max_{\lambda} g(\lambda) = \arg\max_{\lambda} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$$

thỏa mãn

$$\begin{cases} \lambda \succeq 0 \\ \sum_{n=1}^{N} \lambda_n y_n = 0 \end{cases}$$

Và để xác định dấu của một điểm dữ liệu mới $\mathbf{x}$, ta xét dấu của:

$$\mathbf{a}^T \mathbf{x} + b = \sum_{n \in \mathcal{S}} \lambda_n y_n k(\mathbf{x}_n, \mathbf{x}) + \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m k(\mathbf{x}_m, \mathbf{x}_n) \right)$$

5 Hàm Kernel

5.1 Tính chất của hàm Kernel

1. Tính đối xứng
   Hàm Kernel có tính đối xứng $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$. (Do tính vô hướng của 2 vector có tính đối xứng)

2. Thỏa mãn điều kiện Mercer
   Hàm Kernel cần thỏa mãn điều kiện Mercer

   $$\sum_{n=1}^{N} \sum_{m=1}^{N} k(\mathbf{x}_m, \mathbf{x}_n) c_n c_m \geqslant 0, \quad \forall c_i \geqslant 0$$

   Với hàm Kernel thỏa mãn điều kiện Mercer, ta xét ma trận $\mathbf{K}$ đối xứng thỏa mãn $k_{nm} = y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$. Khi đó xét $c_n = y_n \lambda_n$, ta có

   $$\lambda^T \mathbf{K} \lambda = \sum_{n=1}^{N} \sum_{m=1}^{N} k(\mathbf{x}_m \mathbf{x}_n) y_n y_m \lambda_n \lambda_m \geqslant 0$$

   Từ đây suy ra được $\mathbf{K}$ là một ma trận nửa xác định dương, điều này đảm bảo cho bài toán tối ưu của ta là một bài toán tối ưu lồi.

5.2 Một số hàm Kernel

Một số hàm Kernel thông dụng:

1. Linear Kernel
   Đây là hàm Kernel đơn giản, là tích vô hướng của hai vector

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

2. Polynomial Kernel
   Polynomial Kernel được định nghĩa là

$$k(\mathbf{x}, \mathbf{y}) = \left(r + \mu \mathbf{x}^T \mathbf{y}\right)^d$$

   trong đó d là bậc (là một số dương, có thể không nguyên), r, μ là các số thực.
   Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.

3. Radial Basic Function Kernel
   Radical Basis Function (RBF) là hàm Kernel được sử dụng phổ biến nhất (còn có tên gọi là Gaussian Kernel), được xác định bởi

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\mu \left\|\mathbf{x} - \mathbf{y}\right\|_2^2\right), \quad \mu > 0$$

   Lí do RBF được sử dụng phổ biến là do hàm mũ sau khi khai triển bằng nội suy Taylor thì ta được một hàm đa thức bậc vô hạn, do đó hàm exp là hàm biến đổi lên không gian vô hạn chiều, ta không phải lo lắng về số chiều để dữ liệu phân biệt tuyến tính, vì số chiều là vô hạn.

4. Sigmoid Kernel
   Hàm Sigmoid Kernel được định nghĩa là

$$k(\mathbf{x}, \mathbf{y}) = \tanh\left(\mu \mathbf{x}^T \mathbf{y} + r\right), \quad \mu, r \in \mathbb{R}$$

6 Áp dụng SVM vào mô hình cụ thể

Trên thế giới, số lượng bệnh nhân ung thư đã tăng chóng mặt trong vài thập niên gần đây. Do tính nghiêm trọng của bệnh, các bác sĩ cần đưa ra kết quả chẩn đoán sớm nhất có thể. Tuy nhiên, việc phân tích kết quả xét nghiệm thủ công vô cùng tốn thời gian, chưa kể việc kết luận có thể không chính xác. May mắn thay, máy học có thể giải quyết vấn đề này, bằng cách đưa ra kết quả chẩn đoán nhanh chóng mà vẫn đảm bảo độ chính xác cao.

Ở đây, chúng tôi áp dụng Machine Learning để chẩn đoán bệnh ung thư vú - đứng đầu danh sách những căn bệnh ung thư phổ biến ở nữ giới Việt Nam. Với dữ liệu của một khối u vú cho trước, cần xác định xem khối u đó là lành tính hay ác tính (ung thư). Đây chính là một bài toán phân loại nhị phân, do đó ta có thể giải quyết bằng Support Vector Machines.

Bộ dữ liệu được sử dụng là Wisconsin Diagnostic Breast Cancer (WDBC), cung cấp 10 đặc tính nhân tế bào khối u của 569 bệnh nhân, bao gồm chu vi, bán kính, diện tích bề

mặt, tính đối xứng,... Mỗi đặc tính được biểu hiện bởi 3 thông số: giá trị trung bình, độ lệch chuẩn, giá trị "tồi nhất" (trung bình 3 giá trị lớn nhất). Trong quá trình giải bài toán, mỗi khối u được xác định bởi một vector 30 chiều. Mỗi khối u được gán một trong 2 nhãn: *benign (lành tính)* hoặc *malignant (ác tính)*. Bộ dữ liệu sẽ được xử lý bởi công cụ *Support Vector Classification* (*sklearn.svm.SVC*, thư viện *scikit-learn*) trong môi trường Python.

Mã nguồn:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
bcdata = pd.read_csv(directory)
X = bcdata.drop('class', axis=1).drop('id', axis=1)
y = bcdata['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Kết quả thu được, sử dụng SVM, với *train size=419, test size=150*:

SVC	Precision	Recall	Support
Lành tính	0.99	0.98	103
Ác tính	0.96	0.98	47
Trung bình/Tổng	0.98	0.98	150

Chú thích

Precision: Số phần tử được phân loại chính xác / Tổng số phần tử được phân loại.

$$Precision = \frac{true\ positive}{predicted\ positive}$$

Recall: Số phần tử nhãn X được phân loại chính xác / Tổng số phần tử nhãn X.

$$Recall = \frac{true\ positive}{actual\ positive}$$

Support: Số phần tử trong tập kiểm tra *(test data)* được phân loại là X.
Một mô hình phân loại tốt sẽ có Precision và Recall cao.

Với tiêu chuẩn đó, từ kết quả huấn luyện, ta dễ thấy SVM là một mô hình phân loại tốt.

Không chỉ vậy, khi bộ dữ liệu huấn luyện bị hạn chế, SVM tỏ ra ưu việt hơn các mô hình phân loại khác.

Kết quả thu được khi dữ liệu huấn luyện bị hạn chế, so sánh với thuật toán Perceptron, lấy giá trị trung bình của 1000 lần huấn luyện riêng biệt, với *train size=19, test size=550*:

SVC	Precision	Recall	Support
Trung bình/Tổng	0.92	0.92	550

Perceptron	Precision	Recall	Support
Trung bình/Tổng	0.63	0.68	550

Dễ thấy, dù bộ dữ liệu huấn luyện có bị hạn chế, kết quả phân loại bởi SVM vẫn có độ chính xác cao, trong khi Perceptron đưa ra kết quả có độ chính xác thấp hơn.

Tài liệu

[1] *https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners*.

[2] *Bài giảng PiMA*. 2018.

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[4] Trinh Thanh Deo, Le Van Luyen, Bui Xuan Hai, and Tran Ngoc Hoi. *Đại số Tuyến tính và Ứng dụng (Tập 1)*. Nhà Xuất bản Đại học Quốc gia Tp.HCM, 2009.

[5] Nguyen Huu Viet Hung. *Đại số tuyến tính*. Nhà Xuất bản Đại học Quốc gia Hà Nội, 2001.

[6] Witten D. Hastie T. Tibshiranie R. James, G. *An Introduction to Statistical Learning*. 2013.

[7] Vu Huu Tiep. *Machine learning cơ bản*.