Câu Chuyện Phân Loại Của Rô-bốt: Khám Phá Sức Mạnh Của SVM

Người Kể Chuyện AI

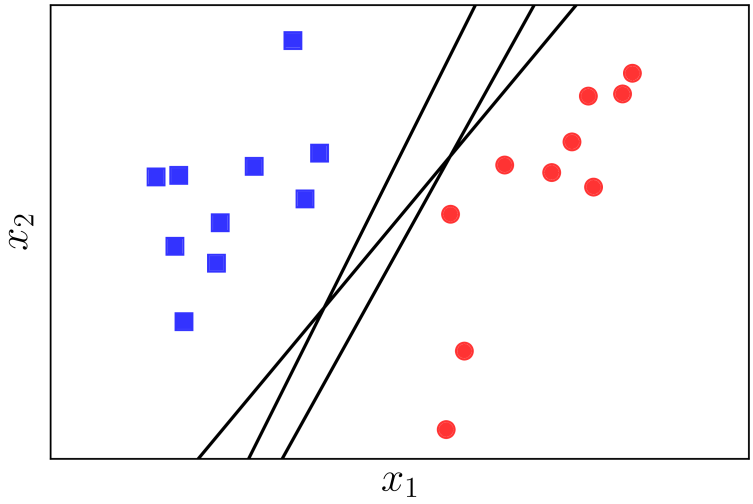
Ngày 12 tháng 4 năm 2025

Mục lục

1 Giới thiệu: Rô-bốt Muốn Sắp Xếp Đồ Chơi

Xin chào các bạn nhỏ! Lần trước chúng ta đã biết bạn rô-bốt học "nhìn"bằng Mạng Nơ-ron Tích Chập (CNN). Hôm nay, bạn rô-bốt của chúng ta lại có một nhiệm vụ mới: sắp xếp đồ chơi!

Tưởng tượng bạn ấy có một đống đồ chơi gồm bóng (màu đỏ) và các khối vuông (màu xanh) lẫn lộn trên sàn nhà. Bạn ấy muốn tìm một cách thật "thông minh"để kẻ một đường thẳng (hay một mặt phẳng nếu đồ chơi ở trong không gian 3D) chia sàn nhà thành hai khu vực riêng biệt, một bên chỉ có bóng đỏ, một bên chỉ có khối vuông xanh.



Hình 1: Có nhiều đường thẳng (mặt phẳng) có thể chia hai loại đồ chơi (điểm xanh và đỏ). Đường nào là tốt nhất?

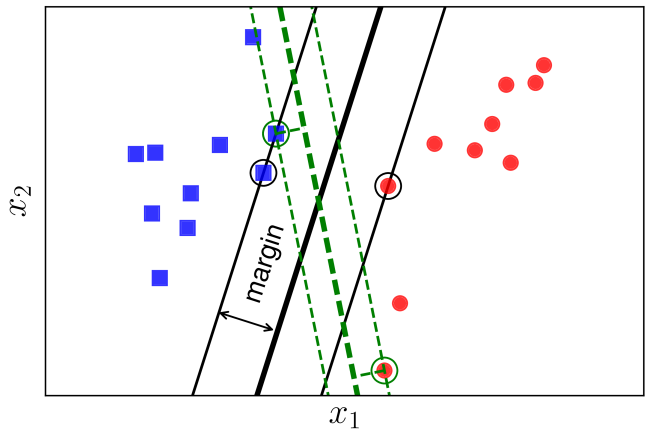
Có rất nhiều đường thẳng có thể làm được việc này (như trong Hình 1). Nhưng bạn rô-bốt muốn tìm ra đường thẳng **tốt nhất**. Đường thẳng tốt nhất là đường thẳng tạo ra một "con đường"hay "vùng an toàn"rộng nhất giữa hai nhóm đồ chơi. Phương pháp giúp rô-bốt tìm ra con đường rộng nhất này gọi là **Máy Vector Hỗ trợ (Support Vector Machine - SVM)**. Chúng ta cùng khám phá nhé!

2 SVM Biên Cứng (Hard Margin SVM): Khi Đồ Chơi Được Xếp Hoàn Hảo

2.1 Ý tưởng của đứa trẻ 5 tuổi: Tìm con đường rộng nhất

Đầu tiên, chúng ta hãy xét trường hợp dễ nhất: các quả bóng đỏ và khối vuông xanh nằm hoàn toàn tách biệt, không bị lẫn vào nhau. Bạn rô-bốt muốn tìm một cây thước (đường thẳng) đặt xuống sàn sao cho nó chia đúng hai nhóm đồ chơi.

Nhưng không chỉ chia đúng, bạn rô-bốt còn muốn cây thước này nằm ở vị trí mà khoảng cách từ cây thước đến món đồ chơi (bóng hoặc vuông) gần nó nhất ở mỗi bên là **lớn nhất có thể**. Khoảng trống ở hai bên cây thước này giống như một "con đường"hay "vùng đệm an toàn". SVM muốn tìm cây thước tạo ra con đường rộng nhất!



Hình 2: SVM tìm đường phân chia (nét liền đen) sao cho "con đường"(margin - khoảng cách giữa hai đường nét đứt) là rộng nhất. Các đồ chơi được khoanh tròn là quan trọng nhất.

Trong Hình 2, đường nét liền màu đen chính là cây thước (gọi là **siêu mặt phẳng - hyperplane**) mà SVM tìm được. Khoảng cách giữa hai đường nét đứt song song với nó chính là chiều rộng của "con đường"(gọi là **biên - margin**). SVM muốn làm cho cái biên này rộng tối đa.

2.2 Những món đồ chơi quan trọng nhất (Support Vectors)

Bạn có thấy những món đồ chơi được khoanh tròn trong Hình 2 không? Chúng là những món đồ chơi nằm **sát nhất** với mép của con đường (nằm trên các đường nét đứt). Chúng rất đặc biệt! Chúng được gọi là các **Vector Hỗ trợ (Support Vectors)**.

Tại sao chúng lại quan trọng? Bởi vì chính những món đồ chơi này quyết định vị trí của cây thước và độ rộng của con đường. Nếu bạn di chuyển một món đồ chơi nằm xa tít bên trong khu vực của nó, vị trí cây thước tốt nhất sẽ không thay đổi. Nhưng nếu bạn di chuyển một trong các Vector Hỗ trợ này, cây thước có thể sẽ phải dịch chuyển theo để đảm bảo con đường vẫn rộng nhất có thể.

SVM chỉ cần quan tâm đến các Vector Hỗ trợ này thôi! Đây là một điểm rất hay của SVM, nó giúp tiết kiệm tính toán khi có rất nhiều đồ chơi (dữ liệu).

2.3 Toán học một chút: Khoảng cách và Tối ưu hóa

Làm sao rô-bốt tính được khoảng cách và tìm đường rộng nhất?

Mô tả cây thước (Siêu mặt phẳng): Cây thước này được mô tả bằng một phương trình toán học. Nếu đồ chơi có vị trí là x (ví dụ $x = (x_1, x_2)$ trong không gian 2 chiều), phương trình của cây thước là:

$$w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b = \mathbf{w}^T \mathbf{x} + b = 0$$

Ở đây, $\mathbf{w}$ là một vector (giống như mũi tên chỉ hướng) cho biết "hướng nghiêng"của cây thước, còn b là một con số cho biết vị trí của cây thước.

Tính khoảng cách: Khoảng cách từ một món đồ chơi $\mathbf{x}_0$ đến cây thước được tính bằng công thức:

$$\text{distance} = \frac{|\mathbf{w}^T \mathbf{x}_0 + b|}{\|\mathbf{w}\|_2}$$

Dấu giá trị tuyệt đối |...| đảm bảo khoảng cách luôn dương. $\|\mathbf{w}\|_2 = \sqrt{w_1^2 + w_2^2 + \cdots + w_d^2}$ là "độ dài"của vector $\mathbf{w}$.

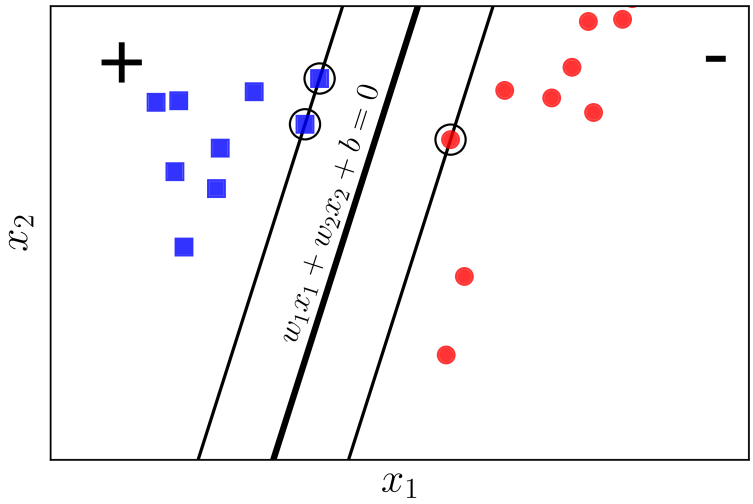
Tìm con đường rộng nhất (Tối ưu hóa): Rô-bốt muốn tìm $\mathbf{w}$ và b sao cho: 1. Tất cả bóng đỏ (gán nhãn $y_i = +1$) nằm về một phía: $\mathbf{w}^T\mathbf{x}_i + b \geq +1$. 2. Tất cả khối vuông xanh (gán nhãn $y_i = -1$) nằm về phía kia: $\mathbf{w}^T\mathbf{x}_i + b \leq -1$. 3. Độ rộng của con đường (margin) là lớn nhất. Độ rộng này bằng $\frac{2}{\|\mathbf{w}\|_2}$.

Vậy, muốn margin lớn nhất, thì $\|\mathbf{w}\|_2$ phải nhỏ nhất. Hay nói cách khác, $\|\mathbf{w}\|_2^2$ phải nhỏ nhất (bình phương lên cho dễ tính toán).

Bài toán của rô-bốt trở thành: Tìm $\mathbf{w}$ và b để **giảm thiểu** $\frac{1}{2}\|\mathbf{w}\|_2^2$ (nhân 1/2 cho đẹp thôi) với điều kiện là:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{cho mọi đồ chơi } i$$

Điều kiện này gộp cả hai trường hợp $y_i = +1$ và $y_i = -1$ lại. Nó đảm bảo tất cả đồ chơi phải nằm đúng phía và cách xa cây thước ít nhất một khoảng nào đó (nằm ngoài hoặc trên mép con đường).



Hình 3: Phân tích bài toán SVM: Tìm siêu mặt phẳng $\mathbf{w}^T\mathbf{x} + b = 0$ sao cho margin là lớn nhất. Các điểm support vector nằm trên $\mathbf{w}^T\mathbf{x} + b = \pm 1$.

Các Vector Hỗ trợ chính là những điểm mà $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$.

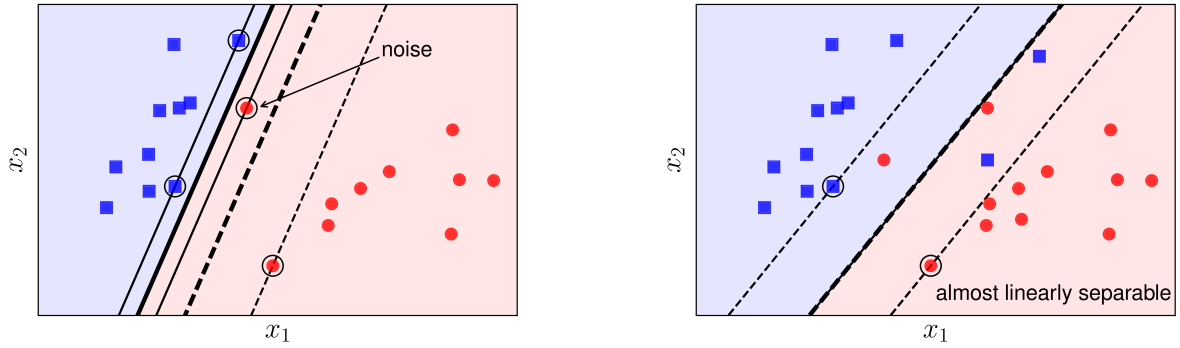
Tại sao lại là số 1? Chúng ta có thể chọn bất kỳ hằng số dương nào thay cho 1 ở vế phải (ví dụ số 5, số 10). Nếu thay $\mathbf{w}$ bằng $k\mathbf{w}$ và b bằng kb (với k dương), thì mặt phẳng phân chia không đổi, nhưng giá trị $y_i(\mathbf{w}^T\mathbf{x}_i + b)$ sẽ thay đổi. Người ta chọn số 1 cho tiện tính toán và để định nghĩa margin một cách rõ ràng.

3 SVM Biên Mềm (Soft Margin SVM): Khi Có Vài Món Đồ Chơi Lộn Xộn

3.1 Ý tưởng của đứa trẻ 5 tuổi: Cho phép một chút lộn xộn

Đời không phải lúc nào cũng hoàn hảo! Đôi khi, có một vài quả bóng đỏ lại nằm hơi lấn sang khu vực của khối vuông xanh, hoặc ngược lại. Hoặc có thể hai nhóm đồ chơi bị trộn lẫn một

chút ở vùng ranh giới.



Hình 4: a) Có nhiễu (điểm đỏ gần nhóm xanh). b) Hai nhóm hơi chồng lấn. SVM Biên Cứng sẽ không hoạt động tốt hoặc không tìm được nghiệm.

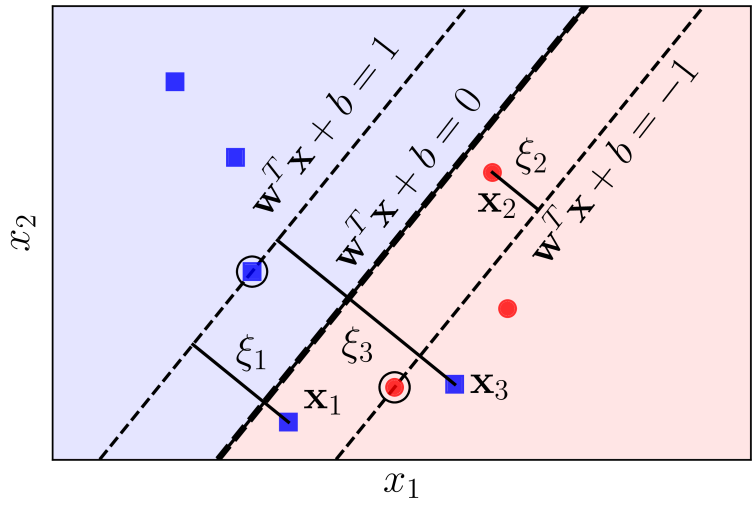
Trong những trường hợp này (Hình 4), nếu rô-bốt cứ cố tìm một "con đường"hoàn hảo (Hard Margin), bạn ấy có thể sẽ: 1. Tạo ra một con đường rất hẹp (vì cố né điểm nhiễu ở hình a). 2. Hoặc không thể tìm được đường nào cả (vì đồ chơi bị trộn lẫn ở hình b).

Giải pháp là gì? Rô-bốt cần phải "mềm dẻo"hơn một chút. Bạn ấy cho phép một vài món đồ chơi được phép: * Nằm bên trong "con đường an toàn". * Thậm chí nằm hẳn sang phía bên kia của cây thước (bị xếp nhầm!).

Đây chính là ý tưởng của **SVM Biên Mềm (Soft Margin SVM)**. Nó cố gắng tìm con đường rộng nhất có thể, nhưng đồng thời cũng cố gắng **giảm thiểu mức độ lộn xộn** (số lượng đồ chơi bị đặt sai chỗ và mức độ sai của chúng).

3.2 Toán học một chút: Biến bù và Cái giá phải trả

Để đo mức độ "lộn xộn"của từng món đồ chơi, rô-bốt dùng một con số gọi là **biến bù (slack variable)**, ký hiệu là ξ_i (đọc là "xi"). * Nếu món đồ chơi i nằm đúng vị trí và ở ngoài con đường an toàn, thì $\xi_i = 0$ (không lộn xộn). * Nếu món đồ chơi i nằm bên trong con đường nhưng vẫn đúng phía, thì $0 < \xi_i \leq 1$. * Nếu món đồ chơi i nằm sai phía của cây thước, thì $\xi_i > 1$.



Hình 5: Giới thiệu các biến bù ξ_n. Điểm trong vùng an toàn có $\xi_n = 0$. Điểm x_2 nằm trong biên có $0 < \xi_2 < 1$. Điểm x_1, x_3 bị phân loại sai có $\xi_1, \xi_3 > 1$.

Điều kiện $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ bây giờ được nới lỏng thành:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \text{với } \xi_i \geq 0$$

Bây giờ, mục tiêu của rô-bốt là tìm $\mathbf{w}$, b và cả các ξ_i để: * Giữ cho con đường vẫn rộng (tức là $\|\mathbf{w}\|_2^2$ nhỏ). * Giữ cho tổng mức độ lộn xộn $\sum \xi_i$ là nhỏ nhất.

Bài toán tối ưu trở thành: **Giảm thiểu**

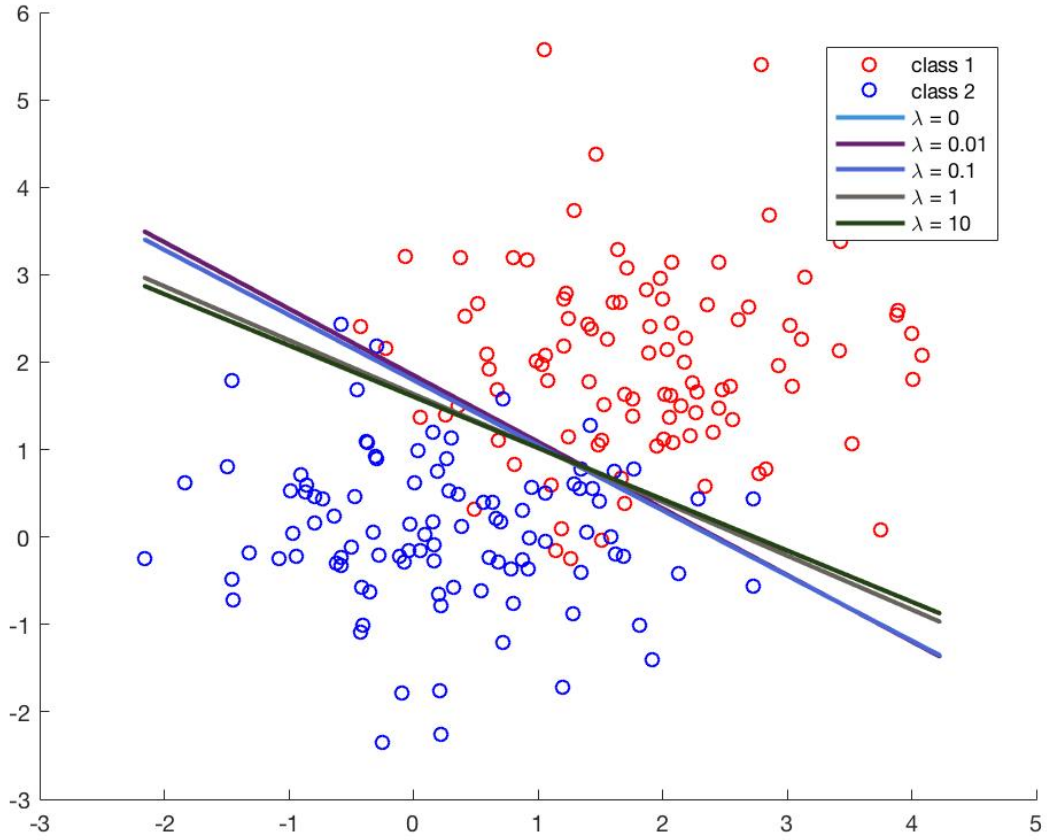
$$\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi_i$$

với các điều kiện:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{và} \quad \xi_i \geq 0 \quad \text{cho mọi } i$$

Ở đây xuất hiện thêm một nhân vật mới là C. Đây là một hằng số dương do chúng ta chọn, gọi là **tham số điều chuẩn (regularization parameter)** hay **tham số phạt (penalty parameter)**. * Nếu C rất lớn: Rô-bốt rất "ghét"sự lộn xộn ($\sum \xi_i$ phải cực nhỏ). Nó sẽ cố gắng giảm thiểu ξ_i hết mức có thể, giống như Hard Margin SVM. * Nếu C nhỏ: Rô-bốt "thoáng"hơn với sự lộn xộn. Nó có thể chấp nhận một vài ξ_i lớn để đổi lấy một con đường rộng hơn (tức $\|\mathbf{w}\|_2^2$ nhỏ hơn).

Việc chọn C là một sự đánh đổi giữa việc có một con đường rộng và việc có ít đồ chơi bị sai vị trí.



Hình 6: *Ở đây C giống với lambda nhé.*Ảnh hưởng của tham số C. Khi C tăng (ví dụ từ 0.1 đến 100), biên (margin) có xu hướng hẹp lại để cố gắng phân loại đúng nhiều điểm hơn.

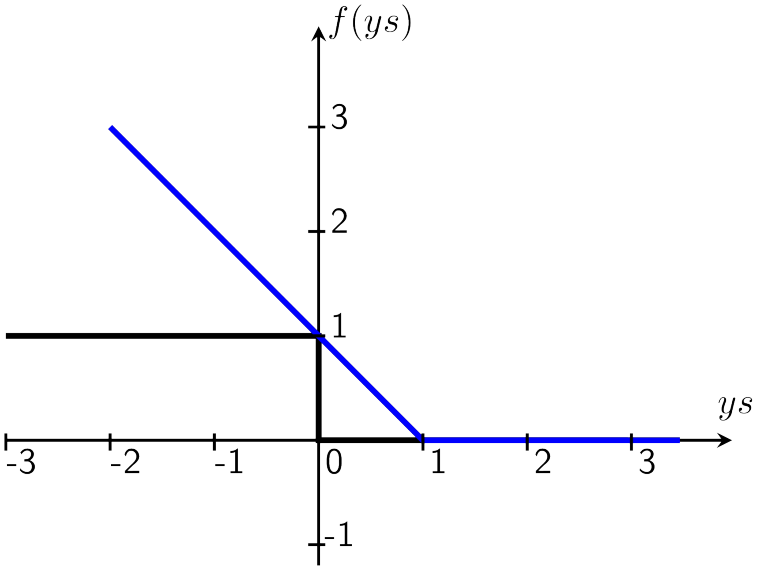
3.3 Hàm Mất mát Hinge Loss

Có một cách nhìn khác về bài toán Soft Margin SVM là thông qua **Hàm mất mát (Loss Function)**. Hàm mất mát đo "độ không hài lòng"của mô hình với dự đoán của nó. SVM sử dụng một hàm mất mát đặc biệt gọi là **Hinge Loss**.

Với mỗi món đồ chơi i, rô-bốt tính một "điểm số"là $z_i = \mathbf{w}^T\mathbf{x}_i + b$. Nhãn đúng là y_i. Hinge loss được tính là:

$$L_i = \max(0, 1 - y_i z_i)$$

Ý nghĩa là: * Nếu $y_i z_i \geq 1$ (đồ chơi nằm đúng phía và đủ xa cây thước - tức là nằm ngoài hoặc trên mép biên an toàn), thì loss bằng 0 (rô-bốt hoàn toàn hài lòng). * Nếu $y_i z_i < 1$ (đồ chơi nằm sai phía, hoặc đúng phía nhưng trong vùng biên), thì loss bằng $1 - y_i z_i$, là một số dương. Càng sai nhiều ($y_i z_i$ càng nhỏ), loss càng lớn.



Hình 7: Đồ thị Hinge loss (màu xanh). Trục hoành là $y \times z$. Nếu $yz \geq 1$, loss bằng 0. Nếu $yz < 1$, loss tăng tuyến tính.

Bài toán tối ưu SVM có thể viết lại thành: **Giảm thiểu**

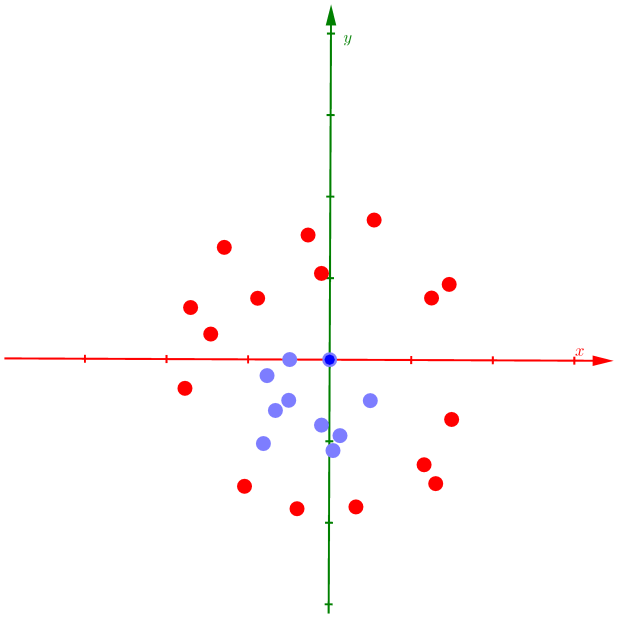
$$\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)) + \lambda \|\mathbf{w}\|_2^2$$

(Ở đây λ là tham số điều chuẩn, liên quan đến C theo kiểu $\lambda \approx 1/C$). Phần đầu là tổng data loss (tính bằng hinge loss), phần sau là regularization loss (phạt nếu $\mathbf{w}$ quá lớn, tức margin quá hẹp).

4 SVM Hạt nhân (Kernel SVM): Phép Thuật Cho Đồ Chơi Không Thể Chia Bằng Đường Thẳng

4.1 Ý tưởng của đứa trẻ 5 tuổi: Đeo kính ma thuật

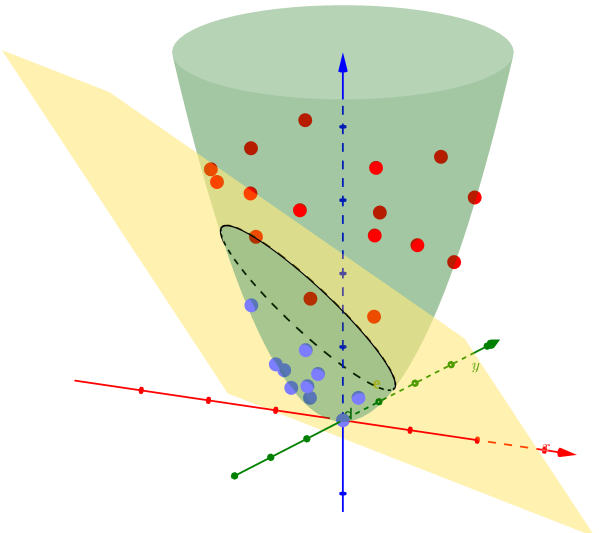
Đến giờ, rô-bốt của chúng ta chỉ dùng cây thước thẳng để chia đồ chơi. Nhưng nếu đồ chơi được xếp theo hình phức tạp thì sao? Ví dụ, các khối vuông xanh nằm thành một vòng tròn ở giữa, còn bóng đỏ nằm bao quanh bên ngoài? Rõ ràng là không có cây thước thẳng nào chia được hai nhóm này cả!



Hình 8: Dữ liệu không thể phân chia bằng đường thẳng (ví dụ: điểm xanh ở giữa, điểm đỏ bao quanh).

Làm thế nào bây giờ? Rô-bốt cần một phép thuật! Phép thuật đó gọi là **Thủ thuật Hạt nhân (Kernel Trick)**.

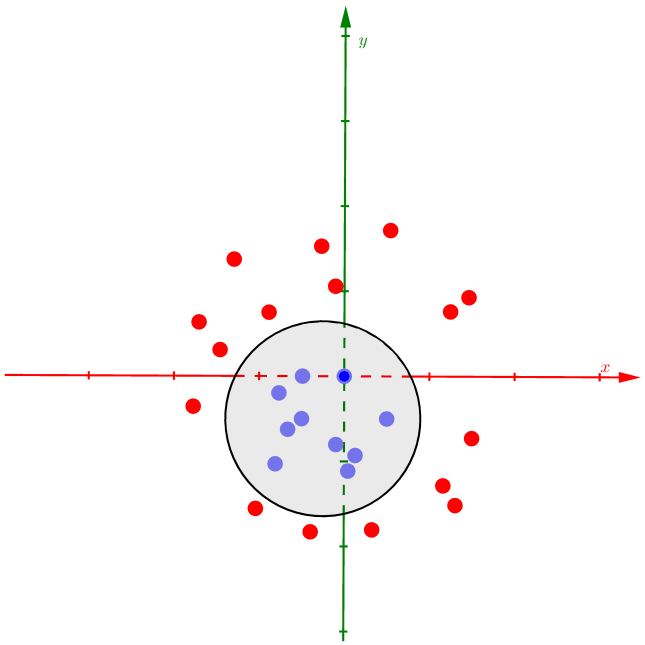
Hãy tưởng tượng rô-bốt đeo một cặp "kính ma thuật". Khi nhìn qua cặp kính này, vị trí của các món đồ chơi dường như thay đổi. Ví dụ, những món đồ chơi ở gần tâm sàn nhà có thể bị "nâng"lên cao hơn những món đồ chơi ở xa. Sau khi "biến hình"trong không gian mới này, điều kỳ diệu xảy ra: rô-bốt **lại có thể** dùng một cây thước thẳng (hoặc một mặt phẳng) để chia hai nhóm đồ chơi một cách hoàn hảo (hoặc gần hoàn hảo)!



Hình 9: Ý tưởng Kernel: Dữ liệu gốc (trái) không chia được bằng đường thẳng. Sau khi "biến đổi"(nhìn qua kính ma thuật - giữa), dữ liệu trở nên tách biệt tuyến tính trong không gian mới (phải) và có thể chia bằng mặt phẳng.

Sau khi tìm được mặt phẳng chia trong không gian mới, rô-bốt "cởi kính"ra, mặt phẳng đó

khi nhìn lại trong không gian cũ sẽ trở thành một đường cong (hoặc một hình phức tạp) chia đúng hai nhóm đồ chơi ban đầu!



Hình 10: Đường phân chia cong tìm được trong không gian ban đầu sau khi áp dụng Kernel SVM.

4.2 Toán học một chút: Hàm Hạt nhân và Không gian mới

Cặp "kính ma thuật"chính là một phép biến đổi toán học, gọi là hàm ánh xạ $\Phi(\mathbf{x})$. Nó đưa điểm dữ liệu $\mathbf{x}$ từ không gian ban đầu sang một không gian mới (thường có số chiều cao hơn nhiều, thậm chí vô hạn chiều!).

Trong không gian mới này, bài toán SVM lại trở thành tìm siêu mặt phẳng phân chia các điểm $\Phi(\mathbf{x}_i)$. Vấn đề là, việc tính toán $\Phi(\mathbf{x})$ trực tiếp có thể rất phức tạp hoặc không thể thực hiện được vì số chiều quá lớn.

Đây là lúc "thủ thuật"phát huy tác dụng. Khi giải bài toán SVM (dù là biên cứng hay biên mềm), người ta thường giải bài toán **đối ngẫu (dual problem)**. Điều thú vị là trong bài toán đối ngẫu và cả trong công thức dự đoán nhãn cho điểm mới, các điểm dữ liệu $\mathbf{x}_i$ chỉ xuất hiện dưới dạng **tích vô hướng (dot product)** $\mathbf{x}_i^T\mathbf{x}_j$.

Khi chuyển sang không gian mới, chúng ta chỉ cần tính tích vô hướng $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. Thủ thuật hạt nhân nói rằng: chúng ta có thể tính giá trị tích vô hướng này trong không gian mới mà **không cần tính** $\Phi(\mathbf{x})$ một cách tường minh! Chúng ta chỉ cần định nghĩa một hàm gọi là **Hàm Hạt nhân (Kernel Function)** $k(\mathbf{x}_i, \mathbf{x}_j)$ sao cho:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$$

Hàm k này tính toán trực tiếp từ $\mathbf{x}_i$ và $\mathbf{x}_j$ trong không gian ban đầu, nhưng lại cho kết quả bằng với tích vô hướng trong không gian mới.

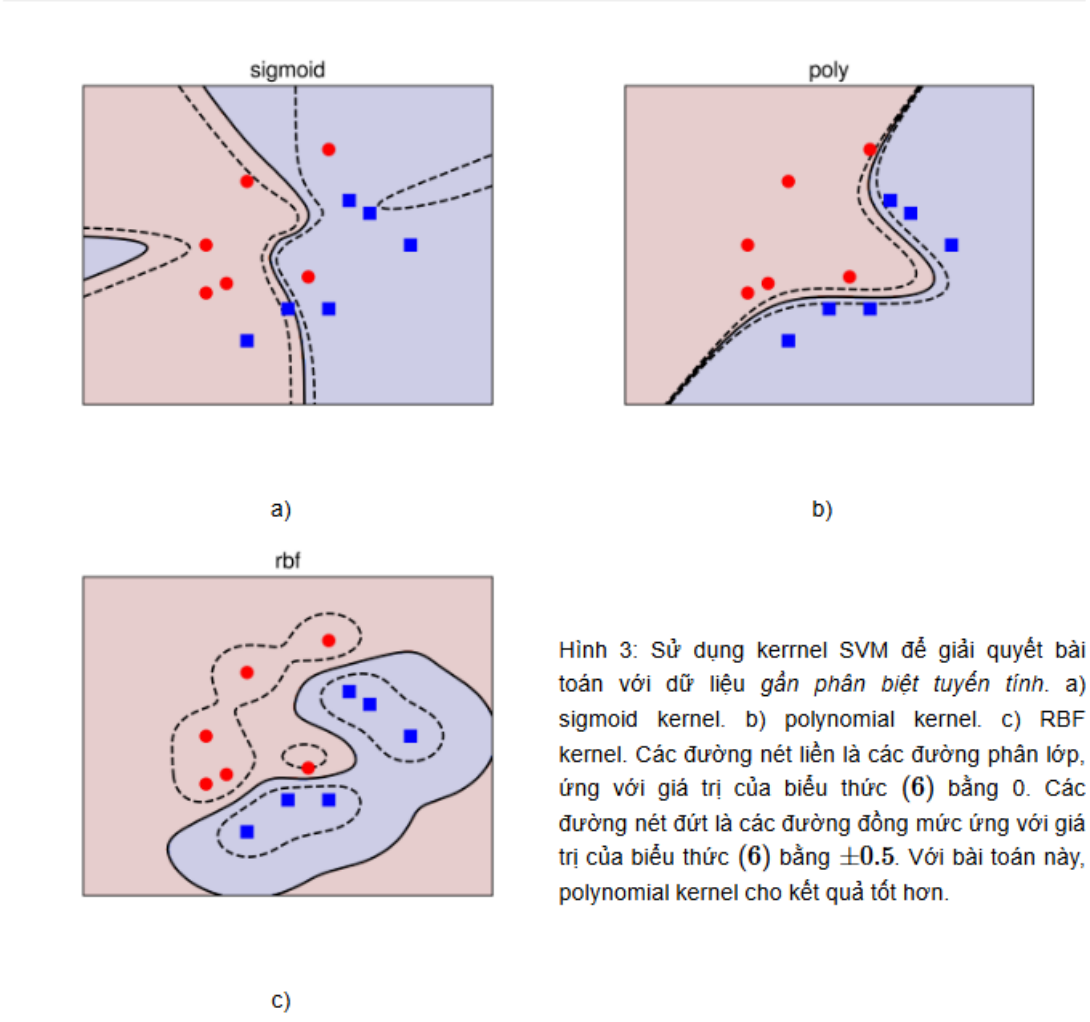
Tại sao lại kỳ diệu vậy? Lấy lại ví dụ biến đổi $\mathbf{x} = [x_1, x_2]^T$ thành $\Phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2,$
Tích vô hướng là:

$$\Phi(\mathbf{x})^T\Phi(\mathbf{z}) = 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 = (1 + x_1z_1 + x_2z_2)^2$$

Chúng ta thấy rằng: $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T\mathbf{z})^2$. Tính $k(\mathbf{x}, \mathbf{z})$ dễ hơn nhiều so với tính $\Phi(\mathbf{x}), \Phi(\mathbf{z})$ rồi mới nhân vô hướng!

4.3 Các loại "Kính Ma Thuật"phổ biến (Các hàm Kernel)

Có nhiều loại hàm kernel khác nhau, tương ứng với các cách "biến hình"dữ liệu khác nhau:
1. **Linear Kernel:** $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$. Đây thực chất là không biến đổi gì cả, tương đương SVM tuyến tính thông thường.
2. **Polynomial Kernel:** $k(\mathbf{x}, \mathbf{z}) = (\gamma\mathbf{x}^T\mathbf{z} + r)^d$. Tạo ra các đường phân chia đa thức bậc d. Tham số γ, r, d cần được chọn.
3. **Radial Basis Function (RBF) Kernel (hay Gaussian Kernel):** $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|^2)$. Đây là kernel mạnh mẽ và phổ biến nhất. Nó dựa trên khoảng cách giữa hai điểm. Nếu hai điểm càng gần nhau, giá trị kernel càng lớn (gần 1). Nếu càng xa nhau, giá trị kernel càng gần 0. Tham số γ kiểm soát độ "ảnh hưởng"của một điểm. Kernel này có thể tạo ra các đường phân chia rất phức tạp.
4. **Sigmoid Kernel:** $k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma\mathbf{x}^T\mathbf{z} + r)$. Ít được dùng hơn.



Hình 3: Sử dụng kerrnel SVM để giải quyết bài toán với dữ liệu *gần phân biệt tuyến tính*. a) sigmoid kernel. b) polynomial kernel. c) RBF kernel. Các đường nét liền là các đường phân lớp, ứng với giá trị của biểu thức **(6)** bằng 0. Các đường nét đứt là các đường đồng mức ứng với giá trị của biểu thức **(6)** bằng $\pm$**0.5**. Với bài toán này, polynomial kernel cho kết quả tốt hơn.

Hình 11: Ví dụ về các đường phân chia tạo bởi các kernel khác nhau (Sigmoid, Poly, RBF) cho bài toán XOR (không tách biệt tuyến tính).

Việc chọn kernel và các tham số của nó (như C, γ, d, r) thường dựa vào thử nghiệm hoặc cross-validation.

5 SVM Đa Lớp (Multi-class SVM): Sắp Xếp Nhiều Loại Đồ Chơi

5.1 Ý tưởng của đứa trẻ 5 tuổi: Nhiều hộp đồ chơi

Bạn rô-bốt giờ không chỉ có bóng đỏ và khối vuông xanh, mà còn có cả ô tô vàng, búp bê hồng, lego xanh lá... Tóm lại là có nhiều hơn hai loại đồ chơi (nhiều **lớp - classes**). Làm sao để phân loại chúng?

Một cách đơn giản là bạn rô-bốt sẽ học riêng từng cặp: học cách phân biệt bóng đỏ với khối vuông, rồi học cách phân biệt bóng đỏ với ô tô vàng, rồi khối vuông với ô tô vàng... (Cách này gọi là **One-vs-One**). Hoặc bạn ấy học cách phân biệt một loại với tất cả các loại còn lại: học cách nhận ra bóng đỏ so với "không phải bóng đỏ", rồi học cách nhận ra khối vuông so với "không phải khối vuông"... (Cách này gọi là **One-vs-Rest**).

Tuy nhiên, có một cách khác trực tiếp hơn, giống như cách Softmax Regression hoạt động, nhưng sử dụng ý tưởng "biên an toàn"của SVM.

5.2 Toán học một chút: Hinge Loss Đa Lớp

Với mỗi món đồ chơi $\mathbf{x}_n$, rô-bốt sẽ tính một "điểm số"cho **mỗi loại** đồ chơi có thể có. Gọi z_{nj} là điểm số của đồ chơi n cho loại j. Gọi y_n là loại đồ chơi đúng của món đồ chơi n.

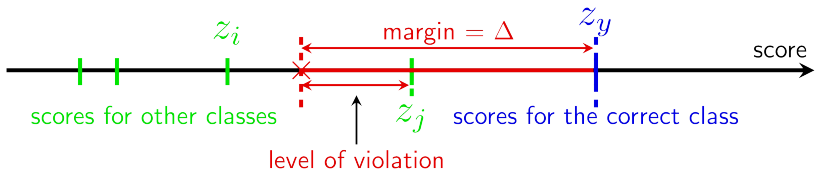
Rô-bốt muốn rằng điểm số của loại đúng (z_{n,y_n}) phải **cao hơn** điểm số của bất kỳ loại sai nào khác (z_{nj} với $j \neq y_n$) ít nhất là một khoảng an toàn Δ (thường chọn $\Delta = 1$).

Nếu điều này không được thỏa mãn, tức là có một loại sai j mà $z_{nj} \geq z_{n,y_n} - \Delta$, thì rô-bốt sẽ bị "phạt". Mức phạt (loss) cho loại sai j đó là:

$$\max(0, z_{nj} - z_{n,y_n} + \Delta)$$

Tổng loss cho món đồ chơi n là tổng các mức phạt này trên tất cả các loại sai j:

$$L_n = \sum_{j \neq y_n} \max(0, z_{nj} - z_{n,y_n} + \Delta)$$



Hình 12: Mô tả Hinge loss đa lớp. Muốn score của lớp đúng (màu lam) cao hơn score của các lớp sai (màu lục) ít nhất một khoảng margin Δ (màu đỏ).

Điểm số z_{nj} được tính bằng $z_{nj} = \mathbf{w}_j^T \mathbf{x}_n$, trong đó $\mathbf{w}_j$ là vector trọng số (giống như "hình mẫu") cho loại đồ chơi j. Tất cả các vector $\mathbf{w}_j$ được xếp thành các cột của ma trận $\mathbf{W}$. Vậy $\mathbf{z}_n = \mathbf{W}^T \mathbf{x}_n$.

Bài toán tối ưu tổng thể là tìm ma trận $\mathbf{W}$ để **giảm thiểu** tổng loss trên tất cả đồ chơi cộng với phần điều chuẩn (regularization) để tránh $\mathbf{W}$ quá lớn:

$$L = \frac{1}{N} \sum_{n=1}^{N} L_n + \frac{\lambda}{2} \|\mathbf{W}\|_F^2$$

Ở đây $\|\mathbf{W}\|_F^2$ là chuẩn Frobenius (tổng bình phương tất cả các phần tử của $\mathbf{W}$), và λ là tham số điều chuẩn.

Bài toán này có thể giải bằng các phương pháp dựa trên gradient descent, tương tự như huấn luyện mạng nơ-ron.

Hình 13: Ví dụ về các vector trọng số $\mathbf{w}_j$ học được cho 10 lớp trong bộ dữ liệu CIFAR-10. Mỗi $\mathbf{w}_j$ trông giống như một "hình ảnh trung bình"của lớp đó.

6 Kết Luận: Rô-bốt Sắp Xếp Siêu Đẳng!

Vậy là bạn rô-bốt của chúng ta đã học thêm một kỹ năng mới rất lợi hại là SVM! 1. Bạn ấy biết tìm "con đường"rộng nhất để chia hai nhóm đồ chơi xếp ngay ngắn (**Hard Margin SVM**). 2. Bạn ấy biết cách xử lý khi đồ chơi hơi lộn xộn hoặc bị nhiễu, bằng cách cho phép một chút sai sót có kiểm soát (**Soft Margin SVM**). 3. Bạn ấy biết dùng "kính ma thuật"để xử lý những nhóm đồ chơi không thể chia bằng đường thẳng (**Kernel SVM**). 4. Và cuối cùng, bạn ấy còn biết cách phân loại nhiều hơn hai loại đồ chơi cùng lúc (**Multi-class SVM**).

SVM là một công cụ rất mạnh mẽ trong Machine Learning, đặc biệt hiệu quả với dữ liệu có số chiều cao và khi chúng ta muốn tìm ra đường phân chia rõ ràng nhất. Mặc dù các mạng nơ-ron sâu đang rất phổ biến, SVM vẫn có vị trí quan trọng và thường cho kết quả rất tốt trong nhiều bài toán!