MACHINE LEARNING

Cao Văn Chung
cvanchung@hus.edu.vn
Informatics Dept., MIM, HUS, VNU Hanoi

Support Vector Machine (Part 1: Hard Margin)

Hyperplan and distance to hyperplan

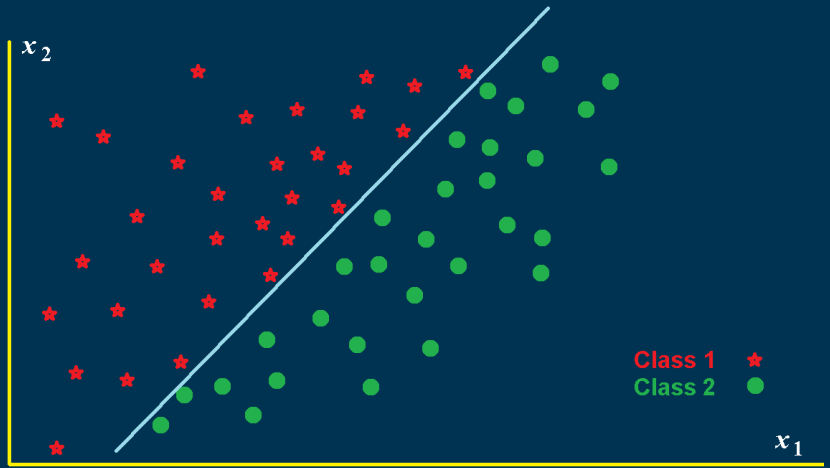
- Ta xét trong không gian 2 chiều để dễ minh họa.
- Khoảng cách từ điểm $p^* = (x_1^*, x_2^*)$ đến đường thẳng $D : w_0 + w_1 x_1 + w_2 x_2 = 0$ xác định bởi

$$\text{dist}(p^*, D) = \frac{|w_0 + w_1 x_1^* + w_2 x_2^*|}{\sqrt{w_1^2 + w_2^2}}.$$

- Xa hơn, nếu nếu bỏ dấu trị tuyệt đối ở tử số, ta có thể xác định được p^* nằm về phía nào của D
  - Tất cả những điểm (x_1, x_2) để $w_0 + w_1 x_1 + w_2 x_2 > 0$ sẽ nằm cùng phía, ta gọi là phía dương;
  - Tất cả những điểm (x_1, x_2) để $w_0 + w_1 x_1 + w_2 x_2 < 0$ sẽ nằm cùng phía, ta gọi là phía âm.

Linearly Separable Classification

Hình: Nearly linearly separable set



Hyperplan and distance to hyperplan

- Trong không gian nhiều chiều $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, siêu phẳng (*hyperplane*) P có phương trình dạng $w_0 + w_1 x_1 + \ldots + w_d x_d = 0$ hay $w_0 + \mathbf{w}^T \mathbf{x} = 0$.
- Khoảng cách từ điểm $\mathbf{x}^*$ đến siêu phẳng P là

$$\text{dist}(x^*, P) = \frac{|w_0 + w_1 x_1^* + \ldots + w_d x_d^*|}{\sqrt{w_1^2 + \ldots + w_2^2}} = \frac{|w_0 + \mathbf{w}^T \mathbf{x}^*|}{\|\mathbf{w}\|_2^2}.$$

- Ngoài ra, P cũng tách không gian thành thành các tập điểm phía dương với $w_0 + \mathbf{w}^T \mathbf{x} > 0$ và phía âm, với $w_0 + \mathbf{w}^T \mathbf{x} < 0$.
- **Khoảng cách từ tập điểm đến siêu phẳng**: Khoảng cách từ tập (hữu hạn) $\mathbf{X}$ đến siêu phẳng P là

$$\text{dist}(\mathbf{X}, P) = \min_{x \in \mathbf{X}} \text{dist}(x, P).$$

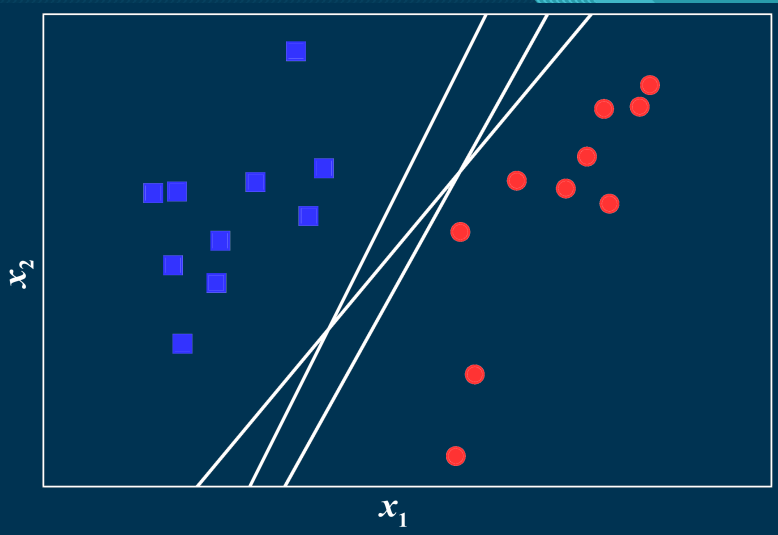
Binary Classification

- Nhắc lại bài toán: Giả sử tập dữ liệu được mô tả như các điểm (vector) trong không gian d, và thuộc về 02 lớp (02 loại: Class 1 và Class 2) phân biệt.
- Ngoài ra, giả thiết hai lớp dữ liệu là tách được tuyến tính (*linearly separable*): tồn tại một siêu phẳng phân chia chính xác hai lớp đó.
- **Bài toán phân lớp nhị phân**: Tìm siêu phẳng $w_0 + \mathbf{w}^T\mathbf{x} = 0$ phân chia các lớp - tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại

$$\forall \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \text{Class } c (c = 1, 2) : \left(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}\right)\left(w_0 + \mathbf{w}^T\mathbf{x}^{(j)}\right) > 0 \quad \text{và}$$
$$\forall \mathbf{x}^{(i)} \in \text{Class } 1; \forall \mathbf{x}^{(j)} \in \text{Class } 2 : \left(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}\right)\left(w_0 + \mathbf{w}^T\mathbf{x}^{(j)}\right) < 0.$$

- **Vấn đề**: Có thể tồn tại vô số nghiệm!

Binary Classification: Linearly separable classes



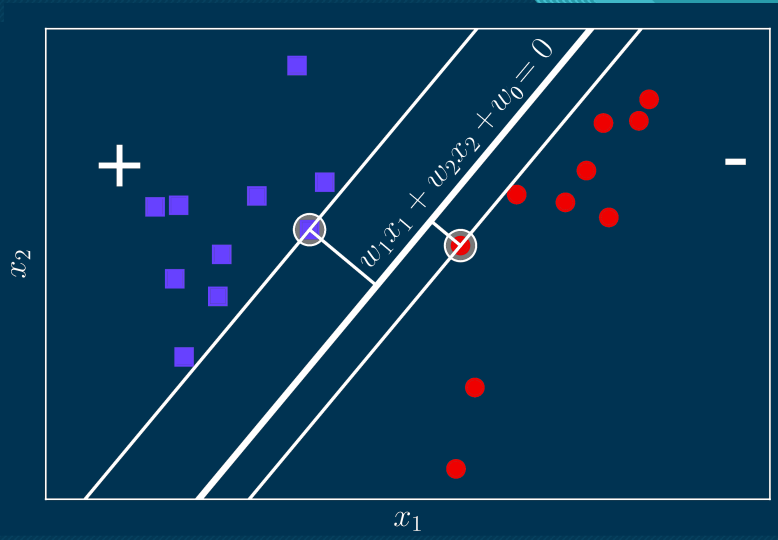
Binary Classification

- **Bài toán mới được đặt ra**: trong các siêu phẳng phân chia đó, tìm siêu phẳng phân chia tối ưu theo một tiêu chuẩn nào đó.
- Một nhóm phương pháp đi tìm siêu phẳng tối ưu theo những tiêu chuẩn sau
  1. **Sự cân đối - hay phân chia công bằng**: Siêu phẳng P tối ưu sẽ phải "cách đều" các tập điểm của mỗi phân lớp.

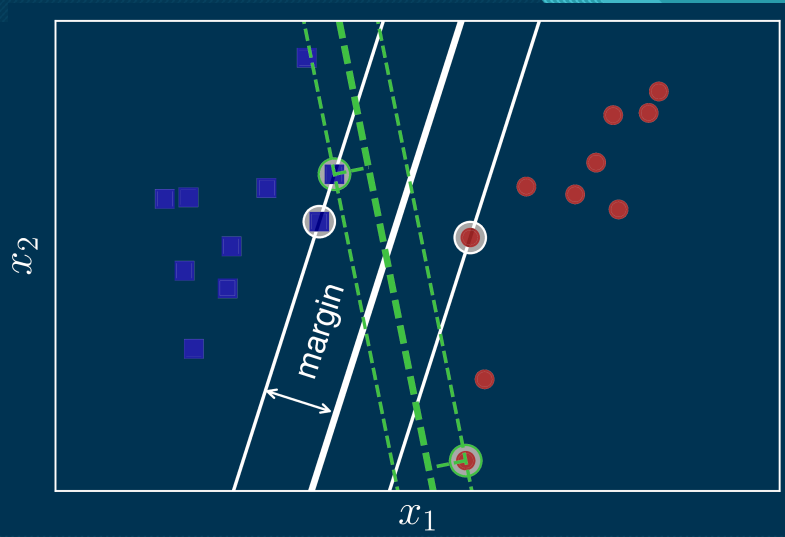
     $$\text{dist}(\textbf{Class1}, P) = \text{dist}(\textbf{Class2}, P).$$

     Khoảng cách này được gọi là *margin* - lề.
  2. **Phân tách rõ ràng nhất**: Khoảng cách lề là lớn nhất.
- Ta có thể thấy, với hai tiêu chí trên, siêu phẳng tối ưu sẽ phân chia hai lớp rạch ròi hơn.

Binary Classification: Linearly separable classes



Binary Classification: Linearly separable classes



Binary Classification

Việc tìm siêu phẳng cách đều 02 tập cho trước và khoảng cách này cực đại cần giải quyết các vấn đề

- Cực đại hóa khoảng cách từ siêu phẳng đến một (một số) điểm của hai lớp dữ liệu;
- Các ràng buộc sau cần thỏa mãn
  - Khoảng cách từ các điểm đó đến siêu phẳng là nhỏ nhất trong số các điểm cùng phân lớp;
  - Khoảng cách từ siêu phẳng đến hai nhóm thuộc hai phân lớp là bằng nhau.

Đây là bài toán tối ưu có ràng buộc (trên hình thức, ràng buộc cũng là một bài toán tối ưu, tuy nhiên tập dữ liệu thực tế hữu hạn, do đó ta không xét bài toán này như là dạng tối ưu hai mức - bilevel optimization).

Some more auxiliary knowledge

Để giải bài toán tối ưu có ràng buộc phát sinh, ta cần các kiến thức sau (học viên tự trang bị)

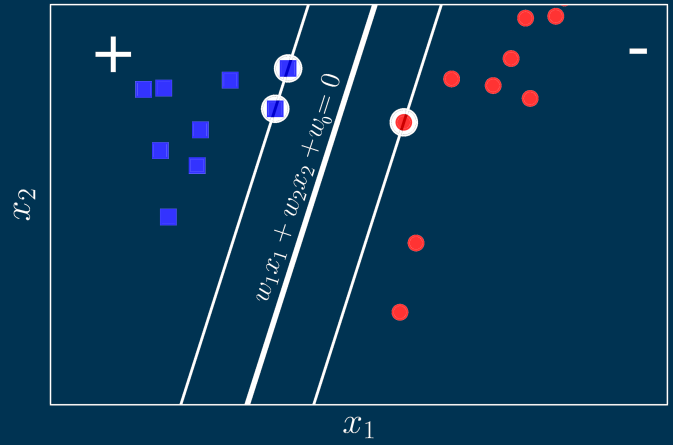
- Tối ưu lồi
- Bài toán đối ngẫu (Duality)
- Phương pháp Nhân tử Lagrange
- Bài toán đối ngẫu Lagrange (The Lagrange dual problem)
  - Điều kiện tối ưu Karush–Kuhn–Tucker (KKT)
  - Điều kiện Slater

Support Vector Machine (SVM) đi tìm siêu phẳng phân chia sao cho margin là lớn nhất, do đó còn được gọi là *Maximum Margin Classifier*.

Some more auxiliary knowledge

- Cho training set: $\mathbf{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ với đầu vào $\mathbf{x}_i \in \mathbb{R}^d$ và đầu ra hay nhãn tương ứng y_i. $d \geq 1$ là số chiều và $N \geq 1$ là số điểm dữ liệu.
- Ta xét bài toán phân loại nhị phân, tức đầu ra thuộc 02 lớp: $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2).
- Giả sử tập dữ liệu là tách được tuyến tính - Linearly Separable.
- Để dễ hình dung, chúng ta sẽ minh họa và phân tích trong trường hợp $d = 2$ - Dữ liệu đầu vào thuộc $\mathbb{R}^2$.
- Sau đó mở rộng kết quả cho trường hợp nhiều chiều tổng quát.

SVD's optimization problem



SMV's optimization problem

- Trong $\mathbb{R}^2$, phương trình đường phân chia có dạng

$$\mathbf{w}^T\mathbf{x} + w_0 = 0 \quad \Leftrightarrow \quad w_0 + w_1x_1 + w_2x_2 = 0.$$

  Trong hình minh họa, class 1 là các điểm vuông-xanh và class -1 là các điểm tròn-đỏ.
- Ta luôn có thể giả thiết class 1 chứa các điểm $\mathbf{x}$ sao cho $\mathbf{w}^T\mathbf{x} + w_0 \geq 0$ và class -1 chứa các điểm $\mathbf{x}$ sao cho $\mathbf{w}^T\mathbf{x} + w_0 < 0$.
  Thật vậy, ngược lại ta chỉ cần đổi dấu $\mathbf{w}$.
- Chú ý rằng bài toán xuất phát từ mô hình sẽ cần đi tìm các hệ số $\mathbf{w}$, w_0.

SMV's optimization problem

- **Chú ý:** Theo cách đặt, khoảng cách từ cặp dữ liệu $(\mathbf{x}_n, y_n)$ bất kỳ đến siêu phẳng phân chia là

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + w_0)}{\|\mathbf{w}\|_2}.$$

  Điều này là vì y_n cùng dấu với $\mathbf{w}^T\mathbf{x}_n + w_0$ theo cách đặt của chúng ta, nên $y_n(\mathbf{w}^T\mathbf{x}_n + w_0) = |\mathbf{w}^T\mathbf{x}_n + w_0|$.

- Với mặt phần chia như trên, *margin* được tính là khoảng cách gần nhất từ một điểm tới mặt đó (bất kể điểm nào trong hai classes):

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + w_0)}{\|\mathbf{w}\|_2}.$$

SMV's optimization problem

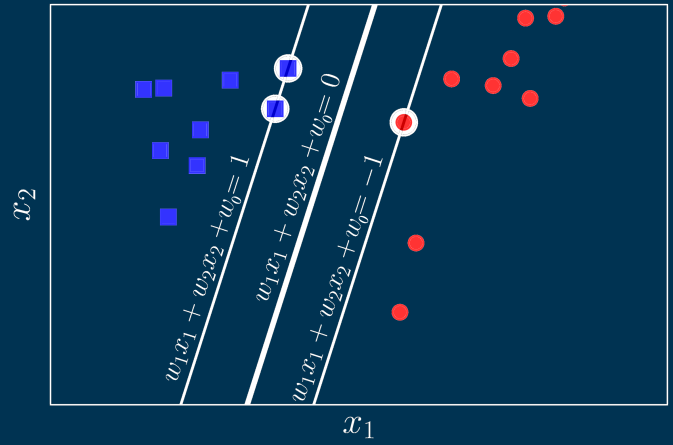
- Bài toán tối ưu trong SVM trở thành tìm $\mathbf{w}$, w_0 cực đại hóa margin

$$\begin{aligned}(\mathbf{w}, w_0) &= \arg\max_{\mathbf{w},w_0} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + w_0)}{\|\mathbf{w}\|_2} \right\} \\ &= \arg\max_{\mathbf{w},w_0} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \right\}\end{aligned} \tag{1}$$

- Việc giải trực tiếp bài toán này quá phức tạp. Ta cần tìm cách để đưa nó về bài toán đơn giản hơn.
- **Nhận xét**: Với hằng số $k > 0$, nếu nhân (chia) các hệ số $\mathbf{w}, w_0$ với k thì đường thẳng $d : \mathbf{w}^T\mathbf{x} + w_0 =$ không thay đổi, tức margin không đổi. Vậy với tất cả các điểm x_n gần d nhất, ta có thể giả thiết

$$y_n(\mathbf{w}^T\mathbf{x}_n + w_0) = 1.$$

SMV's optimization problem



SMV's optimization problem

- Từ nhận xét, suy ra với tất cả các điểm x_n gần siêu phẳng phân cách nhất (hình vẽ), ta có thể giả thiết

$$y_n(\mathbf{w}^T\mathbf{x}_n + w_0) = 1.$$

- Như vậy, với mọi $n = 1, \ldots, N$, ta đều có

$$y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \geq 1.$$

- Do đó (1) có thể đưa về bài toán tối ưu có ràng buộc

$$(\mathbf{w}, w_0) = \arg\max_{\mathbf{w}, w_0} \frac{1}{\|\mathbf{w}\|_2}$$
$$\text{subject to:} \quad y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \geq 1, \forall n = 1, 2, \ldots, N.$$

SMV's optimization problem

- Qua biến đổi đơn giản, có thể viết bài toán tối ưu trong SVM dưới dạng

$$(\mathbf{w}, w_0) = \arg\min_{\mathbf{w}, w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to:} \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \leq 1, \forall n = 1, 2, \ldots, N. \tag{2}$$

- Bài toán (2) có
  - (i) Hàm mục tiêu $\|\mathbf{w}\|_2^2 = \mathbf{w}^T\mathbf{I}\mathbf{w}$ lồi chặt (*strictly convex*) do $\mathbf{I}$ xác định dương.
  - (ii) Bất đẳng thức ràng buộc là các hàm tuyến tính theo $\mathbf{w}, w_0$, do đó là hàm lồi.
- Vậy bài toán (2) là tối ưu lồi, chính xác hơn là dạng quy hoạch toàn phương (*Quadratic Programming* - QP), do đó có nghiệm (toàn cục) duy nhất.
- Việc giải bài toán Quadratic Programming được hỗ trợ trong nhiều thư viện khác nhau, ví dụ CVXOPT, CVXPY, Gurobi, MOSEK, qpOASES và quadprog là các thư viện giải QP cho *Python*.

SMV's optimization problem

- Trường hợp số chiều không gian dữ liệu d lớn hoặc số cặp dữ liệu N rất lớn, việc giải trực tiếp bài toán (2) rất phức tạp.
- Lúc đó thường sẽ cần sử dụng bài toán đối ngẫu để xử lý.
  - (i) Một số tính chất của bài toán đối ngẫu cho phép giải bài toán tối ưu hiệu quả hơn.
  - (ii) Sử dụng bài toán đối ngẫu, SVM có thể được áp dụng cho những bài toán mà dữ liệu không tách được tuyến tính (linearly separable).
- **Xác định class cho một điểm dữ liệu mới**: Sau khi tìm được siêu phẳng phân cách $\mathbf{w}^T\mathbf{x} + w_0 = 0$, điểm $\mathbf{x}$ sẽ được phân vào lớp

$$\text{class}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + w_0).$$

Optimization duality

- **Nhắc lại**: (2) là bài toán lồi, do đó
  (i) Nếu bài toán tối ưu lồi thoả mãn tiêu chuẩn Slater thì xảy ra *Strong Duality* - Đối ngẫu mạnh.
  (ii) Nếu bài toán tối ưu lồi có strong duality thì nghiệm của bài toán chính là nghiệm của hệ điều kiện KKT - *Karush–Kuhn–Tucker*.
- **Chú ý:**
  - Hệ KKT trở thành điều kiện đủ cho nghiệm tối ưu trong trường hợp này.
  - Kết hợp tính chất toàn cục và duy nhất của nghiệm bài toán (2), nếu kiểm tra được rằng (2) thỏa mãn tiêu chuẩn Slater (yếu) thì chỉ cần giải hệ KKT để tìm nghiệm tối ưu.

Slater's condition

- Để chuyển sang giải hệ KKT thay cho (2), cần chứng minh bài toán tối ưu (2) thoả mãn điều kiện Slater.
- **Điều kiện Slater cho** (2): Nếu tồn tại $\mathbf{w}, w_0$ thoả mãn:

$$1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) < 0, \quad \forall n = 1, 2, \ldots, N$$

thì xẩy ra strong duality.

Slater's condition

- **Bài toán** (2) **thỏa mãn tiêu chuẩn Slater. Thật vậy:**
  - Vì hai class là linearly separable, nên tồn tại siêu phẳng tách hai class. Vậy bài toán tồn tại nghiệm, và tập phương án (*feasible set*) của (2) khác rỗng.
  - Do đó luôn tồn tại cặp $(\bar{\mathbf{w}}, \bar{w}_0)$ sao cho

$$1 - y_n(\bar{\mathbf{w}}^T \mathbf{x}_n + \bar{w}_0) \leq 0, \forall n = 1, 2, \ldots, N$$
$$\Leftrightarrow 2 - y_n(2\bar{\mathbf{w}}^T \mathbf{x}_n + 2\bar{w}_0) \leq 0, \forall n = 1, 2, \ldots, N.$$

  - Vậy chỉ cần chọn $\tilde{\mathbf{w}} = 2\bar{\mathbf{w}}$, $\tilde{w}_0 = 2\bar{w}_0$, ta có

$$1 - y_n(\tilde{\mathbf{w}}^T \mathbf{x}_n + \tilde{w}_0) \leq -1 < 0, \quad \forall n = 1, 2, \ldots, N.$$

- Từ đó suy ra (2) thỏa mãn điều kiện Slater.

SVM's Lagrangian

- **Lagrangian của bài toán** (2)**:**

$$\mathcal{L}(\mathbf{w}, w_0, \lambda) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{n=1}^{N} \lambda_n (1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)) \tag{3}$$

ở đây vector nhân tử Lagrange $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T$ có $\lambda_n \geq 0$, $\forall n = \overline{1..N}$.

- Hàm đối ngẫu Lagrange:

$$g(\lambda) = \min_{\mathbf{w}, w_0} \mathcal{L}(\mathbf{w}, w_0, \lambda) \quad \text{với} \quad \lambda \geq 0. \tag{4}$$

- Chú ý tính tồn tại duy nhất nghiệm bài toán gốc, việc tìm cực tiểu $\mathcal{L}(\mathbf{w}, w_0, \lambda)$ theo $\mathbf{w}$, w_0 có thể thực hiện qua việc giải hệ: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$; $\frac{\partial \mathcal{L}}{\partial w_0} = 0$.

SVM's Lagrangian

- Cực tiểu $\mathcal{L}(\mathbf{w}, w_0, \lambda)$ theo $\mathbf{w}$, w_0 thông qua giải hệ đạo hàm bằng 0:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n \tag{5}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \lambda)}{\partial w_0} = -\sum_{n=1}^{N} \lambda_n y_n = 0 \tag{6}$$

- Thay (5) và (6) vào (3), sau đó áp dụng (4) ta viết được $g(\lambda)$ dưới dạng

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m. \tag{7}$$

SVM's Lagrangian

- Đặt vector $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^N$ và ma trận

$$\mathbf{V} = (y_1\mathbf{x}_1, y_2\mathbf{x}_2, \ldots, y_N\mathbf{x}_N)$$

(chú ý $\mathbf{x}_n \in \mathbb{R}^d$ là các vector cột) ta có thể viết lại $g(\lambda)$ dưới dạng ma trận

$$g(\lambda) = -\frac{1}{2}\lambda^T\mathbf{V}^T\mathbf{V}\lambda + \mathbf{1}^T\lambda. \tag{8}$$

- Đặt ma trận $\mathbf{K} = \mathbf{V}^T\mathbf{V}$, với mọi λ ta có $\lambda^T\mathbf{K}\lambda = \lambda^T\mathbf{V}^T\mathbf{V}\lambda = ||\mathbf{V}\lambda||_2^2 \geq 0$ do đó $\mathbf{K}$ là ma trận nửa xác định dương.
- Do vậy $g(\lambda) = -\frac{1}{2}\lambda^T\mathbf{K}\lambda + \mathbf{1}^T\lambda$ là một hàm lõm (*concave*).

SVM's Lagrangian

Kết hợp hàm đối ngẫu và các điều kiện đối với λ, ta có bài toán đối ngẫu Lagrange

$$\lambda = \arg\max_{\lambda} g(\lambda) \quad \text{subject to:} \quad \lambda_n \geq 0 \quad \forall n = \overline{1..N}; \quad \sum_{n=1}^{N} \lambda_n y_n = 0. \tag{9}$$

- Đây là bài toán tìm cực đại của hàm lõm trên một đa diện lồi (covex polyhedron), do đó nó cũng là một bài toán lồi và chuyển sang dạng tìm cực tiểu, ta cũng được một bài toán dạng quy hoạch toàn phương.
- Trong bài toán đối ngẫu, số tham số phải tìm là N (số chiều của λ), tức số điểm dữ liệu, trong rất nhiều trường hợp, sẽ lớn hơn số chiều dữ liệu rất nhiều.
- Vậy nếu giải trực tiếp bằng các công cụ giải Quadratic Programming, có thể bài toán đối ngẫu còn phức tạp hơn (tốn thời gian hơn) so với bài toàn gốc.

SVM's Lagrangian

- Tuy nhiên dựa vào tính chất hệ điều kiện KKT mà SVM có thể được giải bằng nhiều phương pháp hiệu quả hơn.
- Do bài toán (2) lồi và thỏa mãn strong duality, nghiệm $\mathbf{w}, w_0, \lambda$ của nó thoả mãn hệ điều kiện KKT sau

$$1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \leq 0; \ \lambda_n \geq 0; \tag{10}$$

$$\lambda_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0)) = 0, \ \forall n = 1, 2, \ldots, N \tag{11}$$

$$\mathbf{w} = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n; \tag{12}$$

$$\sum_{n=1}^{N} \lambda_n y_n = 0. \tag{13}$$

SVM's Lagrangian

- Chú ý từ điều kiện $\lambda_n \geq 0$ và điều kiện $\lambda_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0)) = 0$ suy ra với $n = 1, 2, \ldots, N$ bất kỳ, hoặc

$$1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) = 0; \quad \text{hoặc} \quad \lambda_n = 0.$$

- Chú ý $y_n^2 = 1$ với mọi $n = 1, \ldots, N$, nên trường hợp thứ nhất chính là:

$$\mathbf{w}^T\mathbf{x}_n + w_0 = y_n = \pm 1 \tag{14}$$

- Các cặp $(\mathbf{x}_n, y_n)$ thỏa mãn (14) sẽ là "điểm tựa" của hai "siêu phẳng lề" $\mathbf{w}^T\mathbf{x}_n + w_0 = \pm 1$.
- Các "điểm tựa" nói trên là các vector xác định nghiệm của bài toán tối ưu trong phương pháp, do đó phương pháp này được gọi là *Support Vector Machine*

SVM's Lagrangian

- Thực tế cho thấy số lượng điểm tựa thường rất ít so với toàn tập dữ liệu, tức số lượng cặp $(\mathbf{x}_n, y_n)$ thỏa mãn (14) rất nhỏ.
  (Do nếu ngược lại thì tập dữ liệu sẽ tập trung trên hai siêu phẳng lề).
- Suy ra với hầu hết $n \in \{1, 2, \ldots, N\}$, $\lambda_n = 0$. Tức là vector λ là một *vector thưa - sparse vector*.
- Đây là lý do Support Vector Machine còn được xếp vào *Sparse Models*. Có rất nhiều phương pháp giải hiệu quả cho các Sparse Model.
- Cũng vì vậy, bài toán đối ngẫu của SVM thường được quan tâm nhiều hơn là bài toán gốc.
- Tuy vậy việc giải trực tiếp hệ KKT (10)-(13) với N lớn là không khả thi (sẽ cần xét 2^N trường hợp). Do đó ta thường tìm λ từ (9), sau đó kết hợp với các biểu thức (12) để xác định $\mathbf{w}$; cuối cùng dùng (11) và (13) để tính w_0.

SVM's Lagrangian

- Đặt $\mathcal{S} = \{n : \lambda_n \neq 0\}$ và $N_S = |\mathcal{S}|$. Lúc đó với mỗi $n \in \mathcal{S}$ ta có

$$1 = y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \Leftrightarrow w_0 + \mathbf{w}^T\mathbf{x}_n = y_n$$

- Vậy chỉ cần 01 cặp $(\mathbf{x}_n, y_n)$, với $n \in \mathcal{S}$, là đủ để ta có thể tính được w_0.
- Một cách tính w_0 *ổn định hơn trong tính toán* (*numerically more stable*) và thường được dùng là

$$w_0 = \frac{1}{N_S}\sum_{n\in\mathcal{S}}(y_n - \mathbf{w}^T\mathbf{x}_n) = \frac{1}{N_S}\sum_{n\in\mathcal{S}}\left(y_n - \sum_{m\in\mathcal{S}}\lambda_m y_m \mathbf{x}_m^T\mathbf{x}_n\right) \qquad (15)$$

tức trung bình cộng của các biểu thức tính w_0 ứng với mỗi $\mathbf{x}_n \in \mathcal{S}$.

SVM's Lagrangian

- Lưu ý từ (12) ta đã có

$$\mathbf{w} = \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m. \tag{16}$$

- Cuối cùng, để xác định một điểm $\mathbf{x}$ mới thuộc vào class nào, ta cần xác định dấu của biểu thức

$$\mathbf{w}^T\mathbf{x} + w_0 = \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T\mathbf{x} + \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T\mathbf{x}_n \right).$$

- Biểu thức này phụ thuộc vào cách tính tích vô hướng giữa các cặp vector $\mathbf{x}$ và từng $\mathbf{x}_n \in \mathcal{S}$.

Python code Example

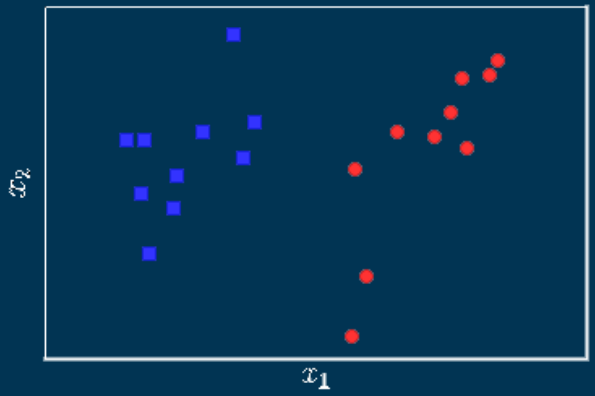
- Có thể sử dụng hai cách sau để giải bài toán tối ưu của SVM
  - Sử dụng thư viện sklearn;
  - Lập trình trực tiếp một số bước giải.
- Ta giới thiệu cả hai cách cho ví dụ nhân tạo với dữ liệu được tạo như sau

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

means = [[2, 2], [4, 2]]
cov = [[.3, .2], [.2, .3]]
N = 10
X0 = np.random.multivariate_normal(means[0], cov, N) # class 1
X1 = np.random.multivariate_normal(means[1], cov, N) # class -1
X = np.concatenate((X0.T, X1.T), axis = 1) # all data
y = np.concatenate((np.ones((1, N)), -1*np.ones((1, N))), axis = 1) # labels
```

Python code Example

Dữ liệu tạo từ đoạn code trên được minh họa trong hình sau



Python code Example

Đoạn code giải bài toán (9) bằng CVXOPT

```
from cvxopt import matrix, solvers
# build K
V = np.concatenate((X0.T, -X1.T), axis = 1)
K = matrix(V.T.dot(V)) # see definition of V, K near eq (8)

p = matrix(-np.ones((2*N, 1))) # all-one vector
# build A, b, G, h
G = matrix(-np.eye(2*N)) # for all lambda_n >= 0
h = matrix(np.zeros((2*N, 1)))
A = matrix(y) # the equality constrain is actually y^T lambda = 0
b = matrix(np.zeros((1, 1)))
solvers.options['show_progress'] = False
sol = solvers.qp(K, p, G, h, A, b)

l = np.array(sol['x']) # solve for lambda, assign to l
```

Python code Example

- Ta lấy ngưỡng với $\epsilon = 10^{-7}$ để loại bỏ các giá trị λ_n quá bé (thực chất ở đó theo lý thuyết $\lambda_n = 0$, giá trị rất bé thu được là do sai số tính toán).
- Vector λ thu được như sau

$$\lambda = l = \begin{pmatrix} 0.854018321 & 0.0 & 1.37095535 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 2.22497367 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

- Có 03 giá trị $\lambda_n \neq 0$, vậy có thể dự đoán sẽ có 03 điểm là support vectors.

Python code Example

- Đoạn code dưới đây xác định tập $\mathcal{S} = \{n : \lambda_n \neq 0\}$ sau đó tính $\mathbf{w}, w_0$ từ (12) và (15)

```
epsilon = 1e-7 # threshold - a small number, greater than 1e-9
S = np.where(l > epsilon)[0]

VS = V[:, S]
XS = X[:, S]
yS = y[:, S]
lS = l[S]
# calculate w and w_0
w = VS.dot(lS)
w_0 = np.mean(yS.T - w.T.dot(XS))
```

- Kết quả thu được

$$\mathbf{w} = \begin{pmatrix} - & 2.00984381 \\ & 0.64068336 \end{pmatrix}; \quad w_0 = 4.66856063387.$$

Python code Example

- Dưới đây ta sẽ sử dụng hàm sklearn.svm.SVC thư viện của thư viện sklearn

```
from sklearn.svm import SVC

y1 = y.reshape((2*N,))
X1 = X.T # each sample is one row
clf = SVC(kernel = 'linear', C = 1e6) # just a big number

clf.fit(X1, y1)

w = clf.coef_
w0 = clf.intercept_
```

- Kết quả thu được $\mathbf{w} = \begin{pmatrix} - & 2.00971102 \\ & 0.64194082 \end{pmatrix}$; $w_0 = 4.66595309$. Kết quả tính theo công thức và kết quả với tính toán dựa trên thư viện khá gần nhau.
- Phần code trên có thể thử trên Jupyter Notebook với Python 3. Thực tế người ta thường sử dụng thư viện libsvm được viết trên ngôn ngữ C, có API cho C, Matlab, Python...

Summarization

- Bài toán binary classification với hai classes là linearly separable, có vô số các siêu phẳng tách hai classes. Với mỗi siêu phẳng tách, ta có một classifier. Khoảng cách gần nhất từ 01 điểm dữ liệu tới siêu phẳng tách là *margin* của classifier đó.
- Support Vector Machine dẫn đến việc tìm siêu phẳng tách sao cho margin tương ứng là lớn nhất. Ý nghĩa: các điểm dữ liệu an toàn nhất so với siêu phẳng tách.
- Bài toán tối ưu từ SVM là lồi với hàm mục tiêu stricly convex, nghiệm của nó tồn tại duy nhất. Hơn nữa, bài toán tối ưu đó có dạng quy hoạch toàn phương (QP).
- Người ta thường giải bài toán đối ngẫu: đây cũng là một QP nhưng nghiệm là thưa (*sparse*) nên có những phương pháp giải hiệu quả hơn.